THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.
IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY

designated nationals list is strictly prohibited.

or reexport to countries subject to U.S. embargo or to entities identified on U.S. export exclusion lists, including, but not limited to, the denied persons and specially
other countries. Nuclear, missile, chemical or biological weapons or nuclear maritime end uses or end users, whether direct or indirect, are strictly prohibited. Export
agreements.

the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPENLOOK GUIs and otherwise comply with Sun's written license

agreements.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other
countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, the Solaris logo, the Java Coffee Cup logo, docs.sun.com, SunOS, JumpStart, Trusted Solaris, OpenSolaris, Java, and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. or its subsidiaries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc. Xylogin product is protected by copyright and licensed to Sun by Xylogics. Xylogin and Annex are trademarks of Xylogin, Inc. Portions of the software copyright 1996 by the Massachusetts Institute of Technology. All rights reserved.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts
of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to
the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPENLOOK GUIs and otherwise comply with Sun's written license agreements.

Products covered by and information contained in this publication are controlled by U.S. Export Control laws and may be subject to the export or import laws in
other countries. Nuclear, missile, chemical or biological weapons or nuclear maritime end uses or end users, whether direct or indirect, are strictly prohibited. Export
or reexport to countries subject to U.S. embargo or to entities identified on U.S. export exclusion lists, including, but not limited to, the denied persons and specially
designated nationals lists is strictly prohibited.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright 2010 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. Tous droits réservés.

Sun Microsystems, Inc. détient les droits de propriété intellectuelle relatifs à la technologie incorporée dans le produit qui est décrit dans ce document. En particulier,
et ce sans limitation, ces droits de propriété intellectuelle peuvent inclure un ou plusieurs brevets américains ou des applications de brevet en attente aux États-Unis
dans d'autres pays.

Cette distribution peut comprendre des composants développés par des tierces personnes.

Certaines composants de ce produit peuvent être dérivées du logiciel Berkeley BSD, licencié par l'Université de Californie. UNIX est une marque déposée aux
États-Unis et dans d'autres pays; elle est licencie exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, le logo Solaris, le logo Java Coffee Cup, docs.sun.com, SunOS, JumpStart, Trusted Solaris, OpenSolaris, Java et Solaris sont des
marques de fabrique ou des marques déposées de Sun Microsystems, Inc., ou ses filiales, aux États-Unis et dans d'autres pays. Toutes les marques SPARC sont
utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux États-Unis et dans d'autres pays. Les produits
portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc. Le produit de Xylogin est protégé par le copyright et autorisé au
Sun par Xylogics. Xylogin et Annex sont des marques déposées de Xylogin, Inc. Copyright 1996 des portions du logiciel par Massachusetts Institute of Technology.
Tous droits réservés.

L'interface d'utilisation graphique OPEN LOOK et Sun a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de
pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient
une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface
d'utilisation graphique OPEN LOOK et qui, en outre, se conforment aux licences écrites de Sun.

Les produits qui font l'objet de cette publication et les informations qu'il contient sont régis par la législation américaine en matière de contrôle des exportations et
can être soumis au droit d'autres pays dans le domaine des exportations et importations. Les utilisations finales, ou utilisateurs finaux, pour des armes nucléaires,
des missiles, des armes chimiques ou biologiques ou pour le nucléaire maritime, directement ou indirectement, sont strictement interdites. Les exportations ou
réexportations vers des pays sous embargo des États-Unis, ou vers des entités figurant sur les listes d'exclusion d'exportation américaines, y compris, mais de manière
non exclusive, la liste de personnes qui sont object d'un ordre de ne pas participer, d'une façon directe ou indirecte, aux exportations des produits ou des services qui
sont régis par la législation américaine en matière de contrôle des exportations et la liste de rassortissants spécifiquement désignés, sont rigoureusement interdites.

LA DOCUMENTATION EST FOURNIE "EN L'ETAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTorisée PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE À LA QUALITÉ MARCHANDE, À L'APTITUDE À UNE UTILISATION PARTICULIERE OU À L'ABSENCE DE CONTREFAÇON.
# Contents

Preface ................................................................................................................................................... 25

Part I Security Overview ............................................................................................................................... 29

1 Security Services (Overview) ..................................................................................................................... 31
   System Security .................................................................................................................................... 32
   Solaris Cryptographic Services .......................................................................................................... 33
   Authentication Services ...................................................................................................................... 34
   Authentication With Encryption ...................................................................................................... 34
   Solaris Auditing ................................................................................................................................... 35
   Security Policy ................................................................................................................................... 35

Part II System, File, and Device Security ..................................................................................................... 37

2 Managing Machine Security (Overview) ..................................................................................................... 39
   Controlling Access to a Computer System ....................................................................................... 39
      Maintaining Physical Security ........................................................................................................ 40
      Maintaining Login Control ............................................................................................................ 40
   Controlling Access to Devices ......................................................................................................... 45
      Device Policy (Overview) .............................................................................................................. 46
      Device Allocation (Overview) .................................................................................................... 47
   Controlling Access to Machine Resources ....................................................................................... 47
      Limiting and Monitoring Superuser .............................................................................................. 47
      Configuring Role-Based Access Control to Replace Superuser ............................................. 48
      Preventing Unintentional Misuse of Machine Resources ....................................................... 48
      Restricting setuid Executable Files ........................................................................................... 49
      Using the Solaris Security Toolkit .............................................................................................. 50
Using the netsevices limited Configuration .................................................................50
Using Solaris Resource Management Features .......................................................... 50
Using Solaris Zones .................................................................................................51
Monitoring Use of Machine Resources .................................................................51
Monitoring File Integrity .....................................................................................51
Controlling Access to Files .........................................................................................52
  Protecting Files With Encryption ...........................................................................52
  Using Access Control Encryption .........................................................................52
  Sharing Files Across Machines .............................................................................53
  Restricting root Access to Shared Files ..................................................................53
Controlling Network Access ......................................................................................53
  Network Security Mechanisms .............................................................................54
  Authentication and Authorization for Remote Access ...........................................54
  Firewall Systems ..................................................................................................56
  Encryption and Firewall Systems ..........................................................................57
Reporting Security Problems ....................................................................................57

### 3 Controlling Access to Systems (Tasks)

- Controlling System Access (Task Map) .................................................................59
- Securing Logins and Passwords (Task Map) .......................................................60
- Securing Logins and Passwords ...........................................................................60
  - ▼ How to Display a User's Login Status ..............................................................60
  - ▼ How to Display Users Without Passwords ......................................................61
  - ▼ How to Temporarily Disable User Logins .......................................................62
  - ▼ How to Monitor Failed Login Attempts ............................................................63
  - ▼ How to Monitor All Failed Login Attempts .....................................................64
  - ▼ How to Create a Dial-Up Password ..................................................................65
  - ▼ How to Temporarily Disable Dial-Up Logins ..................................................67
- Changing the Password Algorithm (Task Map) ..................................................67
- Changing the Default Algorithm for Password Encryption .....................................68
  - ▼ How to Specify an Algorithm for Password Encryption ....................................68
  - ▼ How to Specify a New Password Algorithm for an NIS Domain ....................69
  - ▼ How to Specify a New Password Algorithm for an NIS+ Domain ..................69
  - ▼ How to Specify a New Password Algorithm for an LDAP Domain ...............70
  - ▼ How to Install a Password Encryption Module From a Third Party .................70
Contents

6 Using the Basic Audit Reporting Tool (Tasks) .......................................................... 105
   Basic Audit Reporting Tool (Overview) ........................................................................ 105
      BART Features ........................................................................................................... 106
      BART Components .................................................................................................. 106
   Using BART (Task Map) ............................................................................................. 108
   Using BART (Tasks) ..................................................................................................... 109
      BART Security Considerations ................................................................................ 109
      ▼ How to Create a Manifest ....................................................................................... 109
      ▼ How to Customize a Manifest .................................................................................. 112
      ▼ How to Compare Manifests for the Same System Over Time .......................... 115
      ▼ How to Compare Manifests From a Different System With the Manifest of a Control System .................................................................................................................. 117
      ▼ How to Customize a BART Report by Specifying File Attributes ..................... 120
      ▼ How to Customize a BART Report by Using a Rules File .................................. 121
   BART Manifest, Rules File, and Reporting (Reference) ............................................... 122
      BART Manifest File Format ...................................................................................... 123
      BART Rules File Format .......................................................................................... 124
      BART Reporting ....................................................................................................... 125

7 Controlling Access to Files (Tasks) ........................................................................... 127
   Using UNIX Permissions to Protect Files .................................................................... 127
      Commands for Viewing and Securing Files .............................................................. 127
      File and Directory Ownership .................................................................................. 128
      UNIX File Permissions ............................................................................................. 128
Special File Permissions (setuid, setgid and Sticky Bit) ..................................................... 129
Default umask Value ............................................................................................................. 131
File Permission Modes ........................................................................................................ 131
Using Access Control Lists to Protect Files ........................................................................ 133
ACL Entries for Files .......................................................................................................... 134
ACL Entries for Directories ................................................................................................. 135
Commands for Administering ACLs .................................................................................. 135
Preventing Executable Files From Compromising Security ................................................ 136
Protecting Files With UNIX Permissions (Task Map) ......................................................... 137
 ▼ How to Display File Information .................................................................................... 137
 ▼ How to Change the Owner of a File ................................................................................ 139
 ▼ How to Change Group Ownership of a File ................................................................. 140
 ▼ How to Change File Permissions in Symbolic Mode .................................................... 140
 ▼ How to Change File Permissions in Absolute Mode ..................................................... 141
 ▼ How to Change Special File Permissions in Absolute Mode ........................................ 142
Protecting Files With ACLs (Task Map) ............................................................................ 143
 ▼ How to Check if a File Has an ACL ............................................................................... 144
 ▼ How to Add ACL Entries to a File ............................................................................... 144
 ▼ How to Copy an ACL .................................................................................................... 146
 ▼ How to Change ACL Entries on a File .......................................................................... 146
 ▼ How to Delete ACL Entries From a File ........................................................................ 147
 ▼ How to Display ACL Entries for a File .......................................................................... 148
Protecting Against Programs With Security Risk (Task Map) ............................................. 149
 ▼ How to Find Files With Special File Permissions ......................................................... 149
 ▼ How to Disable Programs From Using Executable Stacks ............................................ 150

Part III Roles, Rights Profiles, and Privileges ........................................................................ 153

8 Using Roles and Privileges (Overview) ........................................................................... 155
Role-Based Access Control (Overview) ............................................................................. 155
RBAC: An Alternative to the Superuser Model .................................................................... 155
Solaris RBAC Elements and Basic Concepts ..................................................................... 157
RBAC Authorizations .......................................................................................................... 160
Authorizations and Privileges ............................................................................................ 160
10 Role-Based Access Control (Reference) ................................................................. 205

Contents of Rights Profiles ..................................................................................... 205
  Primary Administrator Rights Profile ........................................................................ 206
  System Administrator Rights Profile .......................................................................... 206
  Operator Rights Profile ............................................................................................ 207
  Printer Management Rights Profile .......................................................................... 207
  Basic Solaris User Rights Profile .............................................................................. 208
  Console User Rights Profile ..................................................................................... 209
  All Rights Profile ..................................................................................................... 209
  Order of Rights Profiles .......................................................................................... 209
  Viewing the Contents of Rights Profiles ................................................................... 209

Authorization Naming and Delegation .................................................................. 210
  Authorization Naming Conventions ......................................................................... 210
  Example of Authorization Granularity ..................................................................... 210
  Delegation Authority in Authorizations .................................................................. 211

Databases That Support RBAC .................................................................................. 211
  RBAC Database Relationships .................................................................................. 211
  RBAC Databases and the Name Service .................................................................... 212
  user_attr Database .................................................................................................. 213
  auth_attr Database .................................................................................................. 214
  prof_attr Database .................................................................................................. 215
  exec_attr Database .................................................................................................. 216
  policy.conf File ......................................................................................................... 217

RBAC Commands ...................................................................................................... 218
  Commands That Manage RBAC ................................................................................ 218
  Commands That Require Authorizations .................................................................. 219

11 Privileges (Tasks) .................................................................................................. 221

Managing and Using Privileges (Task Map) ............................................................. 221
Managing Privileges (Task Map) .............................................................................. 221
Managing Privileges .................................................................................................. 222
15 Solaris Key Management Framework ................................................................. 279
Managing Public Key Technologies .................................................................... 279
Key Management Framework Utilities ............................................................... 280
KMF Policy Management .................................................................................. 280
KMF Plugin Management .................................................................................. 280
KMF Keystore Management ............................................................................. 281
Using the Key Management Framework (Task Map) ........................................... 281
Using the Key Management Framework (Tasks) ................................................ 282
▼ How to Create a Certificate by Using the pktool gencert Command ............... 282
▼ How to Import a Certificate Into Your Keystore ........................................... 283
▼ How to Export a Certificate and Private Key in PKCS #12 Format ................. 285
▼ How to Generate a Passphrase by Using the pktool setpin Command .............. 286
▼ How to Manage Third-Party Plugins in KMF ................................................. 287

Part V Authentication Services and Secure Communication .................................. 289

16 Using Authentication Services (Tasks) ............................................................. 291
Overview of Secure RPC ................................................................................... 291
Contents

NFS Services and Secure RPC ................................................................. 291
DES Encryption With Secure NFS .......................................................... 292
Kerberos Authentication .............................................................................. 292
Diffie-Hellman Authentication and Secure RPC ...................................... 292
Administering Secure RPC (Task Map) ...................................................... 296
Administering Authentication With Secure RPC ...................................... 296
▼ How to Restart the Secure RPC Keyservers .............................................. 297
▼ How to Set Up a Diffie-Hellman Key for an NIS+ Host.......................... 297
▼ How to Set Up a Diffie-Hellman Key for an NIS+ User......................... 298
▼ How to Set Up a Diffie-Hellman Key for an NIS Host ......................... 299
▼ How to Set Up a Diffie-Hellman Key for an NIS User ......................... 300
▼ How to Share NFS Files With Diffie-Hellman Authentication ............... 301

17 Using PAM ....................................................................................... 303
PAM (Overview) .................................................................................... 303
Benefits of Using PAM ........................................................................... 303
Introduction to the PAM Framework ...................................................... 304
Changes to PAM for the Solaris 10 Release ............................................ 305
Changes to PAM for the Solaris Express 1/06 Release ......................... 306
PAM (Tasks) ......................................................................................... 306
PAM (Task Map) .................................................................................... 307
Planning for Your PAM Implementation ............................................... 307
▼ How to Add a PAM Module ............................................................... 308
▼ How to Prevent Rhost-Style Access From Remote Systems With PAM ... 309
▼ How to Log PAM Error Reports ....................................................... 309
PAM Configuration (Reference) .............................................................. 309
PAM Configuration File Syntax ............................................................. 310
How PAM Stacking Works .................................................................... 310
PAM Stacking Example .......................................................................... 314

18 Using SASL ...................................................................................... 317
SASL (Overview) ................................................................................... 317
SASL (Reference) .................................................................................. 318
SASL Plug-ins ......................................................................................... 318
SASL Environment Variable ................................................................... 318
19 Using Solaris Secure Shell (Tasks) ................................................................. 321
Solaris Secure Shell (Overview) ..................................................................... 321
  Solaris Secure Shell Authentication .............................................................. 322
  Solaris Secure Shell in the Enterprise .......................................................... 324
Solaris Secure Shell and the OpenSSH Project ........................................... 324
Solaris Secure Shell (Task Map) .................................................................... 325
Configuring Solaris Secure Shell (Task Map) ............................................... 325
Configuring Solaris Secure Shell .................................................................. 326
  ▼ How to Set Up Host-Based Authentication for Solaris Secure Shell ........... 326
  ▼ How to Enable Solaris Secure Shell v1 ..................................................... 328
  ▼ How to Configure Port Forwarding in Solaris Secure Shell ................... 329
  ▼ How to Create User and Host Exceptions to SSH System Defaults ........ 330
Using Solaris Secure Shell (Task Map) .......................................................... 331
Using Solaris Secure Shell .............................................................................. 332
  ▼ How to Generate a Public/Private Key Pair for Use With Solaris Secure Shell .... 332
  ▼ How to Change the Passphrase for a Solaris Secure Shell Private Key ....... 334
  ▼ How to Log In to a Remote Host With Solaris Secure Shell .................. 335
  ▼ How to Reduce Password Prompts in Solaris Secure Shell ................... 336
  ▼ How to Use Port Forwarding in Solaris Secure Shell ............................... 337
  ▼ How to Copy Files With Solaris Secure Shell ........................................ 339
  ▼ How to Set Up Default Connections to Hosts Outside a Firewall ........... 340

20 Solaris Secure Shell (Reference) ................................................................. 343
A Typical Solaris Secure Shell Session ......................................................... 343
  Session Characteristics in Solaris Secure Shell .......................................... 344
  Authentication and Key Exchange in Solaris Secure Shell ......................... 344
  Command Execution and Data Forwarding in Solaris Secure Shell .......... 345
Client and Server Configuration in Solaris Secure Shell ............................ 346
  Client Configuration in Solaris Secure Shell .............................................. 346
  Server Configuration in Solaris Secure Shell ............................................. 346
Keywords in Solaris Secure Shell ................................................................. 346
  Host-Specific Parameters in Solaris Secure Shell ....................................... 350
  Solaris Secure Shell and Login Environment Variables .............................. 350
23 Configuring the Kerberos Service (Tasks) ................................................................. 387

Configuring the Kerberos Service (Task Map) .......................................................... 387
Configuring Additional Kerberos Services (Task Map) ............................................. 388
Configuring KDC Servers ....................................................................................... 388
  ▼ How to Automatically Configure a Master KDC .................................................. 389
  ▼ How to Interactively Configure a Master KDC .................................................... 390
  ▼ How to Manually Configure a Master KDC ....................................................... 391
  ▼ How to Configure a KDC to Use an LDAP Data Server ..................................... 396
  ▼ How to Automatically Configure a Slave KDC ................................................... 402
  ▼ How to Interactively Configure a Slave KDC ..................................................... 403
  ▼ How to Manually Configure a Slave KDC .......................................................... 404
  ▼ How to Refresh the Ticket Granting Service Keys on a Master Server ................. 408
Configuring Cross-Realm Authentication ............................................................... 409
  ▼ How to Establish Hierarchical Cross-Realm Authentication .............................. 409
  ▼ How to Establish Direct Cross-Realm Authentication ........................................ 410
Configuring Kerberos Network Application Servers ................................................. 411
  ▼ How to Configure a Kerberos Network Application Server ................................. 411
Configuring Kerberos NFS Servers ....................................................................... 413
  ▼ How to Configure Kerberos NFS Servers ........................................................... 414
### Contents

- **How to Create a Credential Table** ................................................................. 415
- **How to Add a Single Entry to the Credential Table** ......................................... 416
- **How to Provide Credential Mapping Between Realms** ..................................... 417
- **How to Set Up a Secure NFS Environment With Multiple Kerberos Security Modes** 418

**Configuring Kerberos Clients** .................................................................................. 419

- **Configuring Kerberos Clients (Task Map)** ......................................................... 420
- **How to Create a Kerberos Client Installation Profile** .......................................... 420
- **How to Automatically Configure a Kerberos Client** ........................................... 421
- **How to Interactively Configure a Kerberos Client** .............................................. 422
- **How to Configure a Kerberos Client for an Active Directory Server** ................... 426
- **How to Manually Configure a Kerberos Client** .................................................. 427
- **How to Disable Verification of the Ticket Granting Ticket (TGT)** ....................... 432
- **How to Access a Kerberos Protected NFS File System as the root User** .............. 433
- **How to Configure Automatic Migration of Users in a Kerberos Realm** .............. 434

**Synchronizing Clocks Between KDCs and Kerberos Clients** ................................ 437

- **Swapping a Master KDC and a Slave KDC** ......................................................... 438
- **How to Configure a Swappable Slave KDC** ....................................................... 438
- **How to Swap a Master KDC and a Slave KDC** ................................................... 439

**Administering the Kerberos Database** ..................................................................... 443

- **How to Back Up the Kerberos Database** .............................................................. 443
- **How to Restore the Kerberos Database** ............................................................... 445
- **How to Convert a Kerberos Database After a Server Upgrade** ............................ 446
- **How to Reconfigure a Master KDC to Use Incremental Propagation** ................. 447
- **How to Reconfigure a Slave KDC to Use Incremental Propagation** ..................... 449
- **How to Configure a Slave KDC to Use Full Propagation** .................................... 450
- **How to Verify That the KDC Servers Are Synchronized** ..................................... 453
- **How to Manually Propagate the Kerberos Database to the Slave KDCs** .............. 454

**Setting Up Parallel Propagation** ............................................................................. 455

- **Configuration Steps for Setting Up Parallel Propagation** ................................ 456
- **Administering the Stash File** .............................................................................. 457
- **How to Remove a Stash File** .............................................................................. 457

**Managing a KDC on an LDAP Directory Server** .................................................... 457

- **How to Mix Kerberos Principal Attributes in a Non-Kerberos Object Class Type** .... 458
- **How to Destroy a Realm on an LDAP Directory Server** ..................................... 459

**Increasing Security on Kerberos Servers** ............................................................... 459
Contents

24 Kerberos Error Messages and Troubleshooting ................................................................. 463
   Kerberos Error Messages ............................................................................................... 463
      SEAM Administration Tool Error Messages ............................................................ 463
      Common Kerberos Error Messages (A-M) ................................................................. 464
      Common Kerberos Error Messages (N-Z) ................................................................. 472
   Kerberos Troubleshooting ............................................................................................ 475
      Problems With the Format of the krb5.conf File ...................................................... 475
      Problems Propagating the Kerberos Database ......................................................... 475
      Problems Mounting a Kerberized NFS File System ................................................ 476
      Problems Authenticating as root ............................................................................. 477
      Observing Mapping from GSS Credentials to UNIX Credentials ............................. 477

25 Administering Kerberos Principals and Policies (Tasks) .................................................. 479
   Ways to Administer Kerberos Principals and Policies ................................................ 479
   SEAM Administration Tool .......................................................................................... 480
      Command-Line Equivalents of the SEAM Tool ....................................................... 480
      The Only File Modified by the SEAM Tool ............................................................. 481
      Print and Online Help Features of the SEAM Tool ................................................. 481
      Working With Large Lists in the SEAM Tool .......................................................... 482
   How to Start the SEAM Tool ....................................................................................... 482
   Administering Kerberos Principals .............................................................................. 483
      Administering Kerberos Principals (Task Map) ....................................................... 483
      Automating the Creation of New Kerberos Principals .......................................... 484
      How to View the List of Kerberos Principals ........................................................... 485
      How to View a Kerberos Principal’s Attributes ....................................................... 487
      How to Create a New Kerberos Principal ............................................................... 489
      How to Duplicate a Kerberos Principal .................................................................... 492
      How to Modify a Kerberos Principal ...................................................................... 493
      How to Delete a Kerberos Principal ....................................................................... 494
      How to Set Up Defaults for Creating New Kerberos Principals ............................. 494
      How to Modify the Kerberos Administration Privileges ........................................ 495
Contents

Administering Kerberos Policies ................................................................. 497
   Administering Kerberos Policies (Task Map) .................................................. 497
   ▼ How to View the List of Kerberos Policies .................................................... 497
   ▼ How to View a Kerberos Policy's Attributes .............................................. 499
   ▼ How to Create a New Kerberos Policy ....................................................... 501
   ▼ How to Duplicate a Kerberos Policy ......................................................... 503
   ▼ How to Modify a Kerberos Policy .............................................................. 503
   ▼ How to Delete a Kerberos Policy ............................................................... 504
SEAM Tool Reference ................................................................. 505
   SEAM Tool Panel Descriptions ................................................................. 505
   Using the SEAM Tool With Limited Kerberos Administration Privileges .......... 508
Administering Keytab Files ................................................................. 509
   Administering Keytab Files (Task Map) ....................................................... 510
   ▼ How to Add a Kerberos Service Principal to a Keytab File ...................... 510
   ▼ How to Remove a Service Principal From a Keytab File ....................... 512
   ▼ How to Display the Keylist (Principals) in a Keytab File ......................... 513
   ▼ How to Temporarily Disable Authentication for a Service on a Host ............. 513

26 Using Kerberos Applications (Tasks) ............................................................... 517
Kerberos Ticket Management ........................................................................ 517
   Do You Need to Worry About Tickets? ......................................................... 517
Creating a Kerberos Ticket ........................................................................... 518
Viewing Kerberos Tickets ............................................................................. 519
Destroying Kerberos Tickets ......................................................................... 520
Kerberos Password Management .................................................................... 521
   Advice on Choosing a Password ............................................................... 521
Changing Your Password ............................................................................ 522
Granting Access to Your Account ............................................................... 524
Kerberos User Commands ............................................................................ 526
   Overview of Kerberized Commands .......................................................... 526
Forwarding Kerberos Tickets ........................................................................ 529
Using Kerberized Commands (Examples) ..................................................... 530

27 The Kerberos Service (Reference) ............................................................... 533
Kerberos Files .............................................................................................. 533
Part VII Solaris Auditing ................................................................. 549

28 Solaris Auditing (Overview) ..................................................... 551
   What Is Auditing? ................................................................. 551
   How Does Auditing Work? .................................................. 553
   How Is Auditing Related to Security? .............................. 554
   Audit Terminology and Concepts ...................................... 554
      Audit Events ........................................................................ 555
      Audit Classes and Preselection ......................................... 556
      Audit Records and Audit Tokens ...................................... 557
      Audit Plugin Modules ......................................................... 558
      Audit Logs ........................................................................ 558
      Storing the Audit Trail ...................................................... 560
      Examining the Audit Trail ................................................. 560
   Auditing on a System With Zones ....................................... 560

29 Planning for Solaris Auditing .................................................. 563
   Planning Solaris Auditing (Task Map) ................................. 563
   Planning Solaris Auditing (Tasks) ........................................ 564
Managing Solaris Auditing (Tasks) ........................................................................................................575
Solaris Auditing (Task Map) ................................................................................................................575
Configuring Audit Files (Task Map) ..................................................................................................576
Configuring Audit Files (Tasks) .........................................................................................................576
▼ How to Modify the audit_control File .........................................................................................577
▼ How to Configure syslog Audit Logs ..........................................................................................579
▼ How to Change a User’s Audit Characteristics ...........................................................................581
▼ How to Add an Audit Class ............................................................................................................583
▼ How to Change an Audit Event’s Class Membership ................................................................584
Configuring and Enabling the Audit Service (Task Map) .................................................................586
Configuring and Enabling the Audit Service (Tasks) .......................................................................587
▼ How to Create Partitions for Audit Files .......................................................................................587
▼ How to Configure the audit_warn Email Alias ...........................................................................589
▼ How to Configure Audit Policy .....................................................................................................590
▼ How to Configure Audit Queue Parameters .................................................................................592
▼ How to Enable the Audit Service ..................................................................................................594
▼ How to Disable the Audit Service ................................................................................................595
▼ How to Update the Audit Service ................................................................................................596
Configuring the Audit Service in Zones (Tasks) .................................................................................598
▼ How to Configure All Zones Identically for Auditing ..................................................................598
▼ How to Configure Per-Zone Auditing ..........................................................................................600
Managing Audit Records (Task Map) ...............................................................................................601
Managing Audit Records ..................................................................................................................602
▼ How to Display Audit Record Formats ........................................................................................602
Contents

31 Solaris Auditing (Reference) ................................................................. 625

Solaris Audit Service ........................................................................... 625

Audit Commands .................................................................................. 626

auditd Daemon .................................................................................... 626
audit Command .................................................................................... 627
auditrecord Command .......................................................................... 627
auditreduce Command ......................................................................... 628
praudit Command ................................................................................ 629
auditconfig Command ......................................................................... 631

Files Used in the Audit Service ............................................................. 631

system File .......................................................................................... 632
syslog.conf File .................................................................................... 632
audit_class File ................................................................................... 632
audit_control File ................................................................................ 632
audit_event File ................................................................................... 634
audit_user Database ............................................................................ 634
audit_warn Script ................................................................................ 635
bsmconv Script ................................................................................... 636
Preface

System Administration Guide: Security Services is part of a multivolume set that covers a significant part of the Solaris™ Operating System (Solaris OS) administration information. This book assumes that you have already installed a Solaris Express Community Edition release, and you have set up any networking software that you plan to use. The Solaris OS is part of the Solaris product family, which includes many features, such as the Java Desktop System.

Note – This Solaris release supports systems that use the SPARC® and x86 families of processor architectures. The supported systems appear in the Solaris OS: Hardware Compatibility Lists (http://www.sun.com/bigadmin/hcl). This document cites any implementation differences between the platform types.

In this document these x86 related terms mean the following:
- “x86” refers to the larger family of 64-bit and 32-bit x86 compatible products.
- “x64” relates specifically to 64-bit x86 compatible CPUs.
- “32-bit x86” points out specific 32-bit information about x86 based systems.

For supported systems, see the Solaris OS: Hardware Compatibility Lists.

Who Should Use This Book

This book is intended for anyone who is responsible for administering one or more systems that run a Solaris Express Community Edition release. To use this book, you should have more than two years of UNIX® system administration experience. Attending training courses in UNIX system administration might be helpful.
How the System Administration Volumes Are Organized

Here is a list of the topics that are covered by the volumes of the System Administration Guides.

<table>
<thead>
<tr>
<th>Book Title</th>
<th>Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System Administration Guide: Basic Administration</strong></td>
<td>User accounts and groups, server and client support, shutting down and booting a system, managing services, and managing software (packages and patches)</td>
</tr>
<tr>
<td><strong>System Administration Guide: Advanced Administration</strong></td>
<td>Terminals and modems, system resources (disk quotas, accounting, and crontabs), system processes, and troubleshooting Solaris software problems</td>
</tr>
<tr>
<td><strong>System Administration Guide: Devices and File Systems</strong></td>
<td>Removable media, disks and devices, file systems, and backing up and restoring data</td>
</tr>
<tr>
<td><strong>System Administration Guide: IP Services</strong></td>
<td>TCP/IP network administration, IPv4 and IPv6 address administration, DHCP, IPsec, IKE, Solaris IP filter, Mobile IP, IP network multipathing (IPMP), and IPQoS</td>
</tr>
<tr>
<td><strong>System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)</strong></td>
<td>DNS, NIS, and LDAP naming and directory services, including transitioning from NIS to LDAP and transitioning from NIS+ to LDAP</td>
</tr>
<tr>
<td><strong>System Administration Guide: Naming and Directory Services (NIS+)</strong></td>
<td>NIS+ naming and directory services</td>
</tr>
<tr>
<td><strong>System Administration Guide: Network Interfaces and Network Virtualization</strong></td>
<td>Networking stack, NIC driver property configuration, network interface configuration, administration of VLANs and link aggregations, configuring WiFi wireless networking.</td>
</tr>
<tr>
<td><strong>System Administration Guide: Network Services</strong></td>
<td>Web cache servers, time-related services, network file systems (NFS and Autosfs), mail, SLP, and PPP</td>
</tr>
<tr>
<td><strong>System Administration Guide: Security Services</strong></td>
<td>Auditing, device management, file security, BART, Kerberos services, PAM, Solaris Cryptographic Framework, privileges, RBAC, SASL, and Solaris Secure Shell</td>
</tr>
<tr>
<td><strong>System Administration Guide: Virtualization Using the Solaris Operating System</strong></td>
<td>Resource management features, which enable you to control how applications use available system resources; zones software partitioning technology, which virtualizes operating system services to create an isolated environment for running applications; and virtualization using Sun\textsuperscript{TM} xVM hypervisor technology, which supports multiple operating system instances simultaneously</td>
</tr>
<tr>
<td><strong>Solaris CIFS Administration Guide</strong></td>
<td>Solaris CIFS service, which enables you to configure a Solaris system to make CIFS shares available to CIFS clients; and native identity mapping services, which enables you to map user and group identities between Solaris systems and Windows systems</td>
</tr>
</tbody>
</table>
**Related Third-Party Web Site References**

Third party URLs are referenced in this document and provide additional, related information.

Sun is not responsible for the availability of third-party web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused by or in connection with the use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

**Documentation, Support, and Training**

The Sun web site provides information about the following additional resources:


**Typographic Conventions**

The following table describes the typographic conventions that are used in this book.

<table>
<thead>
<tr>
<th>Typeface</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>AaBbCc123</td>
<td>The names of commands, files, and directories, and onscreen computer output</td>
<td>Edit your .login file. Use ls -a to list all files. machine_name% you have mail.</td>
</tr>
</tbody>
</table>
Shell Prompts in Command Examples

The following table shows the default UNIX system prompt and superuser prompt for shells that are included in the Solaris OS. Note that the default system prompt that is displayed in command examples varies, depending on the Solaris release.

<table>
<thead>
<tr>
<th>TABLE P–2</th>
<th>Shell Prompts</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Shell</strong></td>
<td><strong>Prompt</strong></td>
</tr>
<tr>
<td>Bash shell, Korn shell, and Bourne shell</td>
<td>$</td>
</tr>
<tr>
<td>Bash shell, Korn shell, and Bourne shell for superuser</td>
<td>#</td>
</tr>
<tr>
<td>C shell</td>
<td>machine_name%</td>
</tr>
<tr>
<td>C shell for superuser</td>
<td>machine_name#</td>
</tr>
</tbody>
</table>
PART I

Security Overview

This book focuses on the features that enhance security in the Solaris Operating System. This book is intended for system administrators and users of these security features. The overview chapter introduces the topics in the book.
To maintain the security of the Solaris Operating System (Solaris OS), Solaris software provides the following features:

- **“System Security” on page 32** – The ability to prevent intrusion, to protect machine resources and devices from misuse, and to protect files from malicious modification or unintentional modification by users or intruders.
  
  For a general discussion of system security, see Chapter 2, “Managing Machine Security (Overview).”

- **“Solaris Cryptographic Services” on page 33** – The ability to scramble data so that only the sender and the designated receiver can read the contents, and to manage cryptographic providers and public key objects.

- **“Authentication Services” on page 34** – The ability to securely identify a user, which requires the user’s name and some form of proof, typically a password.

- **“Authentication With Encryption” on page 34** – The ability to ensure that authenticated parties can communicate without interception, modification, or spoofing.

- **“Solaris Auditing” on page 35** – The ability to identify the source of security changes to the system, including file access, security-related system calls, and authentication failures.

- **“Security Policy” on page 35** – The design and implementation of security guidelines for a computer or network of computers.
System Security

System security ensures that the system’s resources are used properly. Access controls can restrict who is permitted access to resources on the system. The Solaris OS features for system security and access control include the following:

- **Login administration tools** – Commands for monitoring and controlling a user’s ability to log in. See “Securing Logins and Passwords (Task Map)” on page 60.

- **Hardware access** – Commands for limiting access to the PROM, and for restricting who can boot the system. See “SPARC: Controlling Access to System Hardware (Task Map)” on page 74.

- **Resource access** – Tools and strategies for maximizing the appropriate use of machine resources while minimizing the misuse of those resources. See “Controlling Access to Machine Resources” on page 47.

- **Role-based access control (RBAC)** – An architecture for creating special, restricted user accounts that are permitted to perform specific administrative tasks. See “Role-Based Access Control (Overview)” on page 155.

- **Privileges** – Discrete rights on processes to perform operations. These process rights are enforced in the kernel. See “Privileges (Overview)” on page 164.

- **Device management** – Device policy additionally protects devices that are already protected by UNIX permissions. Device allocation controls access to peripheral devices, such as a microphone or CD-ROM drive. Upon deallocation, device-clean scripts can then erase any data from the device. See “Controlling Access to Devices” on page 45.

- **Basic Audit Reporting Tool (BART)** – A snapshot, called a manifest, of the file attributes of files on a system. By comparing the manifests across systems or on one system over time, changes to files can be monitored to reduce security risks. See Chapter 6, “Using the Basic Audit Reporting Tool (Tasks).”

- **File permissions** – Attributes of a file or directory. Permissions restrict the users and groups that are permitted to read, write, or execute a file, or search a directory. See Chapter 7, “Controlling Access to Files (Tasks).”

- **Antivirus software** – A vscan service checks files for viruses before an application uses the files. A file system can invoke this service to scan files in real time for the most recent virus definitions before the files are accessed by any clients of the file system. The real-time scan is performed by third-party applications. A file can be scanned when it is opened and after it is closed. See Chapter 4, “Virus Scanning Service (Tasks).”
Solaris Cryptographic Services

Cryptography is the science of encrypting and decrypting data. Cryptography is used to insure integrity, privacy, and authenticity. Integrity means that the data has not been altered. Privacy means that the data is not readable by others. Authenticity for data means that what was delivered is what was sent. User authentication means that the user has supplied one or more proofs of identity. Authentication mechanisms mathematically verify the source of the data or the proof of identity. Encryption mechanisms scramble data so that the data is not readable by a casual observer. Cryptographic services provide authentication and encryption mechanisms to applications and users.

Cryptographic algorithms use hashing, chaining, and other mathematical techniques to create ciphers that are difficult to break. Authentication mechanisms require that the sender and the receiver compute an identical number from the data. Encryption mechanisms rely on the sender and the receiver sharing information about the method of encryption. This information enables only the receiver and the sender to decrypt the message. The Solaris OS provides a centralized cryptographic framework, and provides encryption mechanisms that are tied to particular applications.

- **Solaris™ Cryptographic Framework** – A central framework of cryptographic services for kernel-level and user-level consumers. Uses include passwords, IPsec, and third-party applications. The cryptographic framework includes a number of software encryption modules. The framework enables you to specify which software encryption modules or hardware encryption sources an application can use. The framework is built on the PKCS #11 v2 library. This library is implemented according to the following standard: RSA Security Inc. PKCS #11 Cryptographic Token Interface (Cryptoki). The library provides an API for third-party developers to plug in the cryptographic requirements for their applications. See Chapter 13, “Solaris Cryptographic Framework (Overview).”

- **Encryption mechanisms per application** –
  - For the use of DES in Secure RPC, see “Overview of Secure RPC” on page 291.
  - For the use of DES, 3DES, AES, and ARCFOUR in the Kerberos service, see Chapter 21, “Introduction to the Kerberos Service.”
  - For the use of RSA, DSA, and ciphers such as Blowfish in Solaris Secure Shell, see Chapter 19, “Using Solaris Secure Shell (Tasks).”
  - For the use of cryptographic algorithms in passwords, see “Changing the Password Algorithm (Task Map)” on page 67.

In the Solaris Express Community Edition, the Key Management Framework (KMF) provides a central utility for managing public key objects, including policy, keys, and certificates. KMF manages these objects for OpenSSL, NSS, and PKCS #11 public key technologies. See Chapter 15, “Solaris Key Management Framework.”
Authentication Services

Authentication is a mechanism that identifies a user or service based on predefined criteria. Authentication services range from simple name-password pairs to more elaborate challenge-response systems, such as smart cards and biometrics. Strong authentication mechanisms rely on a user supplying information that only that person knows, and a personal item that can be verified. A user name is an example of information that the person knows. A smart card or a fingerprint, for example, can be verified. The Solaris features for authentication include the following:

- **Secure RPC** – An authentication mechanism that uses the Diffie-Hellman protocol to protect NFS mounts and a name service, such as NIS or NIS+. See “Overview of Secure RPC” on page 291.

- **Pluggable Authentication Module (PAM)** – A framework that enables various authentication technologies to be plugged into a system entry service without recompiling the service. Some of the system entry services include `login` and `ftp`. See Chapter 17, “Using PAM.”

- **Simple Authentication and Security Layer (SASL)** – A framework that provides authentication and security services to network protocols. See Chapter 18, “Using SASL.”

- **Solaris Secure Shell** – A secure remote login and transfer protocol that encrypts communications over an insecure network. See Chapter 19, “Using Solaris Secure Shell (Tasks).”

- **Kerberos service** – A client-server architecture that provides encryption with authentication. See Chapter 21, “Introduction to the Kerberos Service.”

Authentication With Encryption

Authentication with encryption is the basis of secure communication. Authentication helps ensure that the source and the destination are the intended parties. Encryption codes the communication at the source, and decodes the communication at the destination. Encryption prevents intruders from reading any transmissions that the intruders might manage to intercept. The Solaris features for secure communication include the following:

- **Solaris Secure Shell** – A protocol for protecting data transfers and interactive user network sessions from eavesdropping, session hijacking, and “man-in-the-middle” attacks. Strong authentication is provided through public key cryptography. X windows services and other network services can be tunneled safely over Secure Shell connections for additional protection. See Chapter 19, “Using Solaris Secure Shell (Tasks).”

- **Kerberos service** – A client-server architecture that provides authentication with encryption. See Chapter 21, “Introduction to the Kerberos Service.”

Solaris Auditing

Auditing is a fundamental concept of system security and maintainability. Auditing is the process of examining the history of actions and events on a system to determine what happened. The history is kept in a log of what was done, when it was done, by whom, and what was affected. See Chapter 28, “Solaris Auditing (Overview).”

Security Policy

The phrase security policy, or policy, is used throughout this book to refer to an organization’s security guidelines. Your site’s security policy is the set of rules that define the sensitivity of the information that is being processed and the measures that are used to protect the information from unauthorized access. Security technologies such as Solaris Secure Shell, authentication, RBAC, authorization, privileges, and resource control provide measures to protect information.

Some security technologies also use the word policy when describing specific aspects of their implementation. For example, Solaris auditing uses audit policy options to configure some aspects of auditing policy. The following table points to glossary, man page, and information on features that use the word policy to describe specific aspects of their implementation.

<table>
<thead>
<tr>
<th>Glossary Definition</th>
<th>Selected Man Pages</th>
<th>Further Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>audit policy</td>
<td>audit_control(4), audit_user(4), auditconfig(1M)</td>
<td>Chapter 28, “Solaris Auditing (Overview)”</td>
</tr>
<tr>
<td>policy in the cryptographic framework</td>
<td>cryptoadm(1M)</td>
<td>Chapter 13, “Solaris Cryptographic Framework (Overview)”</td>
</tr>
<tr>
<td>device policy</td>
<td>getdevpolicy(1M)</td>
<td>“Controlling Access to Devices” on page 45</td>
</tr>
<tr>
<td>Kerberos policy</td>
<td>krb5.conf(4)</td>
<td>Chapter 25, “Administering Kerberos Principals and Policies (Tasks)”</td>
</tr>
<tr>
<td>Glossary Definition</td>
<td>Selected Man Pages</td>
<td>Further Information</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-------------------------------------------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>password policy</td>
<td>passwd(1), nsswitch.conf(4), crypt.conf(4), policy.conf(4)</td>
<td>“Maintaining Login Control” on page 40</td>
</tr>
<tr>
<td>policy for public key technologies</td>
<td>kmfcfg(1)</td>
<td>Chapter 15, “Solaris Key Management Framework”</td>
</tr>
<tr>
<td>RBAC policy</td>
<td>rbac(5)</td>
<td>“exec_attr Database” on page 216</td>
</tr>
</tbody>
</table>
System, File, and Device Security

This section covers security that can be configured on a non-networked system. The chapters discuss planning, monitoring, and controlling access to the disk, to files, and to peripheral devices. This section also includes a chapter about scanning files for viruses.
Keeping a machine’s information secure is an important system administration responsibility. This chapter provides overview information about managing machine security.

The following is a list of the overview information in this chapter:

- “Controlling Access to a Computer System” on page 39
- “Controlling Access to Devices” on page 45
- “Controlling Access to Machine Resources” on page 47
- “Controlling Access to Files” on page 52
- “Controlling Network Access” on page 53
- “Reporting Security Problems” on page 57

**Controlling Access to a Computer System**

In the workplace, all machines that are connected to a server can be thought of as one large multifaceted system. You are responsible for the security of this larger system. You need to defend the network from outsiders who are trying to gain access to the network. You also need to ensure the integrity of the data on the machines within the network.

At the file level, the Solaris OS provides standard security features that you can use to protect files, directories, and devices. At the system and network levels, the security issues are mostly the same. The first line of security defense is to control access to your system.

You can control and monitor system access by doing the following:

- “Maintaining Physical Security” on page 40
- “Maintaining Login Control” on page 40
- “Controlling Access to Devices” on page 45
- “Controlling Access to Machine Resources” on page 47
- “Controlling Access to Files” on page 52
- “Controlling Network Access” on page 53
Maintaining Physical Security

To control access to your system, you must maintain the physical security of your computing environment. For instance, a system that is logged in and left unattended is vulnerable to unauthorized access. An intruder can gain access to the operating system and to the network. The computer’s surroundings and the computer hardware should be physically protected from unauthorized access.

You can protect a SPARC system from unauthorized access to the hardware settings. Use the `eeprom` command to require a password to access the PROM. For more information, see “How to Require a Password for Hardware Access” on page 74.

Maintaining Login Control

You also must prevent unauthorized logins to a system or the network, which you can do through password assignment and login control. All accounts on a system should have a password. A password is a simple authentication mechanism. An account without a password makes your entire network accessible to an intruder who guesses a user name. A strong password algorithm protects against brute force attacks.

When a user logs in to a system, the `login` command checks the appropriate name service or directory service database according to the information that is listed in the `/etc/nsswitch.conf` file. This file can include the following entries:

- `files` – Designates the `/etc` files on the local system
- `ldap` – Designates the LDAP directory service on the LDAP server
- `nis` – Designates the NIS database on the NIS master server
- `nisplus` – Designates the NIS+ database on the NIS+ root server

For a description of the `nsswitch.conf` file, see the `nsswitch.conf(4)` man page. For information about naming services and directory services, see the System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP) or the System Administration Guide: Naming and Directory Services (NIS+).

The `login` command verifies the user name and password that were supplied by the user. If the user name is not in the password file, the `login` command denies access to the system. If the password is not correct for the user name that was specified, the `login` command denies access to the system. When the user supplies a valid user name and its corresponding password, the system grants the user access to the system.

PAM modules can streamline login to applications after a successful system login. For more information, see Chapter 17, “Using PAM.”
Sophisticated authentication and authorization mechanisms are available on Solaris systems. For a discussion of authentication and authorization mechanisms at the network level, see “Authentication and Authorization for Remote Access” on page 54.

## Managing Password Information

When users log in to a system, they must supply both a username and a password. Although logins are publicly known, passwords must be kept secret. Passwords should be known only to each user. You should ask your users to choose their passwords carefully. Users should change their passwords often.

Passwords are initially created when you set up a user account. To maintain security on user accounts, you can set up password aging to force users to routinely change their passwords. You can also disable a user account by locking the password. For detailed information about administering passwords, see Chapter 4, “Managing User Accounts and Groups (Overview),” in System Administration Guide: Basic Administration and the passwd(1) man page.

### Local Passwords

If your network uses local files to authenticate users, the password information is kept in the system’s /etc/passwd and /etc/shadow files. The user name and other information are kept in the password file /etc/passwd. The encrypted password itself is kept in a separate shadow file, /etc/shadow. This security measure prevents a user from gaining access to the encrypted passwords. While the /etc/passwd file is available to anyone who can log in to a system, only superuser or an equivalent role can read the /etc/shadow file. You can use the passwd command to change a user's password on a local system.

### NIS and NIS+ Passwords

If your network uses NIS to authenticate users, password information is kept in the NIS password map. NIS does not support password aging. You can use the command passwd -r nis to change a user’s password that is stored in an NIS password map.

If your network uses NIS+ to authenticate users, password information is kept in the NIS+ database. Information in the NIS+ database can be protected by restricting access to authorized users only. You can use the passwd -r nispplus command to change a user's password that is stored in an NIS+ database.

### LDAP Passwords

The Solaris LDAP naming service stores password information and shadow information in the ou=people container of the LDAP directory tree. On the Solaris LDAP naming service client, you can use the passwd -r ldap command to change a user's password. The LDAP naming service stores the password in the LDAP repository.
In the Solaris 10 release, password policy is enforced on the Sun Java™ System Directory Server. Specifically, the client's pam_ldap module follows the password policy controls that are enforced on the Sun Java System Directory Server. For more information, see “LDAP Naming Services Security Model” in System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP).

**Password Encryption**

Strong password encryption provides an early barrier against attack. Solaris software provides four password encryption algorithms. The two MD5 algorithms and the Blowfish algorithm provide more robust password encryption than the UNIX algorithm.

**Password Algorithm Identifiers**

You specify the algorithms configuration for your site in the `/etc/security/policy.conf` file. In the `policy.conf` file, the algorithms are named by their identifier, as shown in the following table.

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Description</th>
<th>Algorithm Man Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The MD5 algorithm that is compatible with MD5 algorithms on BSD and Linux systems.</td>
<td>crypt_bsdmd5(5)</td>
</tr>
<tr>
<td>2a</td>
<td>The Blowfish algorithm that is compatible with the Blowfish algorithm on BSD systems.</td>
<td>crypt_bsdbf(5)</td>
</tr>
<tr>
<td>md5</td>
<td>The Sun MD5 algorithm, which is considered stronger than the BSD and Linux version of MD5.</td>
<td>crypt_sunmd5(5)</td>
</tr>
<tr>
<td>5</td>
<td>The SHA256 algorithm. SHA stands for Secure Hash Algorithm. This algorithm is a member of the SHA-2 family. SHA256 supports 255-character passwords.</td>
<td>crypt_sha256(5)</td>
</tr>
<tr>
<td>6</td>
<td>The SHA512 algorithm.</td>
<td>crypt_sha512(5)</td>
</tr>
<tr>
<td>&quot;unix&quot;</td>
<td>The traditional UNIX encryption algorithm. This algorithm is the default module in the policy.conf file.</td>
<td>crypt_unix(5)</td>
</tr>
</tbody>
</table>

**Algorithms Configuration in the policy.conf File**

The following shows the default algorithms configuration in the `policy.conf` file:

```
#...
# crypt(3c) Algorithms Configuration
#
# CRYPT_ALGORITHMS_ALLOW specifies the algorithms that are allowed to
```
When you change the value for CRYPT_DEFAULT, the passwords of new users are encrypted with the algorithm that is associated with the new value. When current users change their passwords, how their old password was encrypted affects which algorithm is used to encrypt the new password.

For example, assume that CRYPT_ALGORITHMS_ALLOW=1, 2a, md5, 5, 6 and CRYPT_DEFAULT=1. The following table shows which algorithm would be used to generate the encrypted password.

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Password Algorithm</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 = crypt_bsdmd5</td>
<td>Uses same algorithm</td>
<td>The 1 identifier is also the value of CRYPT_DEFAULT. The user's password continues to be encrypted with the crypt_bsdmd5 algorithm.</td>
</tr>
<tr>
<td>2a = crypt_bsdbf</td>
<td>Uses same algorithm</td>
<td>The 2a identifier is in the CRYPT_ALGORITHMS_ALLOW list. Therefore, the new password is encrypted with the crypt_bsdbf algorithm.</td>
</tr>
<tr>
<td>md5 = crypt_md5</td>
<td>Uses same algorithm</td>
<td>The md5 identifier is in the CRYPT_ALGORITHMS_ALLOW list. Therefore, the new password is encrypted with the crypt_md5 algorithm.</td>
</tr>
<tr>
<td>5 = crypt_sha256</td>
<td>Uses same algorithm</td>
<td>The 5 identifier is in the CRYPT_ALGORITHMS_ALLOW list. Therefore, the new password is encrypted with the crypt_sha256 algorithm.</td>
</tr>
<tr>
<td>6 = crypt_sha512</td>
<td>Uses same algorithm</td>
<td>The 6 identifier is in the CRYPT_ALGORITHMS_ALLOW list. Therefore, the new password is encrypted with the crypt_sha512 algorithm.</td>
</tr>
<tr>
<td><strong>unix</strong> = crypt_unix</td>
<td>Uses crypt_bsdmd5 algorithm</td>
<td>The <strong>unix</strong> identifier is not in the CRYPT_ALGORITHMS_ALLOW list. Therefore, the crypt_unix algorithm cannot be used. The new password is encrypted with the CRYPT_DEFAULT algorithm.</td>
</tr>
</tbody>
</table>
For more information on configuring the algorithm choices, see the `policy.conf(4)` man page. To specify password encryption algorithms, see "Changing the Password Algorithm (Task Map)" on page 67.

**Special System Logins**

Two common ways to access a system are by using a conventional user login, or by using the `root` login. In addition, a number of special system logins enable a user to run administrative commands without using the `root` account. As system administrator, you assign passwords to these login accounts.

The following table lists some system login accounts and their uses. The system logins perform special functions. Each login has its own group identification number (GID). Each login should have its own password, which should be divulged on a need-to-know basis.

<table>
<thead>
<tr>
<th>Login Account</th>
<th>GID</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td>0</td>
<td>Has almost no restrictions. Overrides all other logins, protections, and permissions. The <code>root</code> account has access to the entire system. The password for the <code>root</code> login should be very carefully protected. The <code>root</code> account, superuser, owns most of the Solaris commands.</td>
</tr>
<tr>
<td>daemon</td>
<td>1</td>
<td>Controls background processing.</td>
</tr>
<tr>
<td>bin</td>
<td>2</td>
<td>Owns some Solaris commands.</td>
</tr>
<tr>
<td>sys</td>
<td>3</td>
<td>Owns many system files.</td>
</tr>
<tr>
<td>adm</td>
<td>4</td>
<td>Owns certain administrative files.</td>
</tr>
<tr>
<td>lp</td>
<td>71</td>
<td>Owns the object data files and spooled data files for the printer.</td>
</tr>
<tr>
<td>uucp</td>
<td>5</td>
<td>Owns the object data files and spooled data files for UUCP, the UNIX-to-UNIX copy program.</td>
</tr>
<tr>
<td>nuucp</td>
<td>9</td>
<td>Is used by remote systems to log in to the system and start file transfers.</td>
</tr>
</tbody>
</table>

**Remote Logins**

Remote logins offer a tempting avenue for intruders. The Solaris OS provides several commands to monitor, limit, and disable remote logins. For procedures, see "Securing Logins and Passwords (Task Map)" on page 60.

By default, remote logins cannot gain control or read certain system devices, such as the system mouse, keyboard, frame buffer, or audio device. For more information, see the `logindevperm(4)` man page.
Dial-Up Logins

When a computer can be accessed through a modem or a dial-up port, you can add an extra layer of security. You can require a dial-up password for users who access a system through a modem or dial-up port. The dial-up password is an additional password that a user must supply before being granted access to the system.

Only superuser or a role of equivalent capabilities can create or change a dial-up password. To ensure the integrity of the system, the password should be changed about once a month. The most effective use of this feature is to require a dial-up password to gain access to a gateway system. To set up dial-up passwords, see “How to Create a Dial-Up Password” on page 65.

Two files are involved in creating a dial-up password, /etc/dialups and /etc/d_passwd. The dialups file contains a list of ports that require a dial-up password. The d_passwd file contains a list of shell programs that require an encrypted password as the additional dial-up password. The information in these two files is processed as follows:

- If the user's login shell in /etc/passwd matches an entry in /etc/d_passwd, the user must supply a dial-up password.
- If the user's login shell in /etc/passwd is not found in /etc/d_passwd, the user must supply the default password. The default password is the entry for /usr/bin/sh.
- If the login shell field in /etc/passwd is empty, the user must supply the default password. The default password is the entry for /usr/bin/sh.
- If /etc/d_passwd has no entry for /usr/bin/sh, then those users whose login shell field in /etc/passwd is empty or does not match any entry in /etc/d_passwd are not prompted for a dial-up password.
- Dial-up logins are disabled if /etc/d_passwd has the /usr/bin/sh:*: entry only.

Controlling Access to Devices

Peripheral devices that are attached to a computer system pose a security risk. Microphones can pick up conversations and transmit them to remote systems. CD-ROMs can leave their information behind for reading by the next user of the CD-ROM device. Printers can be accessed remotely. Devices that are integral to the system can also present security issues. For example, network interfaces such as hme0 are considered integral devices.

Solaris software provides two methods of controlling access to devices. Device policy restricts or prevents access to devices that are integral to the system. Device policy is enforced in the kernel. Device allocation restricts or prevents access to peripheral devices. Device allocation is enforced at user allocation time.

Device policy uses privileges to protect selected devices in the kernel. For example, the device policy on network interfaces such as hme requires all privileges for reading or writing.
Device allocation uses authorizations to protect peripheral devices, such as printers or microphones. By default, device allocation is not enabled. Once enabled, device allocation can be configured to prevent the use of a device or to require authorization for access to the device. When a device is allocated for use, no other user can access the device until the current user deallocates it.

A Solaris system can be configured in several areas to control access to devices:

- **Set device policy** – In the Solaris 10 release, you can require that the process that is accessing a particular device be running with a set of privileges. Processes without those privileges cannot use the device. At boot time, Solaris software configures device policy. Third-party drivers can be configured with device policy during installation. After installation, you, as the system administrator can add device policy to a device.

- **Make devices allocatable** – When you enable device allocation, you can restrict the use of a device to one user at a time. You can further require that the user fulfill some security requirements. For example, you can require that the user be authorized to use the device.

- **Prevent devices from being used** – You can prevent the use of a device, such as a microphone, by any user on a computer system. A computer kiosk might be a good candidate for making certain devices unavailable for use.

- **Confine a device to a particular zone** – You can assign the use of a device to a non-global zone. For more information, see “Device Use in Non-Global Zones” in System Administration Guide: Virtualization Using the Solaris Operating System. For a more general discussion of devices and zones, see “Configured Devices in Zones” in System Administration Guide: Virtualization Using the Solaris Operating System.

### Device Policy (Overview)

The device policy mechanism enables you to specify that processes that open a device require certain privileges. Devices that are protected by device policy can only be accessed by processes that are running with the privileges that the device policy specifies. The Solaris OS provides default device policy. For example, network interfaces such as hme0 require that the processes that access the interface be running with the net_rawaccess privilege. The requirement is enforced in the kernel. For more information about privileges, see “Privileges (Overview)” on page 164.

In earlier Solaris OS releases, device nodes were protected by file permissions alone. For example, devices owned by group sys could be opened only by members of group sys. In the Solaris 10 release, file permissions do not predict who can open a device. Instead, devices are protected with file permissions and with device policy. For example, the /dev/1p file has 666 permissions. However, the device can only be opened by a process with the appropriate privileges.

The configuration of device policy can be audited. The AUE_MODEVPLCY audit event records changes in device policy.
For more information about device policy, see the following:

- “Configuring Device Policy (Task Map)” on page 84
- “Device Policy Commands” on page 97
- “Privileges and Devices” on page 171

**Device Allocation (Overview)**

The device allocation mechanism enables you to restrict access to a peripheral device, such as a CD-ROM. You manage the mechanism locally. If device allocation is not enabled, peripheral devices are protected only by file permissions. For example, by default, peripheral devices are available for the following uses:

- Any user can read and write to a diskette or CD-ROM.
- Any user can attach a microphone.
- Any user can access an attached printer.

Device allocation can restrict a device to authorized users. Device allocation can also prevent a device from being accessed at all. A user who allocates a device has exclusive use of that device until the user deallocates the device. When a device is deallocated, device-clean scripts erase any leftover data. You can write a device-clean script to purge information from devices that do not have a script. For an example, see “Writing New Device-Clean Scripts” on page 103.

Attempts to allocate a device, deallocate a device, and list allocatable devices can be audited. The audit events are part of the ot audit class.

For more information on device allocation, see the following:

- “Managing Device Allocation (Task Map)” on page 87
- “Device Allocation” on page 98
- “Device Allocation Commands” on page 99

**Controlling Access to Machine Resources**

As system administrator, you can control and monitor system activity. You can set limits on who can use what resources. You can log resource use, and you can monitor who is using the resources. You can also set up your machines to minimize improper use of resources.

**Limiting and Monitoring Superuser**

Your system requires a root password for superuser access. In the default configuration, a user cannot remotely log in to a system as root. When logging in remotely, a user must log in with the user’s user name and then use the su command to become root. You can monitor who has
been using the `su` command, especially those users who are trying to gain superuser access. For procedures that monitor superuser and limit access to superuser, see “Monitoring and Restricting Superuser (Task Map)” on page 71.

**Configuring Role-Based Access Control to Replace Superuser**

Role-based access control, or RBAC, is designed to limit the capabilities of superuser. Superuser, the root user, has access to every resource in the system. With RBAC, you can replace root with a set of roles with discrete powers. For example, you can set up one role to handle user account creation, and another role to handle system file modification. When you have established a role to handle a function or set of functions, you can remove those functions from root’s capabilities.

Each role requires that a known user log in with their user name and password. After logging in, the user then assumes the role with a specific role password. As a consequence, someone who learns the root password has limited ability to damage your system. For more on RBAC, see “Role-Based Access Control (Overview)” on page 155.

**Preventing Unintentional Misuse of Machine Resources**

You can prevent you and your users from making unintentional errors in the following ways:

- You can keep from running a Trojan horse by correctly setting the `PATH` variable.
- You can assign a restricted shell to users. A restricted shell prevents user error by steering users to those parts of the system that the users need for their jobs. In fact, through careful setup, you can ensure that users access only those parts of the system that help the users work efficiently.
- You can set restrictive permissions on files that users do not need to access.

**Setting the PATH Variable**

You should take care to correctly set the `PATH` variable. Otherwise, you can accidentally run a program that was introduced by someone else. The intruding program can corrupt your data or harm your system. This kind of program, which creates a security hazard, is referred to as a *Trojan horse*. For example, a substitute `su` program could be placed in a public directory where you, as system administrator, might run the substitute program. Such a script would look just like the regular `su` command. Because the script removes itself after execution, you would have little evidence to show that you have actually run a Trojan horse.
The PATH variable is automatically set at login time. The path is set through the startup files: .login, .profile, and .cshrc. When you set up the user search path so that the current directory (.) comes last, you are protected from running this type of Trojan horse. The PATH variable for superuser should not include the current directory at all.

**Assigning a Restricted Shell to Users**

The standard shell allows a user to open files, execute commands, and so on. The restricted shell limits the ability of a user to change directories and to execute commands. The restricted shell is invoked with the /usr/lib/rsh command. Note that the restricted shell is not the remote shell, which is /usr/sbin/rsh.

The restricted shell differs from the standard shell in the following ways:

- The user is limited to the user's home directory, so the user cannot use the cd command to change directories. Therefore, the user cannot browse system files.
- The user cannot change the PATH variable, so the user can use only commands in the path that is set by the system administrator. The user also cannot execute commands or scripts by using a complete path name.
- The user cannot redirect output with > or >>.

The restricted shell enables you to limit a user's ability to stray into system files. The shell creates a limited environment for a user who needs to perform specific tasks. The restricted shell is not completely secure, however, and is only intended to keep unskilled users from inadvertently doing damage.

For information about the restricted shell, use the man -s1m rsh command to see the rsh(1M) man page.

**Restricting Access to Data in Files**

Because the Solaris OS is a multiuser environment, file system security is the most basic security risk on a system. You can use traditional UNIX file protections to protect your files. You can also use the more secure access control lists (ACLs).

You might want to allow some users to read some files, and give other users permission to change or delete some files. You might have some data that you do not want anyone else to see. Chapter 7, “Controlling Access to Files (Tasks),” discusses how to set file permissions.

**Restricting setuid Executable Files**

Executable files can be security risks. Many executable programs have to be run as root, that is, as superuser, to work properly. These setuid programs run with the user ID set to 0. Anyone who is running these programs runs the programs with the root ID. A program that runs with the root ID creates a potential security problem if the program was not written with security in mind.
Except for the executables that Sun ships with the setuid bit set to root, you should disallow the use of setuid programs. If you cannot disallow the use of setuid programs, then you should at least restrict their use. Secure administration requires few setuid programs.

For more information, see "Preventing Executable Files From Compromising Security" on page 136. For procedures, see "Protecting Against Programs With Security Risk (Task Map)" on page 149.

**Using the Solaris Security Toolkit**

The Solaris Security Toolkit provides a flexible and extensible mechanism to minimize, harden, and secure a Solaris system. The Solaris Security Toolkit, informally known as the JASS toolkit, is a tool that enables the user to perform security modifications to a system. The tool can provide a report on the security status of a system. The tool also has the ability to undo previous runs of the tool. The JASS toolkit can be downloaded from the Sun web site, [http://wwws.sun.com/security/jass](http://wwws.sun.com/security/jass). The web site contains pointers to online documentation.


**Using the netservices limited Configuration**

By default, when the Solaris 10 release is installed, a large set of network services are enabled. To limit a system’s network connectivity, you run the `netservices limited` command. This command activates a “Secure by Default” (SBD) configuration. With SBD, the only network service that accepts network requests is the sshd daemon. All other network services are disabled or handle local requests only. To enable individual network services, such as ftp, you use the Solaris Service Management Facility (SMF). For more information, see the `netservices(1M)` and `smf(5)` man pages.

**Using Solaris Resource Management Features**

Solaris software provides sophisticated resource management features. Using these features, you can allocate, schedule, monitor, and cap resource use by applications in a server consolidation environment. The resource controls framework enables you to set constraints on system resources that are consumed by processes. Such constraints help to prevent denial-of-service attacks by a script that attempts to flood a system’s resources.

With Solaris resource management features, you can designate resources for particular projects. You can also dynamically adjust the resources that are available. For more information, see Part I, “Resource Management,” in *System Administration Guide: Virtualization Using the Solaris Operating System.*
Using Solaris Zones

Solaris zones provide an application execution environment in which processes are isolated from the rest of the system within a single instance of the Solaris OS. This isolation prevents processes that are running in one zone from monitoring or affecting processes that are running in other zones. Even a process running with superuser capabilities cannot view or affect activity in other zones.

Solaris zones are ideal for environments that place several applications on a single server. For more information, see Part II, “Zones,” in System Administration Guide: Virtualization Using the Solaris Operating System.

Monitoring Use of Machine Resources

As a system administrator, you need to monitor system activity. You need to be aware of all aspects of your machines, including the following:

- What is the normal load?
- Who has access to the system?
- When do individuals access the system?
- What programs normally run on the system?

With this kind of knowledge, you can use the available tools to audit system use and monitor the activities of individual users. Monitoring is very useful when a breach in security is suspected. For more information on the auditing service, see Chapter 28, “Solaris Auditing (Overview).”

Monitoring File Integrity

As a system administrator, you need assurance that the files that were installed on the systems that you administer have not changed in unexpected ways. In large installations, a comparison and reporting tool about the software stack on each of your systems enables you to track your systems. The Basic Audit Reporting Tool (BART) enables you to comprehensively validate systems by performing file-level checks of one or more systems over time. Changes in a BART manifest across systems, or for one system over time, can validate the integrity of your systems. BART provides manifest creation, manifest comparison, and rules for scripting reports. For more information, see Chapter 6, “Using the Basic Audit Reporting Tool (Tasks).”
**Controlling Access to Files**

The Solaris OS is a multiuser environment. In a multiuser environment, all the users who are logged into a system can read files that belong to other users. With the appropriate file permissions, users can also use files that belong to other users. For more discussion, see Chapter 7, “Controlling Access to Files (Tasks).” For step-by-step instructions on setting appropriate permissions on files, see “Protecting Files (Task Map)” on page 137.

**Protecting Files With Encryption**

You can keep a file secure by making the file inaccessible to other users. For example, a file with permissions of 600 cannot be read except by its owner and by superuser. A directory with permissions of 700 is similarly inaccessible. However, someone who guesses your password or who discovers the root password can access that file. Also, the otherwise inaccessible file is preserved on a backup tape every time that the system files are backed up to offline media.

The Solaris Cryptographic Framework provides `digest`, `mac`, and `encrypt` commands to protect files. For more information, see Chapter 13, “Solaris Cryptographic Framework (Overview).”

**Using Access Control Lists**

ACLs, pronounced “ackkls,” can provide greater control over file permissions. You add ACLs when traditional UNIX file protections are not sufficient. Traditional UNIX file protections provide read, write, and execute permissions for the three user classes: owner, group, and other. An ACL provides finer-grained file security.

ACLs enable you to define the following file permissions:

- Owner file permissions
- File permissions for the owner’s group
- File permissions for other users who are outside the owner’s group
- File permissions for specific users
- File permissions for specific groups
- Default permissions for each of the previous categories

For more information about using ACLs, see “Using Access Control Lists to Protect Files” on page 133.
Sharing Files Across Machines

A network file server can control which files are available for sharing. A network file server can also control which clients have access to the files, and what type of access is permitted for those clients. In general, the file server can grant read-write access or read-only access either to all clients or to specific clients. Access control is specified when resources are made available with the `share` command.

The `/etc/dfs/dfstab` file on the file server lists the file systems that the server makes available to clients on the network. For more information about sharing file systems, see “Automatic File-System Sharing” in *System Administration Guide: Network Services*.

Restricting root Access to Shared Files

In general, superuser is not allowed root access to file systems that are shared across the network. The NFS system prevents root access to mounted file systems by changing the user of the requester to the user nobody with the user ID 60001. The access rights of user nobody are the same as those access rights that are given to the public. The user nobody has the access rights of a user without credentials. For example, if the public has only execute permission for a file, then user nobody can only execute that file.

An NFS server can grant superuser capabilities on a shared file system on a per-host basis. To grant these privileges, use the `root=hostname` option to the `share` command. You should use this option with care. For a discussion of security options with NFS, see Chapter 6, “Accessing Network File Systems (Reference),” in *System Administration Guide: Network Services*.

Controlling Network Access

Computers are often part of a configuration of computers. This configuration is called a network. A network allows connected computers to exchange information. Networked computers can access data and other resources from other computers on the network. Computer networks create a powerful and sophisticated computing environment. However, networks also complicate computer security.

For example, within a network of computers, individual machines allow the sharing of information. Unauthorized access is a security risk. Because many people have access to a network, unauthorized access is more likely, especially through user error. A poor use of passwords can also allow unauthorized access.
Network Security Mechanisms

Network security is usually based on limiting or blocking operations from remote systems. The following figure describes the security restrictions that you can impose on remote operations.

![Diagram of network security mechanisms]

**Authentication** and **Authorization for Remote Access**

*Authentication* is a way to restrict access to specific users when these users access a remote system. Authentication can be set up at both the system level and the network level. After a user has gained access to a remote system, *authorization* is a way to restrict operations that the user can perform. The following table lists the services that provide authentication and authorization.

---

**TABLE 2–1**  Security Restrictions for Remote Operations

<table>
<thead>
<tr>
<th>Service/Scenario</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remote systems use authentication to restrict access to specific users.</td>
<td></td>
</tr>
<tr>
<td>Can I log in?</td>
<td>Depends on who you are.</td>
</tr>
<tr>
<td>Can I copy that file?</td>
<td>Sure, go ahead.</td>
</tr>
<tr>
<td>The firewall restricts the types of remote operations that the systems at a particular site can perform with systems that are outside the firewall.</td>
<td></td>
</tr>
</tbody>
</table>

---

*FIGURE 2–1*  Security Restrictions for Remote Operations
TABLE 2–3 Authentication and Authorization Services for Remote Access

<table>
<thead>
<tr>
<th>Service</th>
<th>Description</th>
<th>For More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPsec</td>
<td>IPsec provides host-based and certificate-based authentication and network traffic encryption.</td>
<td>Chapter 19, &quot;IP Security Architecture (Overview),&quot; in System Administration Guide: IP Services</td>
</tr>
<tr>
<td>Kerberos</td>
<td>Kerberos uses encryption to authenticate and authorize a user who is logging in to the system.</td>
<td>For an example, see &quot;How the Kerberos Service Works&quot; on page 360.</td>
</tr>
<tr>
<td>LDAP and NIS+</td>
<td>The LDAP directory service and the NIS+ name service can provide both authentication and authorization at the network level.</td>
<td>System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP) and System Administration Guide: Naming and Directory Services (NIS+)</td>
</tr>
<tr>
<td>Remote login commands</td>
<td>The remote login commands enable users to log in to a remote system over the network and use its resources. Some of the remote login commands are rlogin, rcp, and ftp. If you are a &quot;trusted host,&quot; authentication is automatic. Otherwise, you are asked to authenticate yourself.</td>
<td>Chapter 29, &quot;Accessing Remote Systems (Tasks),&quot; in System Administration Guide: Network Services</td>
</tr>
<tr>
<td>SASL</td>
<td>The Simple Authentication and Security Layer (SASL) is a framework that provides authentication and optional security services to network protocols. Plugins enable you to choose an appropriate authentication protocol.</td>
<td>&quot;SASL (Overview)&quot; on page 317</td>
</tr>
<tr>
<td>Secure RPC</td>
<td>Secure RPC improves the security of network environments by authenticating users who make requests on remote machines. You can use either the UNIX, DES, or Kerberos authentication system for Secure RPC. Secure RPC can also be used to provide additional security in an NFS environment. An NFS environment with secure RPC is called Secure NFS. Secure NFS uses Diffie-Hellman authentication for public keys.</td>
<td>&quot;Overview of Secure RPC&quot; on page 291</td>
</tr>
<tr>
<td>Solaris Secure Shell</td>
<td>Solaris Secure Shell encrypts network traffic over an unsecured network. Solaris Secure Shell provides authentication by the use of passwords, public keys, or both. Solaris Secure Shell uses RSA and DSA authentication for public keys.</td>
<td>&quot;Solaris Secure Shell (Overview)&quot; on page 321</td>
</tr>
</tbody>
</table>

A possible substitute for Secure RPC is the Solaris privileged port mechanism. A privileged port is assigned a port number less than 1024. After a client system has authenticated the client’s credential, the client builds a connection to the server by using the privileged port. The server then verifies the client credential by examining the connection’s port number.

Clients that are not running Solaris software might be unable to communicate by using the privileged port. If the clients cannot communicate over the port, you see an error message that is similar to the following:
Firewall Systems

You can set up a firewall system to protect the resources in your network from outside access. A **firewall system** is a secure host that acts as a barrier between your internal network and outside networks. The internal network treats every other network as untrusted. You should consider this setup as mandatory between your internal network and any external networks, such as the Internet, with which you communicate.

A firewall acts as a gateway and as a barrier. A firewall acts as a gateway that passes data between the networks. A firewall acts as a barrier that blocks the free passage of data to and from the network. The firewall requires a user on the internal network to log in to the firewall system to access hosts on remote networks. Similarly, a user on an outside network must first log in to the firewall system before being granted access to a host on the internal network.

A firewall can also be useful between some internal networks. For example, you can set up a firewall or a secure gateway computer to restrict the transfer of packets. The gateway can forbid packet exchange between two networks, unless the gateway computer is the source address or the destination address of the packet. A firewall should also be set up to forward packets for particular protocols only. For example, you can allow packets for transferring mail, but not allow packets for the `telnet` or the `rlogin` command.

In addition, all electronic mail that is sent from the internal network is first sent to the firewall system. The firewall then transfers the mail to a host on an external network. The firewall system also receives all incoming electronic mail, and distributes the mail to the hosts on the internal network.

**Caution** – A firewall prevents unauthorized users from accessing the hosts on your network. You should maintain strict and rigidly enforced security on the firewall, but security on other hosts on the network can be more relaxed. However, an intruder who can break into your firewall system can then gain access to all the other hosts on the internal network.

A firewall system should not have any trusted hosts. A *trusted host* is a host from which a user can log in without being required to supply a password. A firewall system should not share any of its file systems, or mount any file systems from other servers.
The following technologies can be used to harden a system into a firewall:

- The Solaris Security Toolkit, informally known as the JASS toolkit, can harden a Solaris system into a firewall. The toolkit can be downloaded from the Sun website, http://wwws.sun.com/security/jass.

**Encryption and Firewall Systems**

Most local area networks transmit data between computers in blocks that are called packets. Through a procedure that is called packet smashing, unauthorized users from outside the network can corrupt or destroy data.

Packet smashing involves capturing the packets before the packets reach their destination. The intruder then injects arbitrary data into the contents, and sends the packets back on their original course. On a local area network, packet smashing is impossible because packets reach all systems, including the server, at the same time. Packet smashing is possible on a gateway, however, so make sure that all gateways on the network are protected.

The most dangerous attacks affect the integrity of the data. Such attacks involve changing the contents of the packets or impersonating a user. Attacks that involve eavesdropping do not compromise data integrity. An eavesdropper records conversations for later replay. An eavesdropper does not impersonate a user. Although eavesdropping attacks do not attack data integrity, the attacks do affect privacy. You can protect the privacy of sensitive information by encrypting data that goes over the network.

- To encrypt remote operations over an insecure network, see Chapter 19, “Using Solaris Secure Shell (Tasks).”
- To encrypt and authenticate data across a network, see Chapter 21, “Introduction to the Kerberos Service.”

**Reporting Security Problems**

If you experience a suspected security breach, you can contact the Computer Emergency Response Team/Coordination Center (CERT/CC). CERT/CC is a Defense Advanced Research Projects Agency (DARPA) funded project that is located at the Software Engineering Institute at Carnegie Mellon University. This agency can assist you with any security problems you are having. This agency can also direct you to other Computer Emergency Response Teams that might be more appropriate for your particular needs. You can call CERT/CC at its 24-hour hotline: (412) 268-7090. Or, contact the team by email at cert@cert.sei.cmu.edu.
This chapter describes the procedures for controlling who can access Solaris systems. The following is a list of the information in this chapter.

- “Controlling System Access (Task Map)” on page 59
- “Securing Logins and Passwords (Task Map)” on page 60
- “Changing the Password Algorithm (Task Map)” on page 67
- “Monitoring and Restricting Superuser (Task Map)” on page 71
- “SPARC: Controlling Access to System Hardware (Task Map)” on page 74

For overview information about system security, see Chapter 2, “Managing Machine Security (Overview).”

**Controlling System Access (Task Map)**

A computer is as secure as its weakest point of entry. The following task map shows the areas that you should monitor and secure.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitor, permit, and deny user login</td>
<td>Monitors unusual login activity. Prevents logins temporarily. Manages dial-up logins.</td>
<td>“Securing Logins and Passwords (Task Map)” on page 60</td>
</tr>
<tr>
<td>Provide strong password encryption</td>
<td>Specifies algorithms to encrypt user passwords. Installs additional algorithms.</td>
<td>“Changing the Password Algorithm (Task Map)” on page 67</td>
</tr>
<tr>
<td>Monitor and restrict superuser activities</td>
<td>Regularly monitors superuser activity. Prevents remote login by a root user.</td>
<td>“Monitoring and Restricting Superuser (Task Map)” on page 71</td>
</tr>
<tr>
<td>Prevent access to hardware settings</td>
<td>Keeps ordinary users away from the PROM.</td>
<td>“SPARC: Controlling Access to System Hardware (Task Map)” on page 74</td>
</tr>
</tbody>
</table>
Securing Logins and Passwords (Task Map)

The following task map points to procedures that monitor user logins and that disable user logins.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display a user's login status</td>
<td>Lists extensive information about a user's login account, such as full name and password aging information.</td>
<td>&quot;How to Display a User's Login Status&quot; on page 60</td>
</tr>
<tr>
<td>Find users who do not have passwords</td>
<td>Finds only those users whose accounts do not require a password.</td>
<td>&quot;How to Display Users Without Passwords&quot; on page 61</td>
</tr>
<tr>
<td>Disable logins temporarily</td>
<td>Denies user logins to a machine as part of system shutdown or routine maintenance.</td>
<td>&quot;How to Temporarily Disable User Logins&quot; on page 62</td>
</tr>
<tr>
<td>Save failed login attempts</td>
<td>Creates a log of users who failed to provide the correct password after five attempts.</td>
<td>&quot;How to Monitor Failed Login Attempts&quot; on page 63</td>
</tr>
<tr>
<td>Save all failed login attempts</td>
<td>Creates a log of failed attempts to log in.</td>
<td>&quot;How to Monitor All Failed Login Attempts&quot; on page 64</td>
</tr>
<tr>
<td>Create a dial-up password</td>
<td>Requires an additional password for users who log in remotely through a modem or dial-up port.</td>
<td>&quot;How to Create a Dial-Up Password&quot; on page 65</td>
</tr>
<tr>
<td>Disable dial-up logins temporarily</td>
<td>Prevents users from dialing in remotely through a modem or port.</td>
<td>&quot;How to Temporarily Disable Dial-Up Logins&quot; on page 67</td>
</tr>
</tbody>
</table>

Securing Logins and Passwords

You can limit remote logins and require users to have passwords. You can also monitor failed access attempts and disable logins temporarily.

**How to Display a User's Login Status**

1. **Assume the Primary Administrator role, or become superuser.**
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2. **Display a user's login status by using the logins command.**

   ```
   # logins -x -l username
   -x                      Displays an extended set of login status information.
   ```
-l username Displays the login status for the specified user. The variable username is a user's login name. Multiple login names must be specified in a comma-separated list.

The logins command uses the appropriate password database to obtain a user's login status. The database can be the local /etc/passwd file, or a password database for the name service. For more information, see the logins(1M) man page.

Example 3–1 Displaying a User’s Login Status

In the following example, the login status for the user rimmer is displayed.

```
# logins -x -l rimmer
rimmer 500 staff 10 Annalee J. Rimmer
/export/home/rimmer
/bin/sh
PS 010103 10 7 -1
```

rimmer Identifies the user's login name.
500 Identifies the user ID (UID).
staff Identifies the user's primary group.
10 Identifies the group ID (GID).
Annalee J. Rimmer Identifies the comment.
/export/home/rimmer Identifies the user's home directory.
/bin/sh Identifies the login shell.
PS 010170 10 7 -1

Specifies the password aging information:
- Last date that the password was changed
- Number of days that are required between changes
- Number of days before a change is required
- Warning period

▼ How to Display Users Without Passwords

1 Assume the Primary Administrator role, or become superuser.

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.
Display all users who have no passwords by using the `logins` command.

```
# logins -p
```

The `-p` option displays a list of users with no passwords. The `logins` command uses the password database from the local system unless a name service is enabled.

**Example 3–2 Displaying Users Without Passwords**

In the following example, the user `pmorph` does not have a password.

```
# logins -p
pmorph      501 other 1 Polly Morph
#
```

**How to Temporarily Disable User Logins**

Temporarily disable user logins during system shutdown or routine maintenance. Superuser logins are not affected. For more information, see the `nologin(4)` man page.

1. **Assume the Primary Administrator role, or become superuser.**
   
The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in *System Administration Guide: Basic Administration*.

2. **Create the `/etc/nologin` file in a text editor.**
   
   ```
   # vi /etc/nologin
   ```

3. **Include a message about system availability.**

4. **Close and save the file.**

**Example 3–3 Disabling User Logins**

In this example, users are notified of system unavailability.

```
# vi /etc/nologin
(Add system message here)

# cat /etc/nologin
***No logins permitted.***

***The system will be unavailable until 12 noon.***
```
You can also bring the system to run level 0, single-user mode, to disable logins. For information on bringing the system to single-user mode, see Chapter 10, “Shutting Down a System (Tasks),” in System Administration Guide: Basic Administration.

### How to Monitor Failed Login Attempts

This procedure captures failed login attempts from terminal windows. This procedure does not capture failed logins from a GNOME login attempt.

1. **Assume the Primary Administrator role, or become superuser.**
   
The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in System Administration Guide: Basic Administration.

2. **Create the loginlog file in the /var/adm directory.**
   
   ```
   # touch /var/adm/loginlog
   ```

3. **Set read-and-write permissions for root user on the loginlog file.**
   
   ```
   # chmod 600 /var/adm/loginlog
   ```

4. **Change group membership to sys on the loginlog file.**
   
   ```
   # chgrp sys /var/adm/loginlog
   ```

5. **Verify that the log works.**
   
   For example, log in to the system five times with the wrong password. Then, display the /var/adm/loginlog file.
   
   ```
   # more /var/adm/loginlog
   jdoe:/dev/pts/2:Tue Nov 4 10:21:10 2003
   jdoe:/dev/pts/2:Tue Nov 4 10:21:21 2003
   jdoe:/dev/pts/2:Tue Nov 4 10:21:30 2003
   jdoe:/dev/pts/2:Tue Nov 4 10:21:40 2003
   jdoe:/dev/pts/2:Tue Nov 4 10:21:49 2003
   #
   ```

   The `loginlog` file contains one entry for each failed attempt. Each entry contains the user's login name, tty device, and time of the failed attempt. If a person makes fewer than five unsuccessful attempts, no failed attempts are logged.

   A growing `loginlog` file can indicate an attempt to break into the computer system. Therefore, check and clear the contents of this file regularly. For more information, see the `loginlog(4)` man page.
How to Monitor All Failed Login Attempts

This procedure captures in a syslog file all failed login attempts.

1. **Assume the Primary Administrator role, or become superuser.**
The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Set up the /etc/default/login file with the desired values for SYLOG and SYLOG_FAILED_LOGINS**
Edit the /etc/default/login file to change the entry. Make sure that SYLOG=YES is uncommented.

   ```bash
   # grep SYSLOG /etc/default/login
   # SYSLOG determines whether the syslog(3) LOG_AUTH facility
   # should be used
   SYSLOG=YES
   ...
   SYSLOG_FAILED_LOGINS=0
   #
   ```

3. **Create a file with the correct permissions to hold the logging information.**

   a. **Create the authlog file in the /var/adm directory.**
      ```bash
      # touch /var/adm/authlog
      ```

   b. **Set read-and-write permissions for root user on the authlog file.**
      ```bash
      # chmod 600 /var/adm/authlog
      ```

   c. **Change group membership to sys on the authlog file.**
      ```bash
      # chgrp sys /var/adm/authlog
      ```

4. **Edit the syslog.conf file to log failed password attempts.**
The failures should be sent to the authlog file.

   a. **Type the following entry into the syslog.conf file.**
      Fields on the same line in syslog.conf are separated by tabs.
      ```bash
      auth.notice <Press Tab> /var/adm/authlog
      ```

   b. **Refresh the configuration information for the syslog daemon.**
      ```bash
      # svcadm refresh system/system-log
      ```
5 **Verify that the log works.**

For example, as an ordinary user, log in to the system with the wrong password. Then, in the Primary Administrator role or as superuser, display the /var/adm/authlog file.

```
# more /var/adm/authlog
Nov 4 14:46:11 example1 login: [ID 143248 auth.notice]
  Login failure on /dev/pts/8 from example2, stacey
#
```

6 **Monitor the /var/adm/authlog file on a regular basis.**

**Example 3–4** Logging Access Attempts After Three Login Failures

Follow the preceding procedure, except set the value of SYSLOG_FAILED_LOGINS to 3 in the /etc/default/login file.

**Example 3–5** Closing Connection After Three Login Failures

Uncomment the RETRIES entry in the /etc/default/login file, then set the value of RETRIES to 3. Your edits take effect immediately. After three login retries in one session, the system closes the connection.

▼ **How to Create a Dial-Up Password**

**Caution** – When you first establish a dial-up password, be sure to remain logged in to at least one port. Test the password on a different port. If you log off to test the new password, you might not be able to log back in. If you are still logged in to another port, you can go back and fix your mistake.

1 **Assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration.*

2 **Create an /etc/dialups file that contains a list of serial devices.**

Include all the ports that are being protected with dial-up passwords. The /etc/dialups file should appear similar to the following:

```
/dev/term/a
/dev/term/b
/dev/term/c
```
3 Create an /etc/d_passwd file that contains the login programs that you are requiring to have a dial-up password.

Include shell programs that a user could be running at login, for example, uucico, sh, ksh, and csh. The /etc/d_passwd file should appear similar to the following:

```
/usr/lib/uucp/uucico: encrypted-password:
/usr/bin/csh: encrypted-password:
/usr/bin/ksh: encrypted-password:
/usr/bin/sh: encrypted-password:
```

Later in the procedure, you are going to add the encrypted password for each login program.

4 Set ownership to root on the two files.

```
# chown root /etc/dialups /etc/d_passwd
```

5 Set group ownership to root on the two files.

```
# chgrp root /etc/dialups /etc/d_passwd
```

6 Set read-and-write permissions for root on the two files.

```
# chmod 600 /etc/dialups /etc/d_passwd
```

7 Create the encrypted passwords.

   a. Create a temporary user.

```
# useradd username
```

   b. Create a password for the temporary user.

```
# passwd username
New Password:  <Type password>
Re-enter new Password:  <Retype password>
passwd: password successfully changed for username
```

   c. Capture the encrypted password.

```
# grep username /etc/shadow > username.temp
```

   d. Edit the username.temp file.

Delete all fields except the encrypted password. The second field holds the encrypted password.

For example, in the following line, the encrypted password is U9gp9SyA/JlSk.

```
temp:U9gp9SyA/JlSk:7967::::7988:
```

   e. Delete the temporary user.

```
# userdel username
```
Copy the encrypted password from username.temp file into the /etc/d_passwd file.
You can create a different password for each login shell. Alternatively, use the same password
for each login shell.

Inform your dial-up users of the password.
You should ensure that your means of informing the users cannot be tampered with.

How to Temporarily Disable Dial-Up Logins

Assume the Primary Administrator role, or become superuser.
The Primary Administrator role includes the Primary Administrator profile. To create the role
and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console
(Tasks)," in System Administration Guide: Basic Administration.

Put the following single-line entry into the /etc/d_passwd file:
/usr/bin/sh:*:

Changing the Password Algorithm (Task Map)
The following task map points to procedures to administer password algorithms.

<table>
<thead>
<tr>
<th>Task</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provide strong password encryption</td>
<td>&quot;How to Specify an Algorithm for Password Encryption&quot; on page 68</td>
</tr>
<tr>
<td>Provide strong password encryption with a name service</td>
<td>&quot;How to Specify a New Password Algorithm for an NIS Domain&quot; on page 69</td>
</tr>
<tr>
<td></td>
<td>&quot;How to Specify a New Password Algorithm for an NIS+ Domain&quot; on page 69</td>
</tr>
<tr>
<td></td>
<td>&quot;How to Specify a New Password Algorithm for an LDAP Domain&quot; on page 70</td>
</tr>
<tr>
<td>Add new password encryption module</td>
<td>&quot;How to Install a Password Encryption Module From a Third Party&quot; on page 70</td>
</tr>
</tbody>
</table>
Changing the Default Algorithm for Password Encryption

By default, user passwords are encrypted with the crypt_unix algorithm. You can use a stronger encryption algorithm, such as MD5 or Blowfish, by changing the default password encryption algorithm.

▼ How to Specify an Algorithm for Password Encryption

In this procedure, the BSD-Linux version of the MD5 algorithm is the default encryption algorithm that is used when users change their passwords. This algorithm is suitable for a mixed network of machines that run the Solaris, BSD, and Linux versions of UNIX. For a list of password encryption algorithms and algorithm identifiers, see Table 2–1.

1 Assume the Primary Administrator role, or become superuser.

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2 Specify the identifier for your chosen encryption algorithm.

Type the identifier as the value for the CRYPT_DEFAULT variable in the /etc/security/policy.conf file.

You might want to comment the file to explain your choice.

```
# cat /etc/security/policy.conf
...
CRYPT_ALGORITHMS_ALLOW=1,2a,md5,5,6
#
# Use the version of MD5 that works with Linux and BSD systems.
# Passwords previously encrypted with __unix__ will be encrypted with MD5
# when users change their passwords.
#
# CRYPT_DEFAULT=__unix__
CRYPT_DEFAULT=1
```

In this example, the algorithms configuration ensures that the weakest algorithm, crypt_unix, is never used to encrypt a password. Users whose passwords were encrypted with the crypt_unix module get a crypt_bsdmd5-encrypted password when they change their passwords.

For more information on configuring the algorithm choices, see the policy.conf(4) man page.
Example 3–6  Using the Blowfish Algorithm for Password Encryption

In this example, the identifier for the Blowfish algorithm, 2a, is specified as the value for the CRYPT_DEFAULT variable in the policy.conf file:

```
CRYPT_ALGORITHMS_ALLOW=1,2a,md5,5,6
#CRYPT_ALGORITHMS_DEPRECATE=unix
CRYPT_DEFAULT=2a
```

This configuration is compatible with BSD systems that use the Blowfish algorithm.

▼ How to Specify a New Password Algorithm for an NIS Domain

When users in an NIS domain change their passwords, the NIS client consults its local algorithms configuration in the /etc/security/policy.conf file. The NIS client machine encrypts the password.

1 Specify the password encryption algorithm in the /etc/security/policy.conf file on the NIS client.

2 Copy the modified /etc/security/policy.conf file to every client machine in the NIS domain.

3 To minimize confusion, copy the modified /etc/security/policy.conf file to the NIS root server and to the slave servers.

▼ How to Specify a New Password Algorithm for an NIS+ Domain

When users in an NIS+ domain change their passwords, the NIS+ name service consults the algorithms configuration in the /etc/security/policy.conf file on the NIS+ master. The NIS+ master, which is running the rpc.nispasswd daemon, creates the encrypted password.

1 Specify the password encryption algorithm in the /etc/security/policy.conf file on the NIS+ master.

2 To minimize confusion, copy the NIS+ master’s /etc/security/policy.conf file to every host in the NIS+ domain.
How to Specify a New Password Algorithm for an LDAP Domain

When the LDAP client is properly configured, the LDAP client can use the new password algorithms. The LDAP client behaves just as an NIS client behaves.

1. Specify a password encryption algorithm in the `/etc/security/policy.conf` file on the LDAP client.

2. Copy the modified `policy.conf` file to every client machine in the LDAP domain.

3. Ensure that the client's `/etc/pam.conf` file does not use a `pam_ldap` module.

   Ensure that a comment sign (#) precedes entries that include `pam_ldap.so.1`. Also, do not use the new server_policy option with the `pam_authtok_store.so.1` module.

   The PAM entries in the client's `pam.conf` file enable the password to be encrypted according to the local algorithms configuration. The PAM entries also enable the password to be authenticated.

   When users in the LDAP domain change their passwords, the LDAP client consults its local algorithms configuration in the `/etc/security/policy.conf` file. The LDAP client machine encrypts the password. Then, the client sends the encrypted password, with a `{crypt}` tag, to the server. The tag tells the server that the password is already encrypted. The password is then stored, as is, on the server. For authentication, the client retrieves the stored password from the server. The client then compares the stored password with the encrypted version that the client has just generated from the user's typed password.

   **Note** – To take advantage of password policy controls on the LDAP server, use the `server_policy` option with the `pam_authtok_store` entries in the `pam.conf` file. Passwords are then encrypted on the server by using the Sun Java™ System Directory Server's cryptographic mechanism. For the procedure, see Chapter 11, “Setting Up Sun Java System Directory Server With LDAP Clients (Tasks),” in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

How to Install a Password Encryption Module From a Third Party

A third-party password encryption algorithm is typically delivered as a module in a software package. When you run the `pkgadd` command, scripts from the vendor should modify the `/etc/security/crypt.conf` file. You then modify the `/etc/security/policy.conf` file to include the new module and its identifier.
1 **Add the software by using the `pkgadd` command.**

For detailed instructions on how to add software, see “Adding or Removing a Software Package (`pkgadd`)” in *System Administration Guide: Basic Administration*.

2 **Confirm that the new module and module identifier have been added.**

Read the list of encryption algorithms in the `/etc/security/crypt.conf` file.

For example, the following lines show that a module that implements the `crypt_rot13` algorithm has been installed.

```plaintext
# crypt.conf
#
md5 /usr/lib/security/$ISA/crypt_md5.so
rot13 /usr/lib/security/$ISA/crypt_rot13.so
```

# For *BSD - Linux compatibility
# 1 is MD5, 2a is Blowfish
1 /usr/lib/security/$ISA/crypt_bsdmd5.so
2a /usr/lib/security/$ISA/crypt_bsdbf.so

3 **Add the identifier of the newly installed algorithm to the `/etc/security/policy.conf` file.**

The following lines show excerpts from the `policy.conf` file that would need to be modified to add the `rot13` identifier.

```plaintext
# Copyright 1999-2002 Sun Microsystems, Inc. All rights reserved.
#
ident "@(#)policy.conf 1.12 08/05/14 SMI"
#...
# crypt(3c) Algorithms Configuration
CRYPT_ALGORITHMS_ALLOW=1,2a,md5,5,6,,rot13
CRYPT_ALGORITHMS_DEPRECATED=unix,
CRYPT_DEFAULT=md5
```

In this example, the `rot13` algorithm is used if the current password was encrypted with the `crypt_rot13` algorithm. New user passwords are encrypted with the `crypt_sunmd5` algorithm. This algorithm configuration works on Solaris-only networks.

# Monitoring and Restricting Superuser (Task Map)

The following task map describes how to monitor and restrict the root user login.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitor who is using the <code>su</code> command</td>
<td>Scans the <code>sulog</code> file on a regular basis.</td>
<td>“How to Monitor Who Is Using the <code>su</code> Command” on page 72</td>
</tr>
</tbody>
</table>
Monitor superuser activity on the console

- Displays superuser activity on the console.
- Monitors superuser access attempts as they occur.

Task Description For Instructions
---
Display superuser activity on the console | Monitors superuser access attempts as the attempts occur. | "How to Restrict and Monitor Superuser Logins" on page 73

Monitoring and Restricting Superuser

An alternative to using the superuser account is to set up role-based access control. Role-based access control is called RBAC. For overview information on RBAC, see "Role-Based Access Control (Overview)" on page 155. To set up RBAC, see Chapter 9, "Using Role-Based Access Control (Tasks)."

**How to Monitor Who Is Using the su Command**

The sulog file lists every use of the su command, not only the su attempts that are used to switch from user to superuser.

- **Monitor the contents of the /var/adm/sulog file on a regular basis.**

  ```
  # more /var/adm/sulog
  SU 12/20 16:26 + pts/0 stacey-root
  SU 12/21 10:59 + pts/0 stacey-root
  SU 01/12 11:11 + pts/0 root-rimmer
  SU 01/12 14:56 + pts/0 pmorph-root
  SU 01/12 14:57 + pts/0 pmorph-root
  ```

  The entries display the following information:
  - The date and time that the command was entered.
  - If the attempt was successful. A plus sign (+) indicates a successful attempt. A minus sign (-) indicates an unsuccessful attempt.
  - The port from which the command was issued.
  - The name of the user and the name of the switched identity.

  The su logging in this file is enabled by default through the following entry in the /etc/default/su file:

  ```
  SULOG=/var/adm/sulog
  ```

  **Troubleshooting**

  Entries that include ??? indicate that the controlling terminal for the su command cannot be identified. Typically, system invocations of the su command before the desktop appears include ???, as in SU 10/10 08:08 + ??? root-root. After the user starts a desktop session, the ttypnam command returns the value of the controlling terminal to the sulog: SU 10/10 10:10 + pts/3 jdoe-root.
Entries similar to the following can indicate that the su command was not invoked on the command line: `SU 10/10 10:20 + ??? root-oracle`. The user might have switched to the oracle role by using a GUI.

### How to Restrict and Monitor Superuser Logins

This method immediately detects superuser attempts to access the local system.

1. **View the CONSOLE entry in the `/etc/default/login` file.**
   
   ```
   CONSOLE=/dev/console
   
   By default, the console device is set to `/dev/console`. With this setting, root can log in to the console. Root cannot log in remotely.
   ```

2. **Verify that root cannot log in remotely.**
   
   From a remote system, try to log in as superuser.
   
   ```
   mach2 % rlogin -l root mach1
   Password: <Type root password of mach1>
   Not on system console
   Connection closed.
   ```

3. **Monitor attempts to become superuser.**
   
   By default, attempts to become superuser are printed to the console by the SYSLOG utility.
   
   a. **Open a terminal console on your desktop.**

   b. **In another window, use the su command to become superuser.**

   ```
   % su -
   Password: <Type root password>
   #
   
   A message is printed on the terminal console.
   ```

   ```
   Sep 7 13:22:57 mach1 su: ‘su root’ succeeded for jdoe on /dev/pts/6
   ```

### Example 3–7 Logging Superuser Access Attempts

In this example, superuser attempts are not being logged by SYSLOG. Therefore, the administrator is logging those attempts by removing the comment from the `#CONSOLE=/dev/console` entry in the `/etc/default/su` file.

```
# CONSOLE determines whether attempts to su to root should be logged
# to the named device
#
CONSOLE=/dev/console
```
When a user attempts to become superuser, the attempt is printed on the terminal console.

SU 09/07 16:38 + pts/8 jdoe-root

**Troubleshooting**  To become superuser from a remote system when the `/etc/default/login` file contains the default CONSOLE entry, users must first log in with their user name. After logging in with their user name, users can then use the `su` command to become superuser.

If the console displays an entry similar to `Mar 16 16:20:36 mach1 login: ROOT LOGIN /dev/pts/14 FROM mach2.example.com`, then the system is permitting remote root logins. To prevent remote superuser access, change the `#CONSOLE=/dev/console` entry to `CONSOLE=/dev/console` in the `/etc/default/login` file.

**SPARC: Controlling Access to System Hardware (Task Map)**

The following task map describes how to protect the PROM from unwanted access.

<table>
<thead>
<tr>
<th>Task Description</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prevent users from changing system hardware settings</td>
<td>Requires a password to modify PROM settings.</td>
<td>&quot;How to Require a Password for Hardware Access&quot; on page 74</td>
</tr>
<tr>
<td>Disable the abort sequence</td>
<td>Prevents users from accessing the PROM.</td>
<td>&quot;How to Disable a System's Abort Sequence&quot; on page 75</td>
</tr>
</tbody>
</table>

**Controlling Access to System Hardware**

You can protect the physical machine by requiring a password to gain access to the hardware settings. You can also protect the machine by preventing a user from using the abort sequence to leave the windowing system.

**How to Require a Password for Hardware Access**

On an x86 system, the equivalent to protecting the PROM is to protect the BIOS. Refer to your machine’s manuals for how to protect the BIOS.

1. **Become superuser or assume a role that includes the Device Security profile, the Maintenance and Repair profile, or the System Administrator profile.**

The System Administrator profile includes the Maintenance and Repair profile. To create a role that includes the System Administrator profile and to assign the role to a user, see "Configuring RBAC (Task Map)" on page 174.
2 In a terminal window, type the PROM security mode.

```bash
# eeprom security-mode=command
```

Changing PROM password:

New password: <Type password>

Retype new password: <Retype password>

Choose the value `command` or `full`. For more details, see the `eeeprom(1M)` man page.

If, when you type the preceding command, you are not prompted for a PROM password, the system already has a PROM password.

3 (Optional) To change the PROM password, type the following command:

```bash
# eeprom security-password=      Press Return
```

Changing PROM password:

New password: <Type password>

Retype new password: <Retype password>

The new PROM security mode and password are in effect immediately. However, they are most likely to be noticed at the next boot.

Caution – Do not forget the PROM password. The hardware is unusable without this password.

### How to Disable a System's Abort Sequence

Some server systems have a key switch. When the key switch is set in the secure position, the switch overrides the software keyboard abort settings. So, any changes that you make with the following procedure might not be implemented.

1 Assume the Primary Administrator role, or become superuser.

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 Change the value of `KEYBOARD_ABORT` to `disable`.

Comment out the enable line in the `/etc/default/kbd` file. Then, add a disable line:

```bash
# cat /etc/default/kbd
...
# KEYBOARD_ABORT affects the default behavior of the keyboard abort
# sequence, see kbd(1) for details. The default value is "enable".
# The optional value is "disable". Any other value is ignored.
...
```
Controlling Access to System Hardware

```
#KEYBOARD_ABORT=enable
KEYBOARD_ABORT=disable
```

3. **Update the keyboard defaults.**
   
   ```
   # kbd -i
   ```
This chapter provides information about using antivirus software, and covers the following topics:

- “About Virus Scanning” on page 77
- “About the Vscan Service” on page 78
- “Using the Vscan Service” on page 78

### About Virus Scanning

Data is protected from viruses by a scanning service, vscan, that uses various scan engines. A scan engine is a third-party application, residing on an external host, that examines a file for known viruses. A file is a candidate for virus scanning if the file system supports the vscan service, the service has been enabled, and the type of file has not been exempted. The virus scan is then performed on a file during open and close operations if the file has not been scanned with the current virus definitions previously or if the file has been modified since it was last scanned.

The vscan service can be configured to use multiple scan engines. It is recommended that the vscan service use a minimum of two scan engines. The requests for virus scans are distributed among all available scan engines. Table 4–1 shows the scan engines that are supported when configured with their most recent patch.

<table>
<thead>
<tr>
<th>Antivirus Software</th>
<th>ICAP Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symantec Antivirus Scan Engine 4.3</td>
<td>Is supported</td>
</tr>
<tr>
<td>Symantec Antivirus Scan Engine 5.1</td>
<td>Is supported</td>
</tr>
</tbody>
</table>
TABLE 4–1  Antivirus Scan Engine Software  

(Continued)

<table>
<thead>
<tr>
<th>Antivirus Software</th>
<th>ICAP Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer Associates eTrust AntiVirus 7.1</td>
<td>Is not supported¹</td>
</tr>
<tr>
<td>Computer Associates Integrated Threat Management 8.1</td>
<td></td>
</tr>
<tr>
<td>Trend Micro Interscan Web Security Suite (IWSS) 2.5</td>
<td>Is supported</td>
</tr>
<tr>
<td>McAfee Secure Internet Gateway 4.5</td>
<td>Is supported</td>
</tr>
</tbody>
</table>


About the Vscan Service

The benefit of the real-time scan method is that a file is scanned with the latest virus definitions before it is used. By using this approach, viruses can be detected before they compromise data.

The following describes the virus scanning process:

1. When a user opens a file from the client, the vscan service determines whether the file needs to be scanned, based on whether the file has been scanned with the current virus definitions previously and if the file has been modified since it was last scanned.
   - If the file needs to be scanned, the file is transferred to the scan engine. If a connection to a scan engine fails, the file is sent to another scan engine. If no scan engine is available, the virus scan fails and access to the file might be denied.
   - If the file does not need to be scanned, the client is permitted to access the file.

2. The scan engine scans the file using the current virus definitions.
   - If a virus is detected, the file is marked as quarantined. A quarantined file cannot be read, executed, or renamed but it can be deleted. The system log records the name of the quarantined file and the name of the virus and, if auditing has been enabled, an audit record with the same information is created.
   - If the file is not infected, the file is tagged with a scan stamp and the client is permitted to access the file.

Using the Vscan Service

Scanning files for viruses is available when the following requirements are met:

- At least one scan engine is installed and configured.
- The files reside on a file system that supports virus scanning.
- Virus scanning is enabled on the file system.
- The vscan service is enabled.
- The vscan service is configured to scan files of the specified file type.
The following table points to the tasks you perform to set up the vsan service.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install a scan engine.</td>
<td>Install and configure one or more of the supported third-party products listed in Table 4–1.</td>
<td>See the product documentation.</td>
</tr>
<tr>
<td>Enable the file system to allow virus scans.</td>
<td>Use the file system command to enable virus scans if necessary. For example, scans are disabled by default in the ZFS file system.</td>
<td>“How to Enable Virus Scanning on a File System” on page 79</td>
</tr>
<tr>
<td>Enable the vsan service.</td>
<td>Use the svcadm(1M) command to start the scan service.</td>
<td>“How to Enable the Vsan Service” on page 80</td>
</tr>
<tr>
<td>Add a scan engine to the vsan service</td>
<td>Use the svcadm(1M) command with its add-engine subcommand to include the scan engine in the vsan service.</td>
<td>“How to Add a Scan Engine” on page 80</td>
</tr>
<tr>
<td>Configure the vsan service.</td>
<td>Use the svcadm(1M) command to view and change vsan properties.</td>
<td>“How to View Vscan Properties” on page 80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>“How to Change Vscan Properties” on page 81</td>
</tr>
<tr>
<td>Configure the vsan service for specific file types.</td>
<td>Use the svcadm(1M) command to set the file type.</td>
<td>“How to Exclude Files From Virus Scans” on page 81</td>
</tr>
</tbody>
</table>

**How to Enable Virus Scanning on a File System**

Use the file system command to allow virus scans of files. For example, to include a ZFS file system in a virus scan, use the zfs(1M) command.

1. **Become superuser or assume an equivalent role.**

   Roles contain authorizations and privileged commands. The ZFS file system allows some administrative tasks to be delegated to specific users. For more information about Delegated Administration, see the ZFS Administration Guide. For more information about roles, see “Configuring RBAC (Task Map)” on page 174. To configure a role with the Primary Administrator profile, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2. **Enable virus scanning on a ZFS file system, for example, pool/volumes/vol1.**

   ```
   # zfs set vscan=on path/pool/volumes/vol1
   ```
How to Enable the Vscan Service

1. Use the “VSCAN Management” RBAC profile to obtain the authorizations needed for managing the vscan service.
   Roles contain authorizations and privileged commands. For more information about roles, see “Configuring RBAC (Task Map)” on page 174.

2. Use the `svcadm(1M)` command to enable virus scanning.
   
   ```
   # svcadm enable vscan
   ```

How to Add a Scan Engine

1. Use the “VSCAN Management” RBAC profile to obtain the authorizations needed for managing the vscan service.
   Roles contain authorizations and privileged commands. For more information about roles, see “Configuring RBAC (Task Map)” on page 174.

2. To add a scan engine to the vscan service with default properties, type:
   
   ```
   # vscanadm add-engine engine_ID
   ```
   See the manpage for the `vscanadm(1M)` command for a description of the command.

How to View Vscan Properties

- View the properties of the vscan service, of all scan engines, or of a specific scan engine.

  - To view the properties of a particular scan engine, type:
    
    ```
    # vscanadm get-engine engineID
    ```

  - To view the properties of all scan engines, type:
    
    ```
    # vscanadm get-engine
    ```

  - To view one of the properties of the vscan service, type:
    
    ```
    # vscanadm get -p property
    ```
    where `property` is one of the parameters described in the manpage for the `vscanadm(1M)` command.

    For example, if you want to see the maximum size of a file that can be scanned, type:

    ```
    # vscanadm get max-size
    ```
**How to Change Vscan Properties**

You can change the properties of a particular scan engine and the general properties of the vscan service. Many scan engines limit the size of the files they scan, so the vscan service's `max-size` property must be set to a value less than or equal to the scan engine's maximum allowed size. You then define whether files that are larger than the maximum size, and therefore not scanned, are accessible.

1. **Use the “VSCAN Management” RBAC profile to obtain the authorizations needed for managing the vscan service.**
   Roles contain authorizations and privileged commands. For more information about roles, see “Configuring RBAC (Task Map)” on page 174.

2. **View the current properties by using the `vscanadm show` command.**

3. **Set the maximum size for virus scans to, for example, 128 megabytes.**
   ```
   # vscanadm set -p max-size=128M
   ```

4. **Specify that access is denied to any file that is not scanned due to its size.**
   ```
   # vscanadm set -p max-size-action=deny
   ```
   See the manpage for the `vscanadm(1M)` command for a description of the command.

**How to Exclude Files From Virus Scans**

When you enable antivirus protection, you can specify that all files of specific types are excluded from the virus scan. Because the vscan service affects the performance of the system, you can conserve system resources by targeting specific file types for virus scans.

1. **Use the “VSCAN Management” RBAC profile to obtain the authorizations needed for managing the vscan service.**
   Roles contain authorizations and privileged commands. For more information about roles, see “Configuring RBAC (Task Map)” on page 174.

2. **View the list of all file types included in the virus scan.**
   ```
   # vscanadm get -p types
   ```

3. **Specify the types of files to be scanned for virus:**
   ```
   ▪ Exclude a specific file type, for example the JPEG type, from the virus scan.
     ```
     # vscanadm set -p types=-.jpg,+
     ```
   ```
Include a specific file type, for example executable files, in the virus scan.

# vscanadm set -p types=+exe,-*

See the manpage for the vscanadm(1M) command for a description of the command.
This chapter provides step-by-step instructions for protecting devices, in addition to a reference section. The following is a list of the information in this chapter.

- “Configuring Devices (Task Map)” on page 83
- “Configuring Device Policy (Task Map)” on page 84
- “Managing Device Allocation (Task Map)” on page 87
- “Allocating Devices (Task Map)” on page 93
- “Device Protection (Reference)” on page 97

For overview information about device protection, see “Controlling Access to Devices” on page 45.

### Configuring Devices (Task Map)

The following task map points to tasks for managing access to devices.

<table>
<thead>
<tr>
<th>Task</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manage device policy</td>
<td>“Configuring Device Policy (Task Map)” on page 84</td>
</tr>
<tr>
<td>Manage device allocation</td>
<td>“Managing Device Allocation (Task Map)” on page 87</td>
</tr>
<tr>
<td>Use device allocation</td>
<td>“Allocating Devices (Task Map)” on page 93</td>
</tr>
</tbody>
</table>
Configuring Device Policy (Task Map)

The following task map points to device configuration procedures that are related to device policy.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>View the device policy for the devices on your system</td>
<td>Lists the devices and their device policy.</td>
<td>&quot;How to View Device Policy&quot; on page 84</td>
</tr>
<tr>
<td>Require privilege for device use</td>
<td>Uses privileges to protect a device.</td>
<td>&quot;How to Change the Device Policy on an Existing Device&quot; on page 85</td>
</tr>
<tr>
<td>Remove privilege requirements from a device</td>
<td>Removes or lessens the privileges that are required to access a device.</td>
<td>Example 5–3</td>
</tr>
<tr>
<td>Audit changes in device policy</td>
<td>Records changes in device policy in the audit trail</td>
<td>&quot;How to Audit Changes in Device Policy&quot; on page 86</td>
</tr>
<tr>
<td>Access /dev/arp</td>
<td>Gets Solaris IP MIB-II information.</td>
<td>&quot;How to Retrieve IP MIB-II Information From a /dev/* Device&quot; on page 87</td>
</tr>
</tbody>
</table>

Configuring Device Policy

Device policy restricts or prevents access to devices that are integral to the system. The policy is enforced in the kernel.

▼ How to View Device Policy

- Display the device policy for all devices on your system.
  
  ```
  % getdevpolicy | more
  DEFAULT
      read_priv_set=none
      write_priv_set=none
  ip:*
      read_priv_set=net_rawaccess
      write_priv_set=net_rawaccess
  ...
  ```

Example 5–1 Viewing the Device Policy for a Specific Device

In this example, the device policy for three devices is displayed.

```
% getdevpolicy /dev/allkmem /dev/ipsecesp /dev/hme
/dev/allkmem
```
read_priv_set=all
write_priv_set=all
/dev/ipsecesp
    read_priv_set=sys_net_config
    write_priv_set=sys_net_config
/dev/hme
    read_priv_set=net_rawaccess
    write_priv_set=net_rawaccess

▼ How to Change the Device Policy on an Existing Device

1 Assume a role that includes the Device Security rights profile, or become superuser.
The Primary Administrator role includes the Device Security rights profile. You can also assign
the Device Security rights profile to a role that you create. To create the role and assign the role
to a user, see Example 9–3.

2 Add policy to a device.
   # update_drv -a -p policy device-driver
   -a specifies a policy for device-driver.
   -p policy is the device policy for device-driver. Device policy specifies two sets of
   privileges. One set is required to read the device. The other set is required to
   write to the device.
   device-driver is the device driver.
   For more information, see the update_drv(1M) man page.

Example 5–2 Adding Policy to an Existing Device

In the following example, device policy is added to the ipnat device.

   # getdevpolicy /dev/ipnat
   /dev/ipnat
     read_priv_set=none
     write_priv_set=none
   # update_drv -a \
   -p 'read_priv_set=net_rawaccess write_priv_set=net_rawaccess' ipnat
   # getdevpolicy /dev/ipnat
   /dev/ipnat
     read_priv_set=net_rawaccess
     write_priv_set=net_rawaccess
Example 5–3 Removing Policy From a Device

In the following example, the read set of privileges is removed from the device policy for the ipnat device.

```
# getdevpolicy /dev/ipnat
/dev/ipnat
    read_priv_set=net_rawaccess
    write_priv_set=net_rawaccess
# update_drv -a -p write_priv_set=net_rawaccess ipnat
# getdevpolicy /dev/ipnat
/dev/ipnat
    read_priv_set=none
    write_priv_set=net_rawaccess
```

How to Audit Changes in Device Policy

By default, the as audit class includes the AUE_MODDEVPLCY audit event.

1. **Assume the Primary Administrator role, or become superuser.**
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in System Administration Guide: Basic Administration.

2. **Preselect the audit class that includes AUE_MODDEVPLCY audit event.**
   Add the as class to the flags line of the audit_control file. The file would appear similar to the following:
   ```
   # audit_control file
dir:/var/audit
   flags:lo,as
   minfree:20
   naflags:lo
   ```
   For detailed instructions, see "How to Modify the audit_control File" on page 577.
How to Retrieve IP MIB-II Information From a /dev/* Device

Applications that retrieve Solaris IP MIB-II information should open /dev/arp, not /dev/ip.

1. **Determine the device policy on /dev/ip and /dev/arp.**
   
   ```bash
   % getdevpolicy /dev/ip /dev/arp
   /dev/ip
       read_priv_set=net_rawaccess
       write_priv_set=net_rawaccess
   /dev/arp
       read_priv_set=none
       write_priv_set=none
   ``

   Note that the net_rawaccess privilege is required for reading and writing to /dev/ip. No privileges are required for /dev/arp.

2. **Open /dev/arp and push the tcp and udp modules.**
   
   No privileges are required. This method is equivalent to opening /dev/ip and pushing the arp, tcp and udp modules. Because opening /dev/ip now requires a privilege, the /dev/arp method is preferred.

---

Managing Device Allocation (Task Map)

The following task map points to procedures that enable and configure device allocation. Device allocation is not enabled by default. After device allocation is enabled, see "Allocating Devices (Task Map)" on page 93.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make a device allocatable</td>
<td>Enables a device to be allocated to one user at a time.</td>
<td>&quot;How to Make a Device Allocatable&quot; on page 88</td>
</tr>
<tr>
<td>Authorize users to allocate a device</td>
<td>Assigns device allocation authorizations to users.</td>
<td>&quot;How to Authorize Users to Allocate a Device&quot; on page 89</td>
</tr>
<tr>
<td>View the allocatable devices on your system</td>
<td>Lists the devices that are allocatable, and the state of the device.</td>
<td>&quot;How to View Allocation Information About a Device&quot; on page 90</td>
</tr>
<tr>
<td>Forcibly allocate a device</td>
<td>Allocates a device to a user who has an immediate need</td>
<td>&quot;Forcibly Allocating a Device&quot; on page 90</td>
</tr>
<tr>
<td>Forcibly deallocate a device</td>
<td>Deallocates a device that is currently allocated to a user</td>
<td>&quot;Forcibly Deallocating a Device&quot; on page 91</td>
</tr>
</tbody>
</table>
Managing Device Allocation

Device allocation restricts or prevents access to peripheral devices. Restrictions are enforced at user allocation time. By default, users must have authorization to access allocatable devices.

## How to Make a Device Allocatable

If you have already run the `bsmconv` command to enable auditing, then device allocation is already enabled on your system. For more information, see the `bsmconv(1M)` man page.

1. **Assume a role that includes the Audit Control rights profile, or become superuser.**
   The Primary Administrator role includes the Audit Control rights profile. You can also assign the Audit Control rights profile to a role that you create. To create the role and assign the role to a user, see Example 9–3.

2. **Enable device allocation.**

   ```
   # bsmconv
   This script is used to enable the Basic Security Module (BSM).
   Shall we continue with the conversion now? [y/n] y
   bsmconv: INFO: turning on audit module.
   bsmconv: INFO: initializing device allocation files.
   
   The Basic Security Module is ready.
   If there were any errors, please fix them now.
   Configure BSM by editing files located in /etc/security.
   Reboot this system now to come up with BSM enabled.
   ```
Note – The Volume Management daemon (/etc/rc3.d/S81volmgt) is disabled by this command.

▼ How to Authorize Users to Allocate a Device

1 **Assume the Primary Administrator role, or become superuser.**
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2 **Create a rights profile that contains the appropriate authorization and commands.**
   Typically, you would create a rights profile that includes the solaris.device.allocate authorization. Follow the instructions in “How to Create or Change a Rights Profile” on page 196. Give the rights profile appropriate properties, such as the following:
   - Rights profile name: Device Allocation
   - Granted authorizations: solaris.device.allocate
   - Commands with security attributes: mount with the sys_mount privilege, and umount with the sys_mount privilege

3 **Create a role for the rights profile.**
   Follow the instructions in “How to Create and Assign a Role by Using the GUI” on page 177. Use the following role properties as a guide:
   - Role name: devicealloc
   - Role full name: Device Allocator
   - Role description: Allocates and mounts allocated devices
   - Rights profile: Device Allocation
     This rights profile must be at the top of the list of profiles that are included in the role.

4 **Assign the role to every user who is permitted to allocate a device.**

5 **Teach the users how to use device allocation.**
   For examples of allocating removable media, see “How to Allocate a Device” on page 93.
   Because the Volume Management daemon (vold) is not running, removable media are not automatically mounted. For examples of mounting a device that has been allocated, see “How to Mount an Allocated Device” on page 94.
How to View Allocation Information About a Device

Before You Begin  Device allocation must be enabled for this procedure to succeed. To enable device allocation, see “How to Make a Device Allocatable” on page 88.

1  Assume a role that includes the Device Security rights profile, or become superuser.
   The Primary Administrator role includes the Device Security rights profile. You can also assign the Device Security rights profile to a role that you create. To create the role and assign the role to a user, see Example 9–3.

2  Display information about allocatable devices on your system.
   
   # list_devices device-name
   
   where device-name is one of the following:
   - audio[n] – Is a microphone and speaker.
   - fd[n] – Is a diskette drive.
   - st[n] – Is a tape drive.

Troubleshooting  If the list_devices command returns an error message similar to the following, then either device allocation is not enabled, or you do not have sufficient permissions to retrieve the information.

   list_devices: No device maps file entry for specified device.

   For the command to succeed, enable device allocation and assume a role with the solaris.device.revoke authorization.

Forcibly Allocating a Device

Forcible allocation is used when someone has forgotten to deallocate a device. Forcible allocation can also be used when a user has an immediate need for a device.

Before You Begin  The user or role must have the solaris.device.revoke authorization.

1  Determine if you have the appropriate authorizations in your role.
   
   $ auths
   solaris.device.allocate solaris.device.revoke

2  Forcibly allocate the device to the user who needs the device.
   In this example, the tape drive is forcibly allocated to the user jdoe.
   
   $ allocate -U jdoe
Forcibly Deallocating a Device

Devices that a user has allocated are not automatically deallocated when the process terminates or when the user logs out. Forcible deallocation is used when a user has forgotten to deallocate a device.

Before You Begin

The user or role must have the `solaris.device.revoke` authorization.

1 Determine if you have the appropriate authorizations in your role.
   
   $ auths
   
   `solaris.device.allocate solaris.device.revoke`

2 Forcibly deallocate the device.
   
   In this example, the printer is forcibly deallocated. The printer is now available for allocation by another user.
   
   $ deallocate -f /dev/lp/printer-1

How to Change Which Devices Can Be Allocated

1 Assume a role that includes the Device Security rights profile, or become superuser.
   
   The Primary Administrator role includes the Device Security rights profile. You can also assign the Device Security rights profile to a role that you create. To create the role and assign the role to a user, see Example 9–3.

2 Specify if authorization is required, or specify the `solaris.device.allocate` authorization.
   
   Change the fifth field in the device entry in the `device_allocate` file.
   
   `audio;audio;reserved;reserved;solaris.device.allocate;/etc/security/lib/audio_clean`  
   `fd0;fd;reserved;reserved;solaris.device.allocate;/etc/security/lib/fd_clean`  
   `sr0;sr;reserved;reserved;solaris.device.allocate;/etc/security/lib/sr_clean`

   where `solaris.device.allocate` indicates that a user must have the `solaris.device.allocate` authorization to use the device.

Example 5–4 Permitting Any User to Allocate a Device

In the following example, any user on the system can allocate any device. The fifth field in every device entry in the `device_allocate` file has been changed to an at sign (@).

   $ whoami
   
   `devicesec`

   $ vi /etc/security/device_allocate
   
   `audio;audio;reserved;reserved;@;/etc/security/lib/audio_clean`
Example 5–5 Preventing Some Peripheral Devices From Being Used

In the following example, the audio device cannot be used. The fifth field in the audio device entry in the device_allocate file has been changed to an asterisk (*).

```bash
$ whoami
devicesec
$ vi /etc/security/device_allocate
audio;audio;reserved;reserved;*/etc/security/lib/audio_clean
fd0;fd;reserved;reserved;*/etc/security/lib/fd_clean
sr0;sr;reserved;reserved;*/etc/security/lib/sr_clean
...
```

Example 5–6 Preventing All Peripheral Devices From Being Used

In the following example, no peripheral device can be used. The fifth field in every device entry in the device_allocate file has been changed to an asterisk (*).

```bash
$ whoami
devicesec
$ vi /etc/security/device_allocate
audio;audio;reserved;reserved;*/etc/security/lib/audio_clean
fd0;fd;reserved;reserved;*/etc/security/lib/fd_clean
sr0;sr;reserved;reserved;*/etc/security/lib/sr_clean
...
```

▼ How to Audit Device Allocation

By default, the device allocation commands are in the other audit class.

1 **Assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **Preselect the ot class for auditing.**

   Add the ot class to the flags line of the audit_control file. The file would appear similar to the following:

   ```bash
   # audit_control file
dir:/var/audit
   ```
Allocating Devices

Device allocation reserves the use of a device to one user at a time. Devices that require a mount point must be mounted.

How to Allocate a Device

Before You Begin

Device allocation must be enabled, as described in “How to Make a Device Allocatable” on page 88. If authorization is required, the user must have the authorization.

1 Allocate the device.
Specify the device by device name.

% allocate device-name

2 Verify that the device is allocated.
Run the identical command.

% allocate device-name
allocate. Device already allocated.

Example 5–7 Allocating a Microphone

In this example, the user jdoe allocates a microphone, audio.
Allocating Devices

% whoami
jdoe
% allocate audio

Example 5–8 Allocating a Printer

In this example, a user allocates a printer. No one else can print to printer-1 until the user deallocates it, or until the printer is forcibly allocated to another user.

% allocate /dev/lp/printer-1

For an example of forcible deallocation, see “Forcibly Deallocating a Device” on page 91.

Example 5–9 Allocating a Tape Drive

In this example, the user jdoe allocates a tape drive, st0.

% whoami
jdoe
% allocate st0

Troubleshooting If the allocate command cannot allocate the device, an error message is displayed in the console window. For a list of allocation error messages, see the allocate(1) man page.

▼ How to Mount an Allocated Device

Before You Begin The user or role has allocated the device. To mount a device, the user or role must have the privileges that are required for mounting the device. To give the required privileges, see “How to Authorize Users to Allocate a Device” on page 89.

1 Assume a role that can allocate and mount a device.
   % su - role-name
   Password: <Type role-name password>
   $

2 Create and protect a mount point in the role’s home directory.
   You only need to do this step the first time you need a mount point.
   $ mkdir mount-point ; chmod 700 mount-point

3 List the allocatable devices.
   $ list_devices -l
   List of allocatable devices
4 Allocate the device.

Specify the device by device name.

$ allocate device-name

5 Mount the device.

$ mount -o ro -F filesystem-type device-path mount-point

where

- o ro Indicates that the device is to be mounted read-only. Use -o rw to indicate that you should be able to write to the device.

-F filesystem-type Indicates the file system format of the device. Typically, a CD-ROM is formatted with an HSFS file system. A diskette is typically formatted with a PCFS file system.

device-path Indicates the path to the device. The output of the list_devices -l command includes the device-path.

mount-point Indicates the mount point that you created in Step 2.

Example 5–10 Allocating a Diskette Drive

In this example, a user assumes a role that can allocate and mount a diskette drive, fd0. The diskette is formatted with a PCFS file system.

% roles
devicectl
% su - devicectl
Password: <Type devicectl password>
$ mkdir /home/devicectl/mymnt
$ chmod 700 /home/devicectl/mymnt
$ list_devices -l
...
device: fd0 type: fd files: /dev/diskette /dev/rdiskette /dev/fd0a
...
$ allocate fd0
$ mount -o ro -F pcfs /dev/diskette /home/devicectl/mymnt/mymnt
$ ls /home/devicectl/mymnt/mymnt
List of the contents of diskette

Example 5–11 Allocating a CD-ROM Drive

In this example, a user assumes a role that can allocate and mount a CD-ROM drive, sr0. The drive is formatted as an HSFS file system.
% roles
devicealloc
% su - devicealloc
Password: <Type devicealloc password>
$ mkdir /home/devicealloc/mymnt
$ chmod 700 /home/devicealloc/mymnt
$ list_devices -l
... 
device: sr0 type: sr files: /dev/sr0 /dev/rsr0 /dev/dsk/c0t2d0s0 ...
...
$ allocate sr0
$ mount -o ro -F hsfs /dev/sr0 /home/devicealloc/mymnt
$ cd /home/devicealloc/mymnt ; ls
List of the contents of CD-ROM

Troubleshooting
If the mount command cannot mount the device, an error message is displayed: mount: insufficient privileges. Check the following:

- Make sure that you are executing the mount command in a profile shell. If you have assumed a role, the role has a profile shell. If you are a user who has been assigned a profile with the mount command, you must create a profile shell. The commands pfsh, pfksh, and pfcs create a profile shell.

- Make sure that you own the specified mount point. You should have read, write, and execute access to the mount point.

Contact your administrator if you still cannot mount the allocated device.

▶ How to Deallocate a Device
Deallocation enables other users to allocate and use the device when you are finished.

Before You Begin
You must have allocated the device.

1 If the device is mounted, unmount the device.
   $ cd $HOME
   $ umount mount-point

2 Deallocate the device.
   $ deallocate device-name

Example 5–12 Deallocating a Microphone
In this example, the user jdoe deallocates the microphone, audio.
Deallocating a CD-ROM Drive

In this example, the Device Allocator role deallocates a CD-ROM drive. After the message is printed, the CD-ROM is ejected.

```
$ whoami
devicealloc
$ cd /home/devicealloc
$ umount /home/devicealloc/mymnt
$ ls /home/devicealloc/mymnt
$
$ deallocate sr0
/dev/sr0: 326o
/dev/rsr0: 326o
...
sr_clean: Media in sr0 is ready. Please, label and store safely.
```

Device Protection (Reference)

Devices in the Solaris OS are protected by device policy. Peripheral devices can be protected by device allocation. Device policy is enforced by the kernel. Device allocation is optionally enabled, and is enforced at the user level.

Device Policy Commands

Device management commands administer the device policy on local files. Device policy can include privilege requirements. Only superuser or a role of equivalent capabilities can manage devices.

The following table lists the device management commands.
### Device Management Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
<th>Man Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>devfsadm</td>
<td>Administers devices and device drivers on a running system. Also loads device policy. The devfsadm command enables the cleanup of dangling /dev links to disk, tape, port, audio, and pseudo devices. Devices for a named driver can also be reconfigured.</td>
<td>devfsadm(1M)</td>
</tr>
<tr>
<td>getdevpolicy</td>
<td>Displays the policy associated with one or more devices. This command can be run by any user.</td>
<td>getdevpolicy(1M)</td>
</tr>
<tr>
<td>add_drv</td>
<td>Adds a new device driver to a running system. Contains options to add device policy to the new device. Typically, this command is called in a script when a device driver is being installed.</td>
<td>add_drv(1M)</td>
</tr>
<tr>
<td>update_drv</td>
<td>Updates the attributes of an existing device driver. Contains options to update the device policy for the device. Typically, this command is called in a script when a device driver is being installed.</td>
<td>update_drv(1M)</td>
</tr>
<tr>
<td>rem_drv</td>
<td>Removes a device or device driver.</td>
<td>rem_drv(1M)</td>
</tr>
</tbody>
</table>

### Device Allocation

Device allocation can protect your site from loss of data, computer viruses, and other security breaches. Unlike device policy, device allocation is optional. Devices are not allocatable until the bsmconv script is run. Device allocation uses authorizations to limit access to allocatable devices.

#### Components of Device Allocation

The components of the device allocation mechanism are as follows:

- The allocate, deallocate, dinfo, and list_devices commands. For more information, see “Device Allocation Commands” on page 99.
- Device-clean scripts for each allocatable device.

These commands and scripts use the following local files to implement device allocation:

- The /etc/security/device_allocate file. For more information, see the device_allocate(4) man page.
- The /etc/security/device_maps file. For more information, see the device_maps(4) man page.
- A lock file, in the /etc/security/dev directory, for each allocatable device.
- The changed attributes of the lock files that are associated with each allocatable device.
Note – The /etc/security/dev directory might not be supported in future releases of the Solaris OS.

Device Allocation Commands

With uppercase options, the `allocate`, `deallocate`, and `list_devices` commands are administrative commands. Otherwise, these commands are user commands. The following table lists the device allocation commands.

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
<th>Man Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>bsmconv</td>
<td>Creates databases to handle device allocation. Also enables the auditing service. You must be superuser or in the Primary Administrator role.</td>
<td>bsmconv(1M)</td>
</tr>
<tr>
<td>dminfo</td>
<td>Searches for an allocatable device by device type, by device name, and by full path name.</td>
<td>dminfo(1M)</td>
</tr>
<tr>
<td>list_devices</td>
<td>Lists the status of allocatable devices.</td>
<td>list_devices(1)</td>
</tr>
<tr>
<td>list_devices -U</td>
<td>Lists all the device-special files that are associated with any device that is listed in the device_maps file.</td>
<td></td>
</tr>
<tr>
<td>allocate</td>
<td>Reserves an allocatable device for use by one user.</td>
<td>allocate(1)</td>
</tr>
<tr>
<td></td>
<td>By default, a user must have the <code>solaris.device.allocate</code> authorization to allocate a device. You can modify the device Allocate file to not require user authorization. Then, any user on the system can request the device to be allocated for use.</td>
<td></td>
</tr>
<tr>
<td>deallocate</td>
<td>Removes the allocation reservation from a device.</td>
<td>deallocate(1)</td>
</tr>
</tbody>
</table>

Authorizations for the Allocation Commands

By default, users must have the `solaris.device.allocate` authorization to reserve an allocatable device. To create a rights profile to include the `solaris.device.allocate` authorization, see “How to Authorize Users to Allocate a Device” on page 89.

Administrators must have the `solaris.device.revoke` authorization to change the allocation state of any device. For example, the `-U` option to the allocate and list_devices commands, and the `-F` option to the deallocate command require the `solaris.device.revoke` authorization.
For more information, see “Commands That Require Authorizations” on page 219.

Allocate Error State

A device is put in an allocate error state when the deallocate command fails to deallocate, or when the allocate command fails to allocate. When an allocatable device is in an allocate error state, then the device must be forcibly deallocated. Only superuser or a role with the Device Management rights profile or the Device Security rights profile can handle an allocate error state.

The deallocate command with the -F option forces deallocation. Or, you can use allocate -U to assign the device to a user. Once the device is allocated, you can investigate any error messages that appear. After any problems with the device are corrected, you can forcibly deallocate it.

device_maps File

Device maps are created when you set up device allocation. A default /etc/security/device_maps file is created by the bsmconv command when the auditing service is enabled. This initial device_maps file can be customized for your site. The device_maps file includes the device names, device types, and device-special files that are associated with each allocatable device.

The device_maps file defines the device-special file mappings for each device, which in many cases is not intuitive. This file allows programs to discover which device-special files map to which devices. You can use the dminfo command, for example, to retrieve the device name, the device type, and the device-special files to specify when you set up an allocatable device. The dminfo command uses the device_maps file to report this information.

Each device is represented by a one-line entry of the form:

device-name:device-type:device-list

EXAMPLE 5–14 Sample device_maps Entry

The following is an example of an entry in a device_maps file for a diskette drive, fd0:

```
fd0:\
  fd:\
    /dev/diskette /dev/rdiskette /dev/fd0a /dev/rfd0a \
    /dev/fd0b /dev/rfd0b /dev/fd0c /dev/rfd0c /dev/rfd0:\
```

Lines in the device_maps file can end with a backslash (\) to continue an entry on the next line. Comments can also be included. A pound sign (#) comments all subsequent text until the next newline that is not immediately preceded by a backslash. Leading and trailing blanks are allowed in any field. The fields are defined as follows:
device-name  Specifies the name of the device. For a list of current device names, see "How to View Allocation Information About a Device" on page 90.

device-type  Specifies the generic device type. The generic name is the name for the class of devices, such as st, fd, or audio. The device-type field logically groups related devices.

device-list  Lists the device-special files that are associated with the physical device. The device-list must contain all of the special files that allow access to a particular device. If the list is incomplete, a malevolent user can still obtain or modify private information. Valid entries for the device-list field reflect the device files that are located in the /dev directory.

device_allocate File

An initial /etc/security/device_allocate file is created by the bsmconv command when the auditing service is enabled. This initial device_allocate file can be used as a starting point. You can modify the device_allocate file to change devices from allocatable to non.allocatable, or to add new devices. A sample device_allocate file follows.

st0;st;;;;/etc/security/lib/st_clean
fd0;fd;;;;/etc/security/lib/fd_clean
sr0;sr;;;;/etc/security/lib/sr_clean
audio;audio;;;*;/etc/security/lib/audio_clean

An entry in the device_allocate file does not mean that the device is allocatable, unless the entry specifically states that the device is allocatable. In the sample device_allocate file, note the asterisk (*) in the fifth field of the audio device entry. An asterisk in the fifth field indicates to the system that the device is not allocatable. Therefore, the device cannot be used. Other values or no value in this field indicates that the device can be used.

In the device_allocate file, each device is represented by a one-line entry of the form:

device-name;device-type;reserved;reserved;auths;device-exec

Lines in the device_allocate file can end with a backslash (\) to continue an entry on the next line. Comments can also be included. A pound sign (#) comments all subsequent text until the next newline that is not immediately preceded by a backslash. Leading and trailing blanks are allowed in any field. The fields are defined as follows:

device-name  Specifies the name of the device. For a list of current device names, see "How to View Allocation Information About a Device" on page 90.

device-type  Specifies the generic device type. The generic name is the name for the class of devices, such as st, fd, and sr. The device-type field logically groups related devices. When you make a device allocatable, retrieve the device name from the device-type field in the device_maps file.
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reserved</td>
<td>Sun reserves the two fields that are marked reserved for future use.</td>
</tr>
<tr>
<td>auths</td>
<td>Specifies whether the device is allocatable. An asterisk (*) in this field indicates that the device is not allocatable. An authorization string, or an empty field, indicates that the device is allocatable. For example, the string <code>solaris.device.allocate</code> in the auths field indicates that the <code>solaris.device.allocate</code> authorization is required to allocate the device. An at sign (@) in this file indicates that the device is allocatable by any user.</td>
</tr>
<tr>
<td>device-exec</td>
<td>Supplies the path name of a script to be invoked for special handling, such as cleanup and object-reuse protection during the allocation process. The <code>device-exec</code> script is run any time that the device is acted on by the <code>deallocate</code> command. For example, the following entry for the sr0 device indicates that the CD-ROM drive is allocatable by a user with the <code>solaris.device.allocate</code> authorization:</td>
</tr>
</tbody>
</table>

```
sr0;sr;reserved;reserved;solaris.device.allocate;/etc/security/lib/sr_clean
```

You can decide to accept the default devices and their defined characteristics. After you install a new device, you can modify the entries. Any device that needs to be allocated before use must be defined in the `device_allocate` and `device_maps` files for that device's system. Currently, cartridge tape drives, diskette drives, CD-ROM drives, and audio chips are considered allocatable. These device types have device-clean scripts.

**Note** – Xylogics™ tape drives or Archive tape drives also use the `st_clean` script that is supplied for SCSI devices. You need to create your own device-clean scripts for other devices, such as modems, terminals, graphics tablets, and other allocatable devices. The script must fulfill object-reuse requirements for that type of device.

---

### Device-Clean Scripts

Device allocation satisfies part of what is called the object reuse requirement. The `device-clean` scripts address the security requirement that all usable data be purged from a physical device before reuse. The data is cleared before the device is allocatable by another user. By default, cartridge tape drives, diskette drives, CD-ROM drives, and audio devices require device-clean scripts. The Solaris OS provides the scripts. This section describes what device-clean scripts do.

**Device-Clean Script for Tapes**

The `st_clean` device-clean script supports three tape devices:

- SCSI ¼-inch tape
- Archive ¼-inch tape
- Open-reel ½-inch tape
The st_clean script uses the rewoffl option to the mt command to clean up the device. For more information, see the mt(1) man page. If the script runs during system boot, the script queries the device to determine if the device is online. If the device is online, the script determines if the device has media in it. The ¼-inch tape devices that have media in them are placed in the allocate error state. The allocate error state forces the administrator to manually clean up the device.

During normal system operation, when the deallocate command is executed in interactive mode, the user is prompted to remove the media. Deallocation is delayed until the media is removed from the device.

**Device-Clean Scripts for Diskettes and CD-ROM Drives**

The following device-cleanscripts are provided for diskettes and CD-ROM drives:

- **fd_clean** script – Is a device-cleanscript for diskettes.
- **sr_clean** script – Is a device-cleanscript for CD-ROM drives.

The scripts use the eject command to remove the media from the drive. If the eject command fails, the device is placed in the allocate error state. For more information, see the eject(1) man page.

**Device-Clean Script for Audio**

Audio devices are cleaned up with an audio_clean script. The script performs an AUDIO_GETINFO ioctl system call to read the device. The script then performs an AUDIO_SETINFO ioctl system call to reset the device configuration to the default.

**Writing New Device-Clean Scripts**

If you add more allocatable devices to the system, you might need to create your own device-cleanscripts. The deallocate command passes a parameter to the device-cleanscripts. The parameter, which is shown here, is a string that contains the device name. For more information, see the device_allocate(4) man page.

```
clean-script -[I|i|f|S] device-name
```

Device-cleanscripts must return "0" for success and greater than "0" for failure. The options -I, -f, and -S determine the running mode of the script:

- **-I** Is needed during system boot only. All output must go to the system console. Failure or inability to forcibly eject the media must put the device in the allocate error state.
- **-i** Similar to the -I option, except that output is suppressed.
- f  Is for forced cleanup. The option is interactive and assumes that the user is available to respond to prompts. A script with this option must attempt to complete the cleanup if one part of the cleanup fails.

- S  Is for standard cleanup. The option is interactive and assumes that the user is available to respond to prompts.
Using the Basic Audit Reporting Tool (Tasks)

This chapter describes how to create a manifest of the files on a system and how to use that manifest to check the integrity of the system. The Basic Audit Reporting Tool (BART) enables you to comprehensively validate systems by performing file-level checks of a system over time.

The following is a list of the information in this chapter:
- “Using BART (Task Map)” on page 108
- “Basic Audit Reporting Tool (Overview)” on page 105
- “Using BART (Tasks)” on page 109
- “BART Manifest, Rules File, and Reporting (Reference)” on page 122

Basic Audit Reporting Tool (Overview)

BART is a file tracking tool that operates entirely at the file system level. Using BART gives you the ability to quickly, easily, and reliably gather information about the components of the software stack that is installed on deployed systems. Using BART can greatly reduce the costs of administering a network of systems by simplifying time-consuming administrative tasks.

BART enables you to determine what file-level changes have occurred on a system, relative to a known baseline. You use BART to create a baseline or control manifest from a fully installed and configured system. You can then compare this baseline with a snapshot of the system at a later time, generating a report that lists file-level changes that have occurred on the system since it was installed.

The bart command is a standard UNIX command. You can redirect the output of the bart command to a file for later processing.
BART Features

BART has been designed with an emphasis on a simple syntax that is both powerful and flexible. The tool enables you to generate manifests of a given system over time. Then, when the system’s files need to be validated, you can generate a report by comparing the old and new manifests. Another way to use BART is to generate manifests of several similar systems and run system-to-system comparisons. The main difference between BART and existing auditing tools is that BART is flexible, both in terms of what information is tracked and what information is reported.

Additional benefits and uses of BART include the following:

- Provides an efficient and easy method for cataloging a system that is running the Solaris software at the file level.
- Enables you to define which files to monitor and gives you the ability to modify profiles when necessary. This flexibility allows you to monitor local customizations and enables you to reconfigure software easily and efficiently.
- Ensures that systems are running reliable software.
- Allows you to monitor file-level changes of a system over time, which can help you locate corrupted or unusual files.
- Helps you troubleshoot system performance issues.

BART Components

BART has two main components and one optional component:

- BART Manifest
- BART Report
- BART Rules File

BART Manifest

You use the bart create command to take a file-level snapshot of a system at a particular time. The output is a catalog of files and file attributes called a manifest. The manifest lists information about all the files or specific files on a system. It contains information about attributes of files, which can include some uniquely identifying information, such as an MD5 checksum. For more information about the MD5 checksum, see the md5(3EXT) man page. A manifest can be stored and transferred between client and server systems.
Note – BART does not cross file system boundaries, with the exception of file systems of the same type. This constraint makes the output of the bart create command more predictable. For example, without arguments, the bart create command catalogs all ZFS file systems under the root (/) directory. However, no NFS or TMPFS file systems or mounted CD-ROMs would be cataloged. When creating a manifest, do not attempt to audit file systems on a network. Note that using BART to monitor networked file systems can consume large resources to generate manifests that will have little value.

For more information about BART manifests, see “BART Manifest File Format” on page 123.

BART Report

The report tool has three inputs: the two manifests to be compared and an optional user-provided rules file that indicates which discrepancies are to be flagged.

You use the bart compare command to compare two manifests, a control manifest and a test manifest. These manifests must be prepared with the same file systems, options, and rules file that you use with the bart create command.

The output of the bart compare command is a report that lists per-file discrepancies between the two manifests. A discrepancy is a change to any attribute for a given file that is cataloged for both manifests. Additions or deletions of file entries between the two manifests are also considered discrepancies.

There are two levels of control when reporting discrepancies:

■ When generating a manifest
■ When producing reports

These levels of control are intentional, since generating a manifest is more costly than reporting discrepancies between two manifests. Once you have created manifests, you have the ability to compare manifests from different perspectives by running the bart compare command with different rules files.

For more information about BART reports, see “BART Reporting” on page 125.

BART Rules File

The rules file is a text file that you can optionally use as input to the bart command. This file uses inclusion and exclusion rules. A rules file is used to create custom manifests and reports. A rules file enables you to express in a concise syntax which sets of files you want to catalog, as well as which attributes to monitor for any given set of files. When you compare manifests, the rules file aids in flagging discrepancies between the manifests. Using a rules file is an effective way to gather specific information about files on a system.
You create a rules file by using a text editor. With a rules file, you can perform the following tasks:

- Use the `bart create` command to create a manifest that lists information about all or specific files on a system.
- Use the `bart compare` command to generate a report that monitors specific attributes of a file system.

**Note** – You can create several rules files for different purposes. However, if you create a manifest by using a rules file, you must use the same rules file when you compare the manifests. If you do not use the same rules file when comparing manifests that were created with a rules file, the output of the `bart compare` command will list many invalid discrepancies.

A rules file can also contain syntax errors and other ambiguous information as a result of user error. If a rules file does contain misinformation, these errors will also be reported.

Using a rules file to monitor specific files and file attributes on a system requires planning. Before you create a rules file, decide which files and file attributes on the system you want to monitor. Depending on what you are trying to accomplish, you might use a rules file to create manifests, compare manifests, or for purposes.

For more information about the BART rules file, see “BART Rules File Format” on page 124 and the `bart_rules(4)` man page.

### Using BART (Task Map)

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a manifest.</td>
<td>Obtain a manifest that lists information about every file that is installed on a system.</td>
<td>“How to Create a Manifest” on page 109</td>
</tr>
</tbody>
</table>
| Create a custom manifest. | Obtain a manifest that lists information about specific files that are installed on a system in one of the following ways:  
  - By specifying a subtree  
  - By specifying a file name  
  - By using a rules file | “How to Customize a Manifest” on page 112 |
| Compare manifests for the same system over time. Or, compare manifests for different systems with a control system manifest. | Obtain a report that compares changes to a system over time. Or, obtain a report that compares one or several systems to control system. | “How to Compare Manifests for the Same System Over Time” on page 115  
“How to Compare Manifests From a Different System With the Manifest of a Control System” on page 117 |
Using BART (Tasks)

You can run the `bart` command as a regular user, superuser, or a user who has assumed the Primary Administrator role. If you run the `bart` command as a regular user, you will only be able to catalog and monitor files that you have permission to access, for example, information about files in your home directory. The advantage of becoming superuser when you run the `bart` command is that the manifests you create will contain information about hidden and private files that you might want to monitor. If you need to catalog and monitor information about files that have restricted permissions, for example, the `/etc/passwd` or `/etc/shadow` file, run the `bart` command as superuser or assume an equivalent role. For more information about using role-based access control, see “Configuring RBAC (Task Map)” on page 174.

BART Security Considerations

Running the `bart` command as superuser makes the output readable by anyone. This output might contain file names that are intended to be private. If you become superuser when you run the `bart` command, take appropriate measures to protect the output. For example, use options that generate output files with restrictive permissions.

Note – The procedures and examples in this chapter show the `bart` command run by superuser. Unless otherwise specified, running the `bart` command as superuser is optional.

▼ How to Create a Manifest

You can create a manifest of a system immediately after an initial Solaris software installation. This type of manifest will provide you with a baseline for comparing changes to the same system over time. Or, you can use this manifest to compare with the manifests for different systems. For example, if you take a snapshot of each system on your network, and then compare each test manifest with the control manifest, you can quickly determine what you need to do to synchronize the test system with the baseline configuration.
1 Assume the Primary Administrator role, or become superuser.
   The Primary Administrator role includes the Primary Administrator profile. To create the role
   and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console
   (Tasks)," in System Administration Guide: Basic Administration.

2 After installing the Solaris software, create a control manifest and redirect the output to a file.
   # bart create options > control-manifest
   
   -R Specifies the root directory for the manifest. All paths specified by the rules will be
     interpreted relative to this directory. All paths reported in the manifest will be relative to
     this directory.
   
   -I Accepts a list of individual files to be cataloged, either on the command line or read from
     standard input.
   
   -r Is the name of the rules file for this manifest. Note that –, when used with the -r option,
     will be read the rules file from standard input.
   
   -n Turns off content signatures for all regular files in the file list. This option can be used to
     improve performance. Or, you can use this option if the contents of the file list are
     expected to change, as in the case of system log files.

3 Examine the contents of the manifest.

4 Save the manifest for future use.
   Choose a meaningful name for the manifest. For example, use the system name and date that
   the manifest was created.

Example 6–1 Creating a Manifest That Lists Information About Every File on a System

If you run the bart create command without any options, information about every file that is
installed on the system will be cataloged. Use this type of manifest as a baseline when you are
installing many systems from a central image. Or, use this type of manifest to run comparisons
when you want to ensure that the installations are identical.

For example:

   # bart create
   ! Version 1.0
   ! Thursday, December 04, 2003 (16:17:39)
   # Format:
   #fname D size mode acl dirmtime uid gid
   #fname P size mode acl mttime uid gid
   #fname S size mode acl mtime uid gid
   #fname F size mode acl mtime uid gid contents
   #fname L size mode acl lnmtime uid gid dest

110 System Administration Guide: Security Services • January 2010 (Beta)
Each manifest consists of a header and entries. Each manifest file entry is a single line, depending on the file type. For example, for each manifest entry in the preceding output, type F specifies a file and type D specifies a directory. Also listed is information about size, content, user ID, group ID, and permissions. File entries in the output are sorted by the encoded versions of the file names to correctly handle special characters. All entries are sorted in ascending order by filename. All nonstandard file names, such as those that contain embedded newline or tab characters, have the nonstandard characters quoted before being sorted.

Lines that begin with ! supply metadata about the manifest. The manifest version line indicates the manifest specification version. The date line shows the date on which the manifest was created, in date form. See the date(1) man page. Some lines are ignored by the manifest comparison tool. Ignored lines include blank lines, lines that consist only of white space, and comments that begin with #.
How to Customize a Manifest

You can customize a manifest in one of the following ways:

- **By specifying a subtree**
  Creating a manifest for an individual subtree on a system is an efficient way to monitor changes to specific files, rather than the entire contents of a large directory. You can create a baseline manifest of a specific subtree on your system, then periodically create test manifests of the same subtree. Use the `bart compare` command to compare the control manifest with the test manifest. By using this option, you are able to efficiently monitor important file systems to determine whether any files have been compromised by an intruder.

- **By specifying a file name**
  Since creating a manifest that catalogs the entire system is more time-consuming, takes up more space, and is more costly, you might choose to use this option of the `bart` command when you want to only list information about a specific file or files on a system.

- **By using a rules file**
  You use a rules file to create custom manifests that list information about specific files and specific subtrees on a given system. You can also use a rules file to monitor specific file attributes. Using a rules file to create and compare manifests gives you the flexibility to specify multiple attributes for more than one file or subtree. Whereas, from the command line, you can only specify a global attribute definition that applies to all files for each manifest you create or report you generate.

1. **Determine which files you want to catalog and monitor.**
2. **Assume the Primary Administrator role, or become superuser.**
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.
3. **After installing the Solaris software, create a custom manifest by using one of the following options:**
   - **By specifying a subtree:**
     # bart create -R root-directory
   - **By specifying a file name or file names:**
     # bart create -I filename...
     For example:
     # bart create -I /etc/system /etc/passwd /etc/shadow
   - **By using a rules file:**
# bart create -r rules-file

4 Examine the contents of the manifest.

5 Save the manifest for future use.

Example 6–2 Creating a Manifest by Specifying a Subtree

This example shows how to create a manifest that contains information about the files in the /etc/ssh subtree only.

# bart create -R /etc/ssh
! Version 1.0
! Saturday, November 29, 2003 (14:05:36)
# Format:
# fname D size mode acl dirmtime uid gid
# fname P size mode acl mtime uid gid
# fname S size mode acl mtime uid gid
# fname F size mode acl mtime uid gid contents
# fname L size mode acl lmtime uid gid dest
# fname B size mode acl mtime uid gid devnode
# fname C size mode acl mtime uid gid devnode
/ D 512 40755 user::rwx,group::r-x,other::r-x 3f81eab9 0 3
/ssh_config F 861 100644 user::rw-,group::r--,
other::r-- 3f81e504 0 4 222453ca0e2348cd9981820935600395
/ssh_host_dsa_key F 668 100600 user::rwx,group::r--,
other::r-- 3f81eab9 0 0 5cc28ecd97e33069fd41ef8e4d9834
/ssh_host_dsa_key.pub F 602 100644 user::rwx,group::r--,
other::r-- 3f81eab9 0 0 16118c736995a4e475555ab4f28c917
/ssh_host_rsa_key F 883 100600 user::rwx,group::r--,
other::r-- 3f81eab9 0 0 6ff002a986ebd20321c448cb9a88409
/ssh_host_rsa_key.pub F 222 100644 user::rwx,group::r--,
other::r-- 3f81eab9 0 0 9ea27617efc76058cb97aa2ca6dd65a
.
.
.

Example 6–3 Customizing a Manifest by Specifying a File Name

This example shows how to create a manifest that lists only information about the /etc/passwd and /etc/shadow files on a system.

# bart create -I /etc/passwd /etc/shadow
! Version 1.0
# Format:
Example 6–4 Customizing a Manifest by Using a Rules File

This example shows how to create a manifest by using a rules file to catalog only those files in the /etc directory. The same rules file includes directives to be used by the bart compare command for monitoring changes to the acl attribute of the /etc/system file.

- Use a text editor to create a rules file that catalogs only those files in the /etc directory.
  
  # List information about all the files in the /etc directory.
  
  CHECK all
  /etc

  # Check only acl changes in the /etc/system file

  IGNORE all
  CHECK acl
  /etc/system

  For more information about creating a rules file, see “BART Rules File” on page 107.

- Create a control manifest by using the rules file you created.

  # bart create -r etc.rules-file > etc.system.control-manifest
  ! Version 1.0
Create a test manifest whenever you want to monitor changes to the system. Prepare the test manifest identically to the control manifest by using the same `bart` options and the same rules file.

- Compare manifests by using the same rules file.

**How to Compare Manifests for the Same System Over Time**

Use this procedure when you want to monitor file-level changes to the same system over time. This type of manifest can assist you in locating corrupted or unusual files, detecting security breaches, or in troubleshooting performance issues on a system.

1. **Assume the Primary Administrator role, or become superuser.**
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2. **After installing the Solaris software, create a control manifest of the files that you want to monitor on the system.**
   
   ```
   # bart create -R /etc > control-manifest
   ```

3. **Create a test manifest that is prepared identically to the control manifest whenever you want to monitor changes to the system.**

   ```
   # bart create -R /etc > test-manifest
   ```

4. **Compare the control manifest with the test manifest.**

   ```
   # bart compare options control-manifest test-manifest > bart-report
   ```

   `-r` is the name of the rules file for this comparison. Using the `-r` option with the `–` means that the directives will be read from standard input.
Using BART (Tasks)

- **i** Allows the user to set global IGNORE directives from the command line.
- **p** Is the programmatic mode that generates standard non-localized output for programmatic parsing.

**control-manifest** Is the output from the bart create command for the control system.

**test-manifest** Is the output from the bart create command of the test system.

5 Examine the BART report for oddities.

**Example 6–5  Comparing Manifests for the Same System Over Time**

This example shows how to monitor changes that have occurred in the /etc directory between two points in time. This type of comparison enables you to quickly determine whether important files on the system have been compromised.

- Create a control manifest.

  ```
  # bart create -R /etc > system1.control.121203
  ! Version 1.0
  ! Friday, December 12, 2003 (08:34:51)
  # Format:
  # fname D size mode acl dirmtime uid gid
  # fname P size mode acl mtime uid gid
  # fname S size mode acl mtime uid gid
  # fname F size mode acl mtime uid gid contents
  # fname L size mode acl lmtime uid gid dest
  # fname B size mode acl mtime uid gid devnode
  # fname C size mode acl mtime uid gid devnode
  / D 4096 40755 user::rwx,group::r-x,mask:r-x,other:r-x 3fd9dfb4 0 3
  `/cpr.config F 2236 100644 user::rw-,group::r--,mask:---,other:--- 3fd9991f 0 0
  67cfa2c830b4ce3e112f38c5e33c56a2
  /group.lock F 0 100600 user::rw-,group::----,mask:---,other:--- 3f81f14d 0 1 d41
  d8cd98f0b204e9800998ecf8427e
  /java D 512 40755 user::rw-,group::r-x,mask:r-x,other:r-x 3f81dcb5 0 2
  /java/.systemPrefs D 512 40755 user::rw-,group::r-x,mask:r-x,
  other:r-x 3f81dcb7
  .
  .
  .
  ```

- Create a test manifest when you want to monitor changes to the /etc directory.

  ```
  # bart create -R /etc > system1.test.121503
  Version 1.0
  ! Monday, December 15, 2003 (08:35:28)
  ```
# Format:
# fname D size mode acl dirmtime uid gid
# fname P size mode acl mtime uid gid
# fname S size mode acl mtime uid gid
# fname F size mode acl mtime uid gid contents
# fname L size mode acl lmtime uid gid dest
# fname B size mode acl mtime uid gid devnode
# fname C size mode acl mtime uid gid devnode

/D 4096 40755 user::rwx,group::r-x,mask:r-x,other:r-x 3fd9dfb4 0 3

/.cpr_config F 2236 100644 user::rw-,group::r--,mask:r--,other:r--
3fd9991f 0 0 67cfa2c830b4ce3e112f38c5e33c56a2

/.group.lock F 0 100600 user::rw-,group::r--,mask::r--,other::r--
3f81f14d 0 1 d41d8cd98f00b204e9800998ef8427e

/.java D 512 40755 user::rw-,group::r-x,mask::r-x,other::r-x 3f81dcb5 0 2

/.java/.systemPrefs D 512 40755 user::rw-,group::r-x,mask::r-x,
other::r-x 3f81dcb70 2

/.java/.systemPrefs/.systemRootModFile F 0 100644 user::rw-,group::r--
,mask::r-- other:
3f81dcb5 0 2 d41d8cd98f00b204e9800998ef8427e

/.java/.systemPrefs/.system.lock F 0 100644 user::rw-,group::r--
,mask::r-- other:
3f81dcb5 0 2 d41d8cd98f00b204e9800998ef8427e

Compare the control manifest with the test manifest.

# bart compare system1.control.121203 system1.test.121503

/vfstab:
  mode control:100644 test:100777
  acl control:user::rw-,group::r--,mask::r--,other::r-- test:user::rwx,
group::rwx,mask::rwx,other::rwx

The preceding output indicates permissions on the vfstab file have changed since the control manifest was created. This report can be used to investigate whether ownership, date, content, or any other file attributes have changed. Having this type of information readily available can assist you in tracking down who might have tampered with the file and when the change might have occurred.

### How to Compare Manifests From a Different System

With the Manifest of a Control System

You can run system to system comparisons, thereby enabling you to quickly determine whether there are any file-level differences between a baseline system and the other systems. For
example, if you have installed a particular version of the Solaris software on a baseline system, and you want to know whether other systems have identical packages installed, you can create manifests for those systems and then compare the test manifests with the control manifest. This type of comparison will list any discrepancies in the file contents for each test system that you compare with the control system.

1. **Assume the Primary Administrator role, or become superuser.**
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in System Administration Guide: Basic Administration.

2. **After installing the Solaris software, create a control manifest.**
   
   ```bash
   # bart create options > control-manifest
   ```

3. **Save the control manifest.**

4. **On the test system, use the same bart options to create a manifest, and redirect the output to a file.**
   
   ```bash
   # bart create options > test1-manifest
   ```
   
   Choose a distinct and meaningful name for the test manifest.

5. **Save the test manifest to a central location on the system until you are ready to compare manifests.**

6. **When you want to compare manifests, copy the control manifest to the location of the test manifest. Or, copy the test manifest to the control system.**
   
   For example:
   
   ```bash
   # cp control-manifest /net/test-server/bart/manifests
   ```
   
   If the test system is not an NFS-mounted system, use FTP or some other reliable means to copy the control manifest to the test system.

7. **Compare the control manifest with the test manifest and redirect the output to a file.**
   
   ```bash
   # bart compare control-manifest test1-manifest > test1.report
   ```

8. **Examine the BART report for oddities.**

9. **Repeat Step 4 through Step 9 for each test manifest that you want to compare with the control manifest.**

   Use the same bart options for each test system.
Comparing Manifests From Different Systems With the Manifest of a Control System

This example describes how to monitor changes to the contents of the `/usr/bin` directory by comparing a control manifest with a test manifest from a different system.

- Create a control manifest.

```bash
# bart create -R /usr/bin > control-manifest.121203
! Version 1.0
! Friday, December 12, 2003 (09:19:00)
! Format:
! fname D size mode acl dirmtime uid gid
! fname P size mode acl mtime uid gid
! fname S size mode acl mtime uid gid
! fname F size mode acl mtime uid gid contents
! fname L size mode acl lnmtime uid gid dest
! fname B size mode acl mtime uid gid devnode
! fname C size mode acl mtime uid gid devnode
/D 13312 40755 user::rwx,group::r-x,mask:r-x,other:r-x 3fd9e925 0 2
/.s F 14200 104711 user::rwx,group::--x,mask:--x,other:--x
3f8dbfd6 0 1 8ec7e52d8a35ba3b054a6394cbf71cf6
/ControlPanel L 28 120777 - 3f81dc71 0 1 jre/bin/ControlPanel
/HtmlConverter L 25 120777 - 3f81dcdc 0 1 bin/HtmlConverter
/acctcom F 28300 100555 user::r-x,group::r-x,mask:r-x,other:r-x
3f6b5750 0 2 d6e99b19c847ab4ec084d9088c7c7608
/activation-client F 9172 100755 user::rwx,group::r-x,mask:r-x,other:r-x
other:r-x 3f5cb907 0 1 b3836ad1a656324a6elb01edcba28f0
/adb F 9712 100555 user::r-x,group::r-x,mask:r-x,other:r-x
3f6b576c 0 2 se26413175f65f239ee628a8870eda
/addbib F 11800 100555 user::r-x,group::r-x,mask:r-x,other:r-x
3f6b5803 0 2 a350836c36049feb185f78350f27510
.
.
.
```

- Create a test manifest for each system that you want to compare with the control system.

```bash
# bart create -R /usr/bin > system2-manifest.121503
! Version 1.0
! Friday, December 15, 2003 (13:30:58)
! Format:
! fname D size mode acl dirmtime uid gid
! fname P size mode acl mtime uid gid
! fname S size mode acl mtime uid gid
! fname F size mode acl mtime uid gid contents
! fname L size mode acl lnmtime uid gid dest
! fname B size mode acl mtime uid gid devnode
! fname C size mode acl mtime uid gid devnode
/D 13312 40755 user::rwx,group::r-x,mask:r-x,other:r-x 3fd9ea9c 0 2
```

Example 6–6

Using BART (Tasks)
When you want to compare manifests, copy the manifests to the same location.

```
# cp control-manifest /net/system2.central/bart/manifests
```

Compare the control manifest with the test manifest.

```
# bart compare control-manifest system2.test > system2.report
```

The previous output indicates that the group ID of the su file in the /usr/bin directory is not the same as that of the control system. This information can be helpful in determining whether a different version of the software was installed on the test system or if possibly someone has tampered with the file.

**How to Customize a BART Report by Specifying File Attributes**

This procedure is optional and explains how to customize a BART report by specifying file attributes from the command line. If you create a baseline manifest that lists information about all the files or specific on your system, you can run the bart compare command, specifying different attributes, whenever you need to monitor changes to a particular directory, subdirectory, file or files. You can run different types of comparisons for the same manifests by specifying different file attributes from the command line.

1. **Determine which file attributes you want to monitor.**
2. **Assume the Primary Administrator role, or become superuser.**
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in System Administration Guide: Basic Administration.
3. **After installing the Solaris software, create a control manifest.**
4 Create a test manifest when you want to monitor changes.
Prepare the test manifest identically to the control manifest.

5 Compare the manifests.
For example:

```
# bart compare -i dirmtime,lnmtime,mtime control-manifest.121503 \test-manifest.010504 > bart.report.010504
```

Note that a comma separates each attribute you specify in the command-line syntax.

6 Examine the BART report for oddities.

▼ How to Customize a BART Report by Using a Rules File

This procedure is also optional and explains how to customize a BART report by using a rules file as input to the `bart compare` command. By using a rules file, you can customize a BART report, which allows you the flexibility of specifying multiple attributes for more than one file or subtree. You can run different comparisons for the same manifests by using different rules files.

1 Assume the Primary Administrator role, or become superuser.
The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2 Determine which files and file attributes you want to monitor.

3 Use a text editor to create a rules file with the appropriate directives.

4 After installing the Solaris software, create a control manifest by using the rules file you created.

```
# bart create -r rules-file > control-manifest
```

5 Create a test manifest that is prepared identically to the control manifest.

```
# bart create -r rules-file > test-manifest
```

6 Compare the control manifest with the test manifest by using the same rules file.

```
# bart compare -r rules-file control-manifest test-manifest > bart.report
```

7 Examine the BART report for oddities.
Example 6–7  Customizing a BART Report by Using a Rules File

The following rules file includes directives for both the bart create and the bart compare commands. The rules file directs the bart create command to list information about the contents of the /usr/bin directory. In addition, the rules file directs the bart compare command to track only size and content changes in the same directory.

```bash
# Check size and content changes in the /usr/bin directory.
# This rules file only checks size and content changes.
# See rules file example.

IGNORE all
CHECK size contents
/usr/bin

- Create a control manifest by using the rules file you created.
  
  # bart create -r bartrules.txt > usr_bin.control-manifest.121003

- Create a test manifest whenever you want to monitor changes to the /usr/bin directory.
  
  # bart create -r bartrules.txt > usr_bin.test-manifest.121103

- Compare the manifests by using the same rules file.
  
  # bart compare -r bartrules.txt usr_bin.control-manifest 
  usr_bin.test-manifest

- Examine the output of the bart compare command.
  
  /usr/bin/gunzip: add
  /usr/bin/ypcat:
      delete

In the preceding output, the bart compare command reported a discrepancy in the /usr/bin directory. This output indicates that /usr/bin/ypcat file was deleted, and the /usr/bin/gunzip file was added.

BART Manifest, Rules File, and Reporting (Reference)

This section includes the following reference information:

- “BART Manifest File Format” on page 123
- “BART Rules File Format” on page 124
- “BART Reporting” on page 125
BART Manifest File Format

Each manifest file entry is a single line, depending on the file type. Each entry begins with
fname, which is the name of the file. To prevent parsing problems that are caused by special
characters embedded in file names, the file names are encoded. For more information, see
“BART Rules File Format” on page 124.

Subsequent fields represent the following file attributes:

- **type**: Type of file with the following possible values:
  - B for a block device node
  - C for a character device node
  - D for a directory
  - F for a file
  - L for a symbolic link
  - P for a pipe
  - S for a socket

- **size**: File size in bytes.

- **mode**: Octal number that represents the permissions of the file.

- **acl**:ACL attributes for the file. For a file with ACL attributes, this contains the output
  from acltotext().

- **uid**: Numerical user ID of the owner of this entry.

- **gid**: Numerical group ID of the owner of this entry.

- **dirmtime**: Last modification time, in seconds, since 00:00:00 UTC, January 1, 1970, for
directories.

- **lnmtime**: Last modification time, in seconds, since 00:00:00 UTC, January 1, 1970, for links.

- **mtime**: Last modification time, in seconds, since 00:00:00 UTC January 1, 1970, for files.

- **contents**: Checksum value of the file. This attribute is only specified for regular files. If you
  turn off context checking, or if checksums cannot be computed, the value of this
  field is --.

- **dest**: Destination of a symbolic link.

- **devnode**: Value of the device node. This attribute is for character device files and block
device files only.

For more information about BART manifests, see the bart_manifest(4) man page.
BART Rules File Format

The input files to the bart command are text files. These files consist of lines that specify which files are to be included in the manifest and which file attributes are to be included in the report. The same input file can be used across both pieces of BART functionality. Lines that begin with #, blank lines, and lines that contain white space are ignored by the tool.

The input files have three types of directives:

- Subtree directive, with optional pattern matching modifiers
- CHECK directive
- IGNORE directive

EXAMPLE 6-8 Rules File Format

```
<Global CHECK/IGNORE Directives>
<subtree1> [pattern1..]
<IGNORE/CHECK Directives for subtree1>

<subtree2> [pattern2..]
<subtree3> [pattern3..]
<subtree4> [pattern4..]
<IGNORE/CHECK Directives for subtree2, subtree3, subtree4>
```

Note – All directives are read in order, with later directives possibly overriding earlier directives.

There is one subtree directive per line. The directive must begin with an absolute pathname, followed by zero or more pattern matching statements.

Rules File Attributes

The bart command uses CHECK and IGNORE statements to define which attributes to track or ignore. Each attribute has an associated keyword.

The attribute keywords are as follows:

- acl
- all
- contents
- dest
- devnode
- dirmtime
- gid
- lnmtime
The all keyword refers to all file attributes.

**Quoting Syntax**
The rules file specification language that BART uses is the standard UNIX quoting syntax for representing nonstandard file names. Embedded tab, space, newline, or special characters are encoded in their octal forms to enable the tool to read file names. This nonuniform quoting syntax prevents certain file names, such as those containing an embedded carriage return, from being processed correctly in a command pipeline. The rules specification language allows the expression of complex file name filtering criteria that would be difficult and inefficient to describe by using shell syntax alone.

For more information about the BART rules file or the quoting syntax used by BART, see the *bart_rules*(4) man page.

**BART Reporting**
In default mode, the *bart compare* command, as shown in the following example, will check all the files installed on the system, with the exception of modified directory timestamps (dirmtime):

```
CHECK all
IGNORE dirmtime
```

If you supply a rules file, then the global directives of CHECK all and IGNORE dirmtime, in that order, are automatically prepended to the rules file.

**BART Output**
The following exit values are returned:

<table>
<thead>
<tr>
<th>Exit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td>Nonfatal error when processing files, such as permission problems</td>
</tr>
<tr>
<td>&gt;1</td>
<td>Fatal error, such as an invalid command-line option</td>
</tr>
</tbody>
</table>
The reporting mechanism provides two types of output: verbose and programmatic:

- Verbose output is the default output and is localized and presented on multiple lines. Verbose output is internationalized and is human-readable. When the bart compare command compares two system manifests, a list of file differences is generated. For example:

  ```
  filename attribute control:xxxx test:yyyy
  filename Name of the file that differs between the control manifest and the test manifest.
  attribute Name of the file attribute that differs between the manifests that are compared.
  xxxx is the attribute value from the control manifest, and yyyy is the attribute value from the test manifest. When discrepancies for multiple attributes occur in the same file, each difference is noted on a separate line.
  ```

  Following is an example of the default output for the bart compare command. The attribute differences are for the /etc/passwd file. The output indicates that the size, mtime, and contents attributes have changed.

  ```
  /etc/passwd:
  size control:74 test:81
  mtime control:3c165879 test:3c165979
  contents control:daca28ae8de97af7da6b91fde8d57afa
                           test:84b2b32c416587355317207b48a6ec7
  ```

- Programmatic output is generated if you use the `-p` option when you run the bart compare command. This output is generated in a form that is suitable for programmatic manipulation. Programmatic output can be easily parsed by other programs and is designed to be used as input for other tools. For example:

  ```
  filename attribute control-val test-val [attribute control-val test-val]*
  filename Same as the filename attribute in the default format
  attribute control-val test-val A description of the file attributes that differ between the control and test manifests for each file
  ```

  For a list of attributes that are supported by the bart command, see "Rules File Attributes" on page 124.

  For more information about BART, see the bart(1M) man page.
Chapter 7

Controlling Access to Files (Tasks)

This chapter describes how to protect files in the Solaris Operating System (Solaris OS). The chapter also describes how to protect against files whose permissions could compromise the system.

The following is a list of the information in this chapter.

- “Using UNIX Permissions to Protect Files” on page 127
- “Using Access Control Lists to Protect Files” on page 133
- “Preventing Executable Files From Compromising Security” on page 136
- “Protecting Files (Task Map)” on page 137
- “Protecting Files With UNIX Permissions (Task Map)” on page 137
- “Protecting Files With ACLs (Task Map)” on page 143
- “Protecting Against Programs With Security Risk (Task Map)” on page 149

Using UNIX Permissions to Protect Files

Files can be secured through UNIX file permissions and through ACLs. Files with sticky bits, and files that are executable, require special security measures.

Commands for Viewing and Securing Files

This table describes the commands for monitoring and securing files and directories.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Man Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ls</td>
<td>Lists the files in a directory and information about the files.</td>
<td>ls(1)</td>
</tr>
<tr>
<td>chown</td>
<td>Changes the ownership of a file.</td>
<td>chown(1)</td>
</tr>
</tbody>
</table>
### File and Directory Ownership

Traditional UNIX file permissions can assign ownership to three classes of users:

- **user** – The file or directory owner, which is usually the user who created the file. The owner of a file can decide who has the right to read the file, to write to the file (make changes to it), or, if the file is a command, to execute the file.
- **group** – Members of a group of users.
- **others** – All other users who are not the file owner and are not members of the group.

The owner of the file can usually assign or modify file permissions. Additionally, users or roles with administrative capabilities, such as superuser or the Primary Administrator role, can change a file's ownership. To override system policy, see Example 7–2.

A file can be one of seven types. Each type is displayed by a symbol:

- **-** (Minus symbol)  Text or program  
- **b**  Block special file  
- **c**  Character special file  
- **d**  Directory  
- **l**  Symbolic link  
- **s**  Socket  
- **D**  Door  
- **P**  Named pipe (FIFO)

### UNIX File Permissions

The following table lists and describes the permissions that you can give to each class of user for a file or directory.
### TABLE 7–2 File and Directory Permissions

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Permission</th>
<th>Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>Read</td>
<td>File</td>
<td>Designated users can open and read the contents of a file.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Directory</td>
<td>Designated users can list files in the directory.</td>
</tr>
<tr>
<td>w</td>
<td>Write</td>
<td>File</td>
<td>Designated users can modify the contents of the file or delete the file.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Directory</td>
<td>Designated users can add files or add links in the directory. They can also remove files or remove links in the directory.</td>
</tr>
<tr>
<td>x</td>
<td>Execute</td>
<td>File</td>
<td>Designated users can execute the file, if it is a program or shell script. They also can run the program with one of the exec(2) system calls.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Directory</td>
<td>Designated users can open files or execute files in the directory. They also can make the directory and the directories beneath it current.</td>
</tr>
<tr>
<td>-</td>
<td>Denied</td>
<td>File and Directory</td>
<td>Designated users cannot read, write, or execute the file.</td>
</tr>
</tbody>
</table>

These file permissions apply to regular files, and to special files such as devices, sockets, and named pipes (FIFOs).

For a symbolic link, the permissions that apply are the permissions of the file that the link points to.

You can protect the files in a directory and its subdirectories by setting restrictive file permissions on that directory. Note, however, that superuser has access to all files and directories on the system.

### Special File Permissions (setuid, setgid and Sticky Bit)

Three special types of permissions are available for executable files and public directories: setuid, setgid, and sticky bit. When these permissions are set, any user who runs that executable file assumes the ID of the owner (or group) of the executable file.

You must be extremely careful when you set special permissions, because special permissions constitute a security risk. For example, a user can gain superuser capabilities by executing a program that sets the user ID (UID) to 0, which is the UID of root. Also, all users can set special permissions for files that they own, which constitutes another security concern.

You should monitor your system for any unauthorized use of the setuid permission and the setgid permission to gain superuser capabilities. A suspicious permission grants ownership of an administrative program to a user rather than to root or bin. To search for and list all files that use this special permission, see “How to Find Files With Special File Permissions” on page 149.
**setuid Permission**

When `setuid` permission is set on an executable file, a process that runs this file is granted access on the basis of the owner of the file. The access is *not* based on the user who is running the executable file. This special permission allows a user to access files and directories that are normally available only to the owner.

For example, the `setuid` permission on the `passwd` command makes it possible for users to change passwords. A `passwd` command with `setuid` permission would resemble the following:

```
-r-sr-x 3 root sys 28144 Jun 17 12:02 /usr/bin/passwd
```

This special permission presents a security risk. Some determined users can find a way to maintain the permissions that are granted to them by the `setuid` process even after the process has finished executing.

*Note* – The use of `setuid` permissions with the reserved UIDs (0–100) from a program might not set the effective UID correctly. Use a shell script, or avoid using the reserved UIDs with `setuid` permissions.

**setgid Permission**

The `setgid` permission is similar to the `setuid` permission. The process’s effective group ID (GID) is changed to the group that owns the file, and a user is granted access based on the permissions that are granted to that group. The `/usr/bin/mail` command has `setgid` permissions:

```
-r-x--s--x 1 root mail 67504 Jun 17 12:01 /usr/bin/mail
```

When the `setgid` permission is applied to a directory, files that were created in this directory belong to the group to which the directory belongs. The files do not belong to the group to which the creating process belongs. Any user who has write and execute permissions in the directory can create a file there. However, the file belongs to the group that owns the directory, not to the group that the user belongs to.

You should monitor your system for any unauthorized use of the `setgid` permission to gain superuser capabilities. A suspicious permission grants group access to such a program to an unusual group rather than to `root` or `bin`. To search for and list all files that use this permission, see “How to Find Files With Special File Permissions” on page 149.

**Sticky Bit**

The *sticky bit* is a permission bit that protects the files within a directory. If the directory has the sticky bit set, a file can be deleted only by the file owner, the directory owner, or by a privileged user. The `root` user and the Primary Administrator role are examples of privileged users. The sticky bit prevents a user from deleting other users’ files from public directories such as `/tmp:`
Be sure to set the sticky bit manually when you set up a public directory on a TMPFS file system. For instructions, see Example 7–5.

**Default umask Value**

When you create a file or directory, you create it with a default set of permissions. The system defaults are open. A text file has 666 permissions, which grants read and write permission to everyone. A directory and an executable file have 777 permissions, which grants read, write, and execute permission to everyone. Typically, users override the system defaults in their /etc/profile file, .cshrc file, or .login file.

The value assigned by the umask command is subtracted from the default. This process has the effect of denying permissions in the same way that the chmod command grants them. For example, the chmod 022 command grants write permission to group and others. The umask 022 command denies write permission to group and others.

The following table shows some typical umask settings and their effect on an executable file.

<table>
<thead>
<tr>
<th>Level of Security</th>
<th>umask Setting</th>
<th>Permissions Disallowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permissive (744)</td>
<td>022</td>
<td>w for group and others</td>
</tr>
<tr>
<td>Moderate (740)</td>
<td>027</td>
<td>w for group, rwx for others</td>
</tr>
<tr>
<td>Moderate (741)</td>
<td>026</td>
<td>w for group, rw for others</td>
</tr>
<tr>
<td>Severe (700)</td>
<td>077</td>
<td>rwx for group and others</td>
</tr>
</tbody>
</table>

For more information on setting the umask value, see the umask(1) man page.

**File Permission Modes**

The chmod command enables you to change the permissions on a file. You must be superuser or the owner of a file or directory to change its permissions.
You can use the `chmod` command to set permissions in either of two modes:

- **Absolute Mode** – Use numbers to represent file permissions. When you change permissions by using the absolute mode, you represent permissions for each triplet by an octal mode number. Absolute mode is the method most commonly used to set permissions.

- **Symbolic Mode** – Use combinations of letters and symbols to add permissions or remove permissions.

The following table lists the octal values for setting file permissions in absolute mode. You use these numbers in sets of three to set permissions for owner, group, and other, in that order. For example, the value 644 sets read and write permissions for owner, and read-only permissions for group and other.

**TABLE 7–4 Setting File Permissions in Absolute Mode**

<table>
<thead>
<tr>
<th>Octal Value</th>
<th>File Permissions Set</th>
<th>Permissions Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>- - -</td>
<td>No permissions</td>
</tr>
<tr>
<td>1</td>
<td>- - x</td>
<td>Execute permission only</td>
</tr>
<tr>
<td>2</td>
<td>- w-</td>
<td>Write permission only</td>
</tr>
<tr>
<td>3</td>
<td>- wx</td>
<td>Write and execute permissions</td>
</tr>
<tr>
<td>4</td>
<td>r - -</td>
<td>Read permission only</td>
</tr>
<tr>
<td>5</td>
<td>r - x</td>
<td>Read and execute permissions</td>
</tr>
<tr>
<td>6</td>
<td>r w-</td>
<td>Read and write permissions</td>
</tr>
<tr>
<td>7</td>
<td>r wx</td>
<td>Read, write, and execute permissions</td>
</tr>
</tbody>
</table>

The following table lists the symbols for setting file permissions in symbolic mode. Symbols can specify whose permissions are to be set or changed, the operation to be performed, and the permissions that are being assigned or changed.

**TABLE 7–5 Setting File Permissions in Symbolic Mode**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td>who</td>
<td>User (owner)</td>
</tr>
<tr>
<td>g</td>
<td>who</td>
<td>Group</td>
</tr>
<tr>
<td>o</td>
<td>who</td>
<td>Others</td>
</tr>
<tr>
<td>a</td>
<td>who</td>
<td>All</td>
</tr>
<tr>
<td>=</td>
<td>operator</td>
<td>Assign</td>
</tr>
</tbody>
</table>
TABLE 7–5 Setting File Permissions in Symbolic Mode  (Continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>operator</td>
<td>Add</td>
</tr>
<tr>
<td>-</td>
<td>operator</td>
<td>Remove</td>
</tr>
<tr>
<td>r</td>
<td>permissions</td>
<td>Read</td>
</tr>
<tr>
<td>w</td>
<td>permissions</td>
<td>Write</td>
</tr>
<tr>
<td>x</td>
<td>permissions</td>
<td>Execute</td>
</tr>
<tr>
<td>l</td>
<td>permissions</td>
<td>Mandatory locking, setgid bit is on, group execution bit is off</td>
</tr>
<tr>
<td>s</td>
<td>permissions</td>
<td>setuid or setgid bit is on</td>
</tr>
<tr>
<td>t</td>
<td>permissions</td>
<td>Sticky bit is on, execution bit for others is on</td>
</tr>
</tbody>
</table>

The who operator permissions designations in the function column specify the symbols that change the permissions on the file or directory.

who Specifies whose permissions are to be changed.

operator Specifies the operation to be performed.

permissions Specifies what permissions are to be changed.

You can set special permissions on a file in absolute mode or symbolic mode. However, you must use symbolic mode to set or remove setuid permissions on a directory. In absolute mode, you set special permissions by adding a new octal value to the left of the permission triplet. The following table lists the octal values for setting special permissions on a file.

TABLE 7–6 Setting Special File Permissions in Absolute Mode

<table>
<thead>
<tr>
<th>Octal Value</th>
<th>Special File Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sticky bit</td>
</tr>
<tr>
<td>2</td>
<td>setgid</td>
</tr>
<tr>
<td>4</td>
<td>setuid</td>
</tr>
</tbody>
</table>

Using Access Control Lists to Protect Files

Traditional UNIX file protection provides read, write, and execute permissions for the three user classes: file owner, file group, and other. An access control list (ACL) provides better file security by enabling you to do the following:

- Define file permissions for the file owner, the group, other, specific users and groups
- Define default permissions for each of the preceding categories
For example, if you want everyone in a group to be able to read a file, you can simply grant group read permissions on that file. Now, assume that you want only one person in the group to be able to write to that file. Standard UNIX does not provide that level of file security. However, an ACL provides this level of file security.

ACL entries define an ACL on a file. The entries are set through the `setfacl` command. ACL entries consist of the following fields separated by colons:

```
entry-type: [uid | gid]: perms
```

- **entry-type**: Is the type of ACL entry on which to set file permissions. For example, `entry-type` can be `user` (the owner of a file) or `mask` (the ACL mask). For a listing of ACL entries, see Table 7–7 and Table 7–8.
- **uid**: Is the user name or user ID (UID).
- **gid**: Is the group name or group ID (GID).
- **perms**: Represents the permissions that are set on `entry-type`. `perms` can be indicated by the symbolic characters `rwx` or an octal number. These are the same numbers that are used with the `chmod` command.

In the following example, an ACL entry sets read and write permissions for the user `stacey`.

```
user:stacey:rw-
```

**Caution** – UFS file system attributes such as ACLs are supported in UFS file systems only. Thus, if you restore or copy files with ACL entries into the `/tmp` directory, which is usually mounted as a TMPFS file system, the ACL entries will be lost. Use the `/var/tmp` directory for temporary storage of UFS files.

### ACL Entries for Files

The following table lists the valid ACL entries that you might use when setting ACLs on files. The first three ACL entries provide the basic UNIX file protection.

<table>
<thead>
<tr>
<th>ACL Entry</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>u[ser]:perms</td>
<td>File owner permissions.</td>
</tr>
<tr>
<td>g[roup]:perms</td>
<td>File group permissions.</td>
</tr>
<tr>
<td>o[ther]:perms</td>
<td>Permissions for users other than the file owner or members of the file group.</td>
</tr>
</tbody>
</table>
TABLE 7–7  ACL Entries for Files  (Continued)

<table>
<thead>
<tr>
<th>ACL Entry</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>m[ask]:perms</td>
<td>The ACL mask. The mask entry indicates the maximum permissions that are allowed for users (other than the owner) and for groups. The mask is a quick way to change permissions on all the users and groups. For example, the mask: r-- mask entry indicates that users and groups cannot have more than read permissions, even though they might have write and execute permissions.</td>
</tr>
<tr>
<td>u[ser]:uid:perms</td>
<td>Permissions for a specific user. For uid, you can specify either a user name or a numeric UID.</td>
</tr>
<tr>
<td>g[roup]:gid:perms</td>
<td>Permissions for a specific group. For gid, you can specify either a group name or a numeric GID.</td>
</tr>
</tbody>
</table>

ACLEntriesforDirectories

In addition to the ACL entries that are described in Table 7–7, you can set default ACL entries on a directory. Files or directories created in a directory that has default ACL entries will have the same ACL entries as the default ACL entries. Table 7–8 lists the default ACL entries for directories.

When you set default ACL entries for specific users and groups on a directory for the first time, you must also set default ACL entries for the file owner, file group, others, and the ACL mask. These entries are required. They are the first four default ACL entries in the following table.

TABLE 7–8  Default ACL Entries for Directories

<table>
<thead>
<tr>
<th>Default ACL Entry</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>d[efault]:u[ser]:perms</td>
<td>Default file owner permissions.</td>
</tr>
<tr>
<td>d[efault]:g[roup]:perms</td>
<td>Default file group permissions.</td>
</tr>
<tr>
<td>d[efault]:o[ther]:perms</td>
<td>Default permissions for users other than the file owner or members of the file group.</td>
</tr>
<tr>
<td>d[efault]:m[ask]:perms</td>
<td>Default ACL mask.</td>
</tr>
<tr>
<td>d[efault]:u[ser]:uid:perms</td>
<td>Default permissions for a specific user. For uid, you can specify either a user name or a numeric UID.</td>
</tr>
<tr>
<td>d[efault]:g[roup]:gid:perms</td>
<td>Default permissions for a specific group. For gid, you can specify either a group name or a numeric GID.</td>
</tr>
</tbody>
</table>

Commands for Administering ACLs

The following commands administer ACLs on files or directories.
Preventing Executable Files From Compromising Security

A number of security bugs are related to default executable stacks when their permissions are set to read, write, and execute. While stacks with execute permissions are allowed, most programs can function correctly without using executable stacks.

The noexec_user_stack variable enables you to specify whether stack mappings are executable. The variable is available as of the Solaris 2.6 release. By default, the variable is set to zero, except on 64-bit applications, which provides ABI-compliant behavior. If the variable is set to a non-zero value, the system marks the stack of every process in the system as readable and writable, but not executable.

Once this variable is set, programs that attempt to execute code on their stack are sent a SIGSEGV signal. This signal usually results in the program terminating with a core dump. Such programs also generate a warning message that includes the name of the offending program, the process ID, and the real UID of the user who ran the program. For example:

a.out[347] attempt to execute code on stack by uid 555

The message is logged by the syslog daemon when the syslog kern facility is set to notice level. This logging is set by default in the syslog.conf file, which means that the message is sent to both the console and the /var/adm/messages file. For more information, see the syslogd(1M) and syslog.conf(4) man pages.

The syslog message is useful for observing potential security problems. The message also identifies valid programs that depend upon executable stacks that have been prevented from correct operation by setting this variable. If you do not want any messages logged, then set the noexec_user_stack_log variable to zero in the /etc/system file. Even though messages are not being logged, the SIGSEGV signal can continue to cause the executing program to terminate with a core dump.

You can use the mprotect() function if you want programs to explicitly mark their stack as executable. For more information, see the mprotect(2) man page.

Because of hardware limitations, the capability of catching and reporting executable stack problems is not available on most x86 based systems. Systems in the AMD64 product family can catch and report executable stack problems.
Protecting Files (Task Map)

The following task map points to sets of procedures for protecting files.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use UNIX permissions to protect files</td>
<td>Views UNIX permissions on files. Protects files with UNIX permissions.</td>
<td>“Protecting Files With UNIX Permissions (Task Map)” on page 137</td>
</tr>
<tr>
<td>Use ACLs to protect files</td>
<td>Adds ACLs to protect files at a more granular level than UNIX permissions</td>
<td>“Protecting Files With ACLs (Task Map)” on page 143</td>
</tr>
<tr>
<td>Protect system from files that pose a security risk</td>
<td>Finds executable files that have suspicious ownership. Disables files that can damage the system.</td>
<td>“Protecting Against Programs With Security Risk (Task Map)” on page 149</td>
</tr>
</tbody>
</table>

Protecting Files With UNIX Permissions (Task Map)

The following task map points to procedures that list file permissions, change file permissions, and protect files with special file permissions.

<table>
<thead>
<tr>
<th>Task</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display file information</td>
<td>“How to Display File Information” on page 137</td>
</tr>
<tr>
<td>Change file ownership</td>
<td>“How to Change the Owner of a File” on page 139</td>
</tr>
<tr>
<td></td>
<td>“How to Change Group Ownership of a File” on page 140</td>
</tr>
<tr>
<td>Change file permissions</td>
<td>“How to Change File Permissions in Symbolic Mode” on page 140</td>
</tr>
<tr>
<td></td>
<td>“How to Change File Permissions in Absolute Mode” on page 141</td>
</tr>
<tr>
<td></td>
<td>“How to Change Special File Permissions in Absolute Mode” on page 142</td>
</tr>
</tbody>
</table>

▼ How to Display File Information

Display information about all the files in a directory by using the `ls` command.

- Type the following command to display a long listing of all files in the current directory.

```bash
% ls -la
```

- `-l` Displays the long format that includes user ownership, group ownership, and file permissions.
- `-a` Displays all files, including hidden files that begin with a dot (`.`).
Example 7-1  Displaying File Information

In the following example, a partial list of the files in the /sbin directory is displayed.

```bash
% cd /sbin
% ls -la
```
```
total 13456
-rwxr-xr-x 2 root sys 512 Sep 1 14:11 .
-rwxr-xr-x 29 root root 1024 Sep 1 15:40 ..
-r-xr-xr-x 1 root bin 218188 Aug 18 15:17 autopush
lrwxrwxrwx 1 root root 21 Sep 1 14:11 bpgetfile -> ...
-r-xr-xr-x 1 root bin 505556 Aug 18 13:24 dhcpagent
-r-xr-xr-x 1 root bin 456064 Aug 20 13:25 dhcpinfo
-r-xr-xr-x 1 root bin 272360 Aug 18 15:19 fdisk
-r-xr-xr-x 1 root bin 824728 Aug 20 13:29 hostconfig
-r-xr-xr-x 1 root bin 603528 Aug 20 13:21 ifconfig
-r-xr-xr-x 1 root sys 556008 Aug 20 13:21 init
-r-xr-xr-x 2 root root 274020 Aug 18 15:28 jsh
-r-xr-xr-x 1 root bin 238736 Aug 21 19:46 mount
-r-xr-xr-x 1 root sys 7696 Aug 18 15:20 mountall
```
```

Each line displays information about a file in the following order:

- **Type of file** – For example, d. For list of file types, see "File and Directory Ownership" on page 128.
- **Permissions** – For example, r-xr-xr-x. For description, see “File and Directory Ownership” on page 128.
- **Number of hard links** – For example, 2.
- **Owner of the file** – For example, root.
- **Group of the file** – For example, bin.
- **Size of the file, in bytes** – For example, 7696.
- **Date the file was created or the last date that the file was changed** – For example, Aug 18 15:20.
- **Name of the file** – For example, mountall.
How to Change the Owner of a File

The file owner, the Primary Administrator role, or superuser can change any file's ownership.

1 Display the permissions on a file.
   
   ```
   % ls -l example-file
   -rw-r--r-- 1 janedoe staff 112640 May 24 10:49 example-file
   ```

2 Assume the Primary Administrator role, or become superuser.
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

3 Change the owner of the file.
   
   ```
   # chown stacey example-file
   ```

4 Verify that the owner of the file has changed.
   
   ```
   # ls -l example-file
   -rw-r--r-- 1 stacey staff 112640 May 26 08:50 example-file
   ```

Enabling Users to Change the Ownership of Files That Others Own

Security Consideration – You should have good reason to override system security policy by changing the setting of the `rstchown` variable to zero. Any user who accesses the system can change the ownership of any file on the system.

In this example, the value of the `rstchown` variable is set to zero in the `/etc/system` file. This setting enables the owner of a file to use the `chown` command to change the file's ownership to another user. This setting also enables the owner to use the `chgrp` command to set the group ownership of a file to a group that the owner does not belong to. The change goes into effect when the system is rebooted.

```
set rstchown = 0
```

For more information, see the `chown(1)` and `chgrp(1)` man pages.

Also, be aware that NFS-mounted file systems have further restrictions on changing ownership and groups. For more information on restricting access to NFS-mounted systems, see Chapter 6, “Accessing Network File Systems (Reference),” in System Administration Guide: Network Services.
How to Change Group Ownership of a File

1. Assume the Primary Administrator role, or become superuser.
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2. Change the group ownership of a file.
   
   $ chgrp scifi example-file
   
   For information on setting up groups, see Chapter 4, “Managing User Accounts and Groups (Overview),” in System Administration Guide: Basic Administration.

3. Verify that the group ownership of the file has changed.
   
   $ ls -l example-file
   
   -rw-r--r-- 1 stacey scifi 112640 June 20 08:55 example-file
   
   Also see Example 7–2.

How to Change File Permissions in Symbolic Mode

1. If you are not the owner of the file or directory, become superuser or assume an equivalent role.
   Only the current owner or superuser can use the chmod command to change file permissions on a file or directory.

2. Change permissions in symbolic mode.
   
   % chmod who operator permissions filename
   
   who Specifies whose permissions are to be changed.
   operator Specifies the operation to be performed.
   permissions Specifies what permissions are to be changed. For the list of valid symbols, see Table 7–5.
   filename Specifies the file or directory.

3. Verify that the permissions of the file have changed.
   
   % ls -l filename
Changing Permissions in Symbolic Mode

In the following example, read permission is taken away from others.

```
% chmod o-r example-file1
```

In the following example, read and execute permissions are added for user, group, and others.

```
$ chmod a+rx example-file2
```

In the following example, read, write, and execute permissions are assigned to group.

```
$ chmod g=rx example-file3
```

How to Change File Permissions in Absolute Mode

1. **If you are not the owner of the file or directory, become superuser or assume an equivalent role.**

   Only the current owner or superuser can use the `chmod` command to change file permissions on a file or directory.

2. **Change permissions in absolute mode.**

   ```
   % chmod nnn filename
   ```

   `nnn` Specifies the octal values that represent the permissions for the file owner, file group, and others, in that order. For the list of valid octal values, see Table 7–4.

   `filename` Specifies the file or directory.

   **Note** – When you use the `chmod` command to change the file group permissions on a file with ACL entries, both the file group permissions and the ACL mask are changed to the new permissions. Be aware that the new ACL mask permissions can change the permissions for other users and groups who have ACL entries on the file. Use the `getfacl` command to make sure that the appropriate permissions are set for all ACL entries. For more information, see the `getfacl(1)` man page.

3. **Verify that the permissions of the file have changed.**

   ```
   % ls -l filename
   ```
Changing Permissions in Absolute Mode

In the following example, the permissions of a public directory are changed from 744 (read, write, execute; read-only; and read-only) to 755 (read, write, execute; read and execute; and read and execute).

```
# ls -ld public_dir
drwxr--r-- 1 ignatz staff 6023 Aug 5 12:06 public_dir
# chmod 755 public_dir
# ls -ld public_dir
 drwxr-xr-x 1 ignatz staff 6023 Aug 5 12:06 public_dir
```

In the following example, the permissions of an executable shell script are changed from read and write to read, write, and execute.

```
% ls -l my_script
-rw------- 1 ignatz staff 6023 Aug 5 12:06 my_script
% chmod 700 my_script
% ls -l my_script
-rwx------ 1 ignatz staff 6023 Aug 5 12:06 my_script
```

## How to Change Special File Permissions in Absolute Mode

1. **If you are not the owner of the file or directory, become superuser or assume an equivalent role.**
   Only the current owner or a user with superuser capabilities can use the `chmod` command to change the special permissions on a file or directory.

2. **Change special permissions in absolute mode.**
   
   ```
   % chmod nnnn filename
   
   nnnn          Specifies the octal values that change the permissions on the file or directory. The leftmost octal value sets the special permissions on the file. For the list of valid octal values for special permissions, see Table 7–6.
   
   filename      Specifies the file or directory.
   ```
Note – When you use the `chmod` command to change the file group permissions on a file with ACL entries, both the file group permissions and the ACL mask are changed to the new permissions. Be aware that the new ACL mask permissions can change the permissions for additional users and groups who have ACL entries on the file. Use the `getfacl` command to make sure that the appropriate permissions are set for all ACL entries. For more information, see the `getfacl(1)` man page.

3 Verify that the permissions of the file have changed.

```
% ls -l filename
```

**Example 7–5** Setting Special File Permissions in Absolute Mode

In the following example, the `setuid` permission is set on the `dbprog` file.

```
# chmod 4555 dbprog
# ls -l dbprog
-r-sr-xr-x 1 db staff 12095 May 6 09:29 dbprog
```

In the following example, the `setgid` permission is set on the `dbprog2` file.

```
# chmod 2551 dbprog2
# ls -l dbprog2
-r-xr-s--x 1 db staff 24576 May 6 09:30 dbprog2
```

In the following example, the sticky bit permission is set on the `public_dir` directory.

```
# chmod 1777 public_dir
# ls -ld public_dir
drwxrwxrwt 2 ignatz staff 512 May 15 15:27 public_dir
```

**Protecting Files With ACLs (Task Map)**

The following task map points to procedures that list the ACLs on a file, change the ACLs, and copy the ACLs to another file.

<table>
<thead>
<tr>
<th>Task</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determine if a file has an ACL</td>
<td>&quot;How to Check if a File Has an ACL&quot; on page 144</td>
</tr>
<tr>
<td>Add an ACL to a file</td>
<td>&quot;How to Add ACL Entries to a File&quot; on page 144</td>
</tr>
<tr>
<td>Copy an ACL</td>
<td>&quot;How to Copy an ACL&quot; on page 146</td>
</tr>
</tbody>
</table>
### How to Check if a File Has an ACL

- **Check if a file has an ACL.**

  ```bash
  ls -l filename
  ```

  where `filename` specifies the file or directory.

  In the output, a plus sign (+) to the right of the mode field indicates that the file has an ACL.

  **Note** – Unless you have added ACL entries that extend UNIX file permissions, a file is considered to have a "trivial" ACL and the plus sign (+) does not display.

### Example 7–6 Checking if a File Has an ACL

In the following example, the `ch1.sgm` file has an ACL. The ACL is indicated by the plus sign (+) to the right of the mode field.

```bash
ls -l ch1.sgm
```

```
-rwxr-----+ 1 stacey techpubs 167 Nov 11 11:13 ch1.sgm
```

### How to Add ACL Entries to a File

1. **Set an ACL on a file by using the `setfacl` command.**

   ```bash
   setfacl -s user::perm,group::perm,other::perm,mask::perm acl-entry-list filename ...
   ```

   - `-s` Sets an ACL on the file. If a file already has an ACL, it is replaced. This option requires at least the `user::`, `group::`, and `other::` entries.

   - `user::perm` Specifies the file owner permissions.

   - `group::perm` Specifies the group ownership permissions.

   - `other::perm` Specifies the permissions for users other than the file owner or members of the group.

   - `mask::perm` Specifies the permissions for the ACL mask. The mask indicates the maximum permissions that are allowed for users (other than the owner) and for groups.
acl-entry-list  Specifies the list of one or more ACL entries to set for specific users and groups on the file or directory. You can also set default ACL entries on a directory. 

Table 7–7 and Table 7–8 show the valid ACL entries.

filename ...  Specifies one or more files or directories on which to set the ACL. Multiple filenames are separated by spaces.

Caution – If an ACL already exists on the file, the -s option replaces the entire ACL with the new ACL.

For more information, see the setfacl(1) man page.

2 Verify that the ACL entries were set on the file.

% getfacl filename

For more information, see “How to Check if a File Has an ACL” on page 144.

Example 7–7 Setting an ACL on a File

In the following example, the file owner permissions are set to read and write, file group permissions are set to read only, and other permissions are set to none on the ch1.sgm file. In addition, the user anusha is given read and write permissions on the file. The ACL mask permissions are set to read and write, which means that no user or group can have execute permissions.

% setfacl -s user::rw-,group::r--,other:---,mask:rw-,user:anusha:rw- ch1.sgm
% ls -l
total 124
-rw-r-----+ 1 stacey techpubs 34816 Nov 11 14:16 ch1.sgm
-rw-r--r-- 1 stacey techpubs 20167 Nov 11 14:16 ch2.sgm
-rw-r--r-- 1 stacey techpubs 8192 Nov 11 14:16 notes
% getfacl ch1.sgm
# file: ch1.sgm
# owner: stacey
# group: techpubs
user::rw-
user:anusha:rw-  #effective:rw-
group::r--  #effective:r--
mask:rw-
other:---

In the following example, the file owner permissions are set to read, write, and execute, file group permissions are set to read only, other permissions are set to none. In addition, the ACL mask permissions are set to read on the ch2.sgm file. Finally, the user anusha is given read and write permissions. However, due to the ACL mask, the permissions for anusha are read only.
Protecting Files With ACLs (Task Map)

% setfacl -s u::7,g::4,o:0,m:4,u:anusha:7 ch2.sgm
% getfacl ch2.sgm
# file: ch2.sgm
# owner: stacey
# group: techpubs
user::rwx
user:anusha:rwx #effective:r--
group::r-- #effective:r--
mask:r--
other:---

▼ How to Copy an ACL

- Copy a file’s ACL to another file by redirecting the `getfacl` output.
  % getfacl filename1 | setfacl -f - filename2
  
  `filename1` Specifies the file from which to copy the ACL.
  `filename2` Specifies the file on which to set the copied ACL.

Example 7–8 Copying an ACL

In the following example, the ACL on ch2.sgm is copied to ch3.sgm.

% getfacl ch2.sgm | setfacl -f - ch3.sgm

▼ How to Change ACL Entries on a File

1 Modify ACL entries on a file by using the `setfacl` command.
  % setfacl -m acl-entry-list filename ...
  
  `-m` Modifies the existing ACL entry.
  `acl-entry-list` Specifies the list of one or more ACL entries to modify on the file or directory.
  You can also modify default ACL entries on a directory. Table 7–7 and Table 7–8 show the valid ACL entries.
  `filename` Specifies one or more files or directories, separated by a space.

2 Verify that the ACL entries were modified on the file.
  % getfacl filename
Example 7–9  Modifying ACL Entries on a File

In the following example, the permissions for the user anusha are modified to read and write.

```bash
% setfacl -m user:anusha:6 ch3.sgm
% getfacl ch3.sgm
```

```
# file: ch3.sgm
# owner: stacey
# group: techpubs
user::rw-
user::anusha:rw-  #effective:r--
group::r-     #effective:r--
mask:r--
other:r-
```

In the following example, the default permissions for the group staff are modified to read on the book directory. In addition, the default ACL mask permissions are modified to read and write.

```bash
% setfacl -m default:group:staff:4,default:mask:6 book
```

▼ How to Delete ACL Entries From a File

1  Delete ACL entries from a file.

   ```bash
   % setfacl -d acl-entry-list filename ...
   
   -d                  Deletes the specified ACL entries.
   acl-entry-list      Specifies the list of ACL entries (without specifying the permissions) to delete from the file or directory. You can only delete ACL entries and default ACL entries for specific users and groups. Table 7–7 and Table 7–8 show the valid ACL entries.
   filename ...       Specifies one or more files or directories, separated by a space.
   
   Alternatively, you can use the setfacl -s command to delete all the ACL entries on a file and replace them with the new ACL entries that are specified.
   ```

2  Verify that the ACL entries were deleted from the file.

   ```bash
   % getfacl filename
   ```
Example 7–10  Deleting ACL Entries on a File

In the following example, the user anusha is deleted from the ch4.sgm file.

% setfacl -d user:anusha ch4.sgm

▼ How to Display ACL Entries for a File

● Display ACL entries for a file by using the `getfacl` command.

% getfacl [-a | -d] filename ...

- `-a` Displays the file name, file owner, file group, and ACL entries for the specified file or directory.

- `-d` Displays the file name, file owner, file group, and the default ACL entries, if they exist, for the specified directory.

`filename ...` Specifies one or more files or directories, separated by a space.

If you specify multiple file names on the command line, the ACL entries are displayed with a blank line between each entry.

Example 7–11  Displaying ACL Entries for a File

In the following example, all the ACL entries for the ch1.sgm file are displayed. The `#effective:` note beside the user and group entries indicates what the permissions are after being modified by the ACL mask.

% getfacl ch1.sgm

# file: ch1.sgm
# owner: stacey
# group: techpubs
user::rw-
user:anusha:r-    #effective:r--
group::rw-        #effective:rw-
mask:rw-
other:---

In the following example, the default ACL entries for the book directory are displayed.

% getfacl -d book

# file: book
# owner: stacey
Protecting Against Programs With Security Risk (Task Map)

The following task map points to procedures that find risky executables on the system, and that prevent programs from exploiting an executable stack.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find files with special permissions</td>
<td>Locates files with the setuid bit set, but that are not owned by the root user.</td>
<td>“How to Find Files With Special File Permissions” on page 149</td>
</tr>
<tr>
<td>Prevent executable stack from overflowing</td>
<td>Prevents programs from exploiting an executable stack.</td>
<td>“How to Disable Programs From Using Executable Stacks” on page 150</td>
</tr>
<tr>
<td>Prevent logging of executable stack messages</td>
<td>Turns off logging of executable stack messages.</td>
<td>Example 7–13</td>
</tr>
</tbody>
</table>

▼ How to Find Files With Special File Permissions

You should monitor your system for any unauthorized use of the setuid and setgid permissions on programs. The setuid and setgid permissions enable ordinary users to gain superuser capabilities. A suspicious executable file grants ownership to a user rather than to root or bin.

1. **Assume the Primary Administrator role, or become superuser.**
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2. **Find files with setuid permissions by using the find command.**
   ```bash
   # find directory -user root -perm -4000 -exec ls -ldb {} \; >/tmp/ filename
   ```
find directory  Checks all mounted paths starting at the specified directory, which can be root (/), sys, bin, or mail.
-user root  Displays files owned only by root.
-perm -4000  Displays files only with permissions set to 4000.
-exec ls -ldb  Displays the output of the find command in ls -ldb format.
>/tmp/filename  Is the file that contains the results of the find command.

3  Display the results in /tmp/filename.
   # more /tmp/filename

For background information on setuid permissions, see “setuid Permission” on page 130.

Example 7–12  Finding Files With setuid Permissions

The output from the following example shows that a user named rar has made a personal copy of /usr/bin/sh, and has set the permissions as setuid to root. As a result, the /usr/rar/bin/sh program runs with root permissions.

This output was saved for future reference by moving the file out of the /tmp directory.

# find / -user root -perm -4000 -exec ls -ldb {} \; > /var/tmp/ckprm
# cat /var/tmp/ckprm
-r-sr-xr-x 1 root bin 38836 Aug 10 16:16 /usr/bin/at
-r-sr-xr-x 1 root bin 19812 Aug 10 16:16 /usr/bin/crontab
---s--x--x 1 root sys 46040 Aug 10 15:18 /usr/bin/ct
-r-sr-xr-x 1 root sys 12092 Aug 11 01:29 /usr/lib/mv_dir
-r-sr-xr-x 1 root bin 33208 Aug 10 15:55 /usr/lib/lpadmin
-r-sr-xr-x 1 root bin 38696 Aug 10 15:55 /usr/lib/lpsched
---s--x--- 1 root rar 45376 Aug 18 15:11 /usr/rar/bin/sh
-r-sr-xr-x 1 root bin 12524 Aug 11 01:27 /usr/bin/df
-rw-r-xr-x 1 root sys 21780 Aug 11 01:27 /usr/bin/newgrp
-r-r-sr-x 1 root sys 23000 Aug 11 01:27 /usr/bin/passwd
-r-r-sr-x 1 root sys 23824 Aug 11 01:27 /usr/bin/su
# mv /var/tmp/ckprm /export/sysreports/ckprm

How to Disable Programs From Using Executable Stacks

For a description of the security risks of executable stacks, see “Preventing Executable Files From Compromising Security” on page 136.
1. **Assume the Primary Administrator role, or become superuser.**
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2. **Edit the /etc/system file, and add the following line:**
   ```
   set noexec_user_stack=1
   ```

3. **Reboot the system.**
   ```
   # init 6
   ```

**Example 7–13 Disabling the Logging of Executable Stack Messages**

In this example, the logging of executable stack messages is disabled, and then the system is rebooted.

```plaintext
# cat /etc/system
set noexec_user_stack=1
set noexec_user_stack_log=0
# init 6
```
Roles, Rights Profiles, and Privileges

This section covers role-based access control (RBAC) and process rights management. RBAC components include roles, rights profiles, and authorizations. Process rights management is implemented through privileges. Privileges work with RBAC to provide a more secure administration alternative than administration of a system by a superuser.
Using Roles and Privileges (Overview)

Solaris role-based access control (RBAC) and privileges provide a more secure alternative to superuser. This chapter provides overview information about RBAC and about privileges.

The following is a list of the overview information in this chapter.

- “Role-Based Access Control (Overview)” on page 155
- “Privileges (Overview)” on page 164

Role-Based Access Control (Overview)

Role-based access control (RBAC) is a security feature for controlling user access to tasks that would normally be restricted to superuser. By applying security attributes to processes and to users, RBAC can divide up superuser capabilities among several administrators. Process rights management is implemented through privileges. User rights management is implemented through RBAC.

- For a discussion of process rights management, see “Privileges (Overview)” on page 164.
- For information on RBAC tasks, see Chapter 9, “Using Role-Based Access Control (Tasks).”
- For reference information, see Chapter 10, “Role-Based Access Control (Reference).”

RBAC: An Alternative to the Superuser Model

In conventional UNIX systems, the root user, also referred to as superuser, is all-powerful. Programs that run as root, or setuid programs, are all-powerful. The root user has the ability to read and write to any file, run all programs, and send kill signals to any process. Effectively, anyone who can become superuser can modify a site’s firewall, alter the audit trail, read confidential records, and shut down the entire network. A setuid program that is hijacked can do anything on the system.
Role-based access control (RBAC) provides a more secure alternative to the all-or-nothing superuser model. With RBAC, you can enforce security policy at a more fine-grained level. RBAC uses the security principle of least privilege. Least privilege means that a user has precisely the amount of privilege that is necessary to perform a job. Ordinary users have enough privilege to use their applications, check the status of their jobs, print files, create new files, and so on. Capabilities beyond ordinary user capabilities are grouped into rights profiles. Users who are expected to do jobs that require some of the capabilities of superuser assume a role that includes the appropriate rights profile.

RBAC collects superuser capabilities into rights profiles. These rights profiles are assigned to special user accounts that are called roles. A user can then assume a role to do a job that requires some of superuser’s capabilities. Predefined rights profiles are supplied with Solaris software. You create the roles and assign the profiles.

Rights profiles can provide broad capabilities. For example, the Primary Administrator rights profile is equivalent to superuser. Rights profiles can also be narrowly defined. For example, the Cron Management rights profile manages at and cron jobs. When you create roles, you can decide to create roles with broad capabilities, or roles with narrow capabilities, or both.

In the RBAC model, superuser creates one or more roles. The roles are based on rights profiles. Superuser then assigns the roles to users who are trusted to perform the tasks of the role. Users log in with their user name. After login, users assume roles that can run restricted administrative commands and graphical user interface (GUI) tools.

The flexibility in setting up roles enables a variety of security policies. Although no roles are shipped with the Solaris Operating System (Solaris OS), three recommended roles can easily be configured. The roles are based on rights profiles of the same name:

- **Primary Administrator** – A powerful role that is equivalent to the root user, or superuser.
- **System Administrator** – A less powerful role for administration that is not related to security. This role can manage file systems, mail, and software installation. However, this role cannot set passwords.
- **Operator** – A junior administrator role for operations such as backups and printer management.

These three roles do not have to be implemented. Roles are a function of an organization’s security needs. Roles can be set up for special-purpose administrators in areas such as security, networking, or firewall administration. Another strategy is to create a single powerful administrator role along with an advanced user role. The advanced user role would be for users who are permitted to fix portions of their own systems.

The superuser model and the RBAC model can co-exist. The following table summarizes the gradations from superuser to restricted ordinary user that are possible in the RBAC model. The table includes the administrative actions that can be tracked in both models. For a summary of the effect of privileges alone on a system, see Table 8–2.
### TABLE 8–1 Superuser Model Versus RBAC With Privileges Model

<table>
<thead>
<tr>
<th>User Capabilities on a System</th>
<th>Superuser Model</th>
<th>RBAC Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can become superuser with full superuser capability</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Can log in as a user with full user capabilities</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Can become superuser with limited capabilities</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Can log in as a user, and have superuser capabilities, sporadically</td>
<td>Yes, with setuid programs only</td>
<td>Yes, with setuid programs and with RBAC</td>
</tr>
<tr>
<td>Can log in as a user with administrative capabilities, but without full superuser capability</td>
<td>No</td>
<td>Yes, with RBAC and with directly-assigned privileges and authorizations</td>
</tr>
<tr>
<td>Can log in as a user with fewer capabilities than an ordinary user</td>
<td>No</td>
<td>Yes, with RBAC and with removed privileges</td>
</tr>
<tr>
<td>Can track superuser actions</td>
<td>Yes, by auditing the su command</td>
<td>Yes, by auditing profile shell commands Also, if root user is disabled, the name of the user who has assumed the root role is in the audit trail</td>
</tr>
</tbody>
</table>

---

**Solaris RBAC Elements and Basic Concepts**

The RBAC model in the Solaris OS introduces the following elements:

- **Authorization** – A permission that enables a user or role to perform a class of actions that could affect security. For example, security policy at installation gives ordinary users the `solaris.device.cdrw` authorization. This authorization enables users to read and write to a CD-ROM device. For a list of authorizations, see the `/etc/security/attr` file.

- **Privilege** – A discrete right that can be granted to a command, a user, a role, or a system. Privileges enable a process to succeed. For example, the `proc_exec` privilege allows a process to call `execve()`(). Ordinary users have basic privileges. To see your basic privileges, run the `ppriv -vI basic` command.

- **Security attributes** – An attribute that enables a process to perform an operation. In a typical UNIX environment, a security attribute enables a process to perform an operation that is otherwise forbidden to ordinary users. For example, `setuid` and `setgid` programs have security attributes. In the RBAC model, operations that ordinary users perform might require security attributes. In addition to `setuid` and `setgid` programs, authorizations and privileges are also security attributes in the RBAC model. For example, a user with the
solaris.device.allocate authorization can allocate a device for exclusive use. A process with the sys_time privilege can manipulate system time.

- **Privileged application** – An application or command that can override system controls by checking for *security attributes*. In a typical UNIX environment and in the RBAC model, programs that use setuid and setgid are privileged applications. In the RBAC model, programs that require privileges or authorizations to succeed are also privileged applications. For more information, see “Privileged Applications and RBAC” on page 160.

- **Rights profile** – A collection of administrative capabilities that can be assigned to a role or to a user. A rights profile can consist of authorizations, of commands with security attributes, and of other rights profiles. Rights profiles offer a convenient way to group security attributes.

- **Role** – A special identity for running privileged applications. The special identity can be assumed by assigned users only. In a system that is run by roles, superuser is unnecessary. Superuser capabilities are distributed to different roles. For example, in a two-role system, security tasks would be handled by a security role. The second role would handle system administration tasks that are not security-related. Roles can be more fine-grained. For example, a system could include separate administrative roles for handling the cryptographic framework, printers, system time, file systems, and auditing.

The following figure shows how the RBAC elements work together.

![Diagram of RBAC elements](https://example.com/solaris-rbac-diagram.png)

**FIGURE 8–1 Solaris RBAC Element Relationships**

In RBAC, roles are assigned to users. When a user assumes a role, the capabilities of the role are available. Roles get their capabilities from rights profiles. Rights profiles can contain authorizations, privileged commands, and other supplementary rights profiles. Privileged commands are commands that execute with security attributes.
The following figure uses the Operator role, the Operator rights profile, and the Printer Management rights profile to demonstrate RBAC relationships.

![Diagram showing the relationship between users, roles, rights profiles, and supplementary rights profiles.

The Operator role is used to maintain printers and to perform media backup. The role is assigned to the user jdoe. jdoe can assume the role by switching to the role, and then supplying the role password.

The Operator rights profile has been assigned to the Operator role. The Operator rights profile contains two supplementary profiles, Printer Management and Media Backup. The supplementary profiles reflect the role’s primary tasks.

The Printer Management rights profile is for managing printers, print daemons, and spoolers. Three authorizations are included in the Printer Management rights profile:

- `solaris.admin.printer.read`
- `solaris.admin.printer.modify`
- `solaris.admin.printer.delete`
solaris.admin.printer.read, solaris.admin.printer.delete, and solaris.admin.printer.modify. These authorizations enable roles and users to manipulate information in the printer queue. The Printer Management rights profile also includes a number of commands with security attributes, such as /usr/sbin/lpshut with euid=lp and /usr/ucb/lpq with euid=0.

RBAC Authorizations

An authorization is a discrete right that can be granted to a role or to a user. Authorizations enforce policy at the user application level. Authorizations can be assigned directly to a role or to a user. Typically, authorizations are included in a rights profile. The rights profile is then included in a role, and the role is assigned to a user. For an example, see Figure 8–2.

RBAC-compliant applications can check a user’s authorizations prior to granting access to the application or specific operations within the application. This check replaces the check in conventional UNIX applications for UID=0. For more information on authorizations, see the following sections:

- “Authorization Naming and Delegation” on page 210
- “auth_attr Database” on page 214
- “Commands That Require Authorizations” on page 219

Authorizations and Privileges

Privileges enforce security policy in the kernel. The difference between authorizations and privileges concerns the level at which the security policy is enforced. Without the proper privilege, a process can be prevented from performing privileged operations by the kernel. Without the proper authorizations, a user might be prevented from using a privileged application or from performing security-sensitive operations within a privileged application. For a fuller discussion of privileges, see “Privileges (Overview)” on page 164.

Privileged Applications and RBAC

Applications and commands that can override system controls are considered privileged applications. Security attributes such as UID=0, privileges, and authorizations make an application privileged.

Applications That Check UIDs and GIDs

Privileged applications that check for root (UID=0) or some other special UID or GID have long existed in the UNIX environment. The rights profile mechanism enables you to isolate commands that require a specific ID. Instead of changing the ID on a command that anyone can access, you can place the command with execution security attributes in a rights profile. A user or role with that rights profile can then run the program without having to become superuser.
IDs can be specified as real or effective. Assigning effective IDs is preferred over assigning real IDs. Effective IDs are equivalent to the setuid feature in the file permission bits. Effective IDs also identify the UID for auditing. However, because some shell scripts and programs require a real UID of root, real UIDs can be set as well. For example, the pkgadd command requires a real rather than an effective UID. If an effective ID is not sufficient to run a command, you need to change the ID to a real ID. For the procedure, see “How to Create or Change a Rights Profile” on page 196.

Applications That Check for Privileges

Privileged applications can check for the use of privileges. The RBAC rights profile mechanism enables you to specify the privileges for specific commands. Instead of requiring superuser capabilities to use an application or command, you can isolate the command with execution security attributes in a rights profile. A user or role with that rights profile can then run the command with just the privileges that the command requires to succeed.

Commands that check for privileges include the following:

- Kerberos commands, such as kadmin, kprop, and kdb5_util
- Network commands, such as ifconfig, routeadm, and snoop
- File and file system commands, such as chmod, chgrp, and mount
- Commands that control processes, such as kill, pcred, and rcapadm

To add commands with privileges to a rights profile, see “How to Create or Change a Rights Profile” on page 196. To determine what commands check for privileges in a particular profile, see “Determining Your Assigned Privileges” on page 230.

Applications That Check Authorizations

The Solaris OS additionally provides commands that check authorizations. By definition, the root user has all authorizations. Therefore, the root user can run any application. Applications that check for authorizations include the following:

- The entire Solaris Management Console suite of tools
- Audit administration commands, such as auditconfig and auditreduce
- Printer administration commands, such as lpadmin and lpfilter
- The batch job-related commands, such as at, atq, batch, and crontab
- Device-oriented commands, such as allocate, deallocate, list_devices, and cdrw.

To test a script or program for authorizations, see Example 9–25. To write a program that requires authorizations, see “About Authorizations” in Solaris Security for Developers Guide.
RBAC Rights Profiles

A rights profile is a collection of system overrides that can be assigned to a role or user. A rights profile can include authorizations, commands with assigned security attributes, and other rights profiles. Rights profile information is split between the prof_attr and exec_attr databases. The rights profile name and authorizations are in the prof_attr database. The rights profile name and the commands with assigned security attributes are in the exec_attr database.

For more information on rights profiles, see the following sections:

- “Contents of Rights Profiles” on page 205
- "prof_attr Database” on page 215
- “exec_attr Database” on page 216

RBAC Roles

A role is a special type of user account from which you can run privileged applications. Roles are created in the same general manner as user accounts. Roles have a home directory, a group assignment, a password, and so on. Rights profiles and authorizations give the role administrative capabilities. Roles cannot inherit capabilities from other roles or other users. Discrete roles parcel out superuser capabilities, and thus enable more secure administrative practices.

When a user assumes a role, the role’s attributes replace all user attributes. Role information is stored in the passwd, shadow, and user_attr databases. Role information can be added to the audit_user database. For detailed information on setting up roles, see the following sections:

- “How to Plan Your RBAC Implementation” on page 175
- “How to Create a Role From the Command Line” on page 180
- “How to Change the Properties of a Role” on page 195

A role can be assigned to more than one user. All users who can assume the same role have the same role home directory, operate in the same environment, and have access to the same files. Users can assume roles from the command line by running the su command and supplying the role name and password. Users can also assume a role in the Solaris Management Console tool.

A role cannot log in directly. A user logs in, and then assumes a role. Having assumed a role, the user cannot assume another role without first exiting their current role. Having exited the role, the user can then assume another role.

You can prevent anonymous root login by changing the root user into a role, as shown in “How to Make root User Into a Role” on page 184. If the profile shell command, pfexec, is being audited, the audit trail contains the login user’s real UID, the roles that the user has assumed, and the actions that the role performed. To audit the system or a particular user for role operations, see “How to Audit Roles” on page 184.
No predefined roles are shipped with Solaris software. However, the rights profiles that ship with the software are designed to map to roles. For example, the Primary Administrator rights profile can be used to create the Primary Administrator role.

- To configure the Primary Administrator role, see "Using the Solaris Management Tools With RBAC (Task Map)" in System Administration Guide: Basic Administration.
- To configure other roles, see “How to Create and Assign a Role by Using the GUI” on page 177.
- To create roles on the command line, see “Managing RBAC (Task Map)” on page 192.

Profile Shell in RBAC

Roles can run privileged applications from the Solaris Management Console launcher or from a profile shell. A profile shell is a special shell that recognizes the security attributes that are included in a rights profile. Profile shells are launched when the user runs the su command to assume a role. The profile shells are pfsh, pfcsh, and pfksh. The shells correspond to Bourne shell (sh), C shell (csh), and Korn shell (ksh), respectively.

Users who have been directly assigned a rights profile must invoke a profile shell to run the commands with security attributes. For usability and security considerations, see “Security Considerations When Directly Assigning Security Attributes” on page 163.

All commands that are executed in a profile shell can be audited. For more information, see “How to Audit Roles” on page 184.

Name Service Scope and RBAC

Name service scope is an important concept for understanding RBAC. The scope of a role might be limited to an individual host. Alternatively, the scope might include all hosts that are served by a name service such as NIS, NIS+, or LDAP. The name service scope for a system is specified in the file /etc/nsswitch.conf. A lookup stops at the first match. For example, if a rights profile exists in two name service scopes, only the entries in the first name service scope are used. If files is the first match, then the scope of the role is limited to the local host.

Security Considerations When Directly Assigning Security Attributes

Typically, a user obtains administrative capabilities through a role. Authorizations and privileged commands are grouped into a rights profile. The rights profile is included in a role, and the role is assigned to a user.
Direct assignment of rights profiles and security attributes is also possible:

- Rights profiles, privileges, and authorizations can be assigned directly to users.
- Privileges and authorizations can be assigned directly to roles.

However, direct assignment is not a secure practice. Users and roles with a directly assigned privilege could override security policy wherever this privilege is required by the kernel. When a privilege is a security attribute of a command in a rights profile, that privilege is available only for that command by someone who has that rights profile. The privilege is not available for other commands that the user or role might run.

Since authorizations act at the user level, direct assignment of authorizations can be less dangerous than direct assignment of privileges. However, authorizations can enable a user to perform highly secure tasks, such as delegate device administration.

A rights profile that is assigned directly to a user presents usability problems more than security problems. The commands with security attributes in the rights profile can only succeed in a profile shell. The user must open a profile shell, then type the commands. A role that is assigned a rights profile gets a profile shell automatically. Therefore, the commands succeed in the role’s shell.

Rights profiles provide an extensible, clean way to group security characteristics for particular administrative tasks.

**Privileges (Overview)**

Process rights management enables processes to be restricted at the command, user, role, or system level. The Solaris OS implements process rights management through privileges. Privileges decrease the security risk that is associated with one user or one process having full superuser capabilities on a system. Privileges and RBAC provide a compelling alternative model to the traditional superuser model.

- For information on RBAC, see “Role-Based Access Control (Overview)” on page 155.
- For information on how to administer privileges, see Chapter 11, “Privileges (Tasks).”
- For reference information on privileges, see Chapter 12, “Privileges (Reference).”

**Privileges Protect Kernel Processes**

A privilege is a discrete right that a process requires to perform an operation. The right is enforced in the kernel. A program that operates within the bounds of the Solaris basic set of privileges operates within the bounds of the system security policy. setuid programs are examples of programs that operate outside the bounds of the system security policy. By using privileges, programs eliminate the need for calls to setuid.
Privileges discretely enumerate the kinds of operations that are possible on a system. Programs can be run with the exact privileges that enable the program to succeed. For example, a program that sets the date and writes the date to an administrative file might require the file_dac_write and sys_time privileges. This capability eliminates the need to run any program as root.

Historically, systems have not followed the privilege model. Rather, systems used the superuser model. In the superuser model, processes run as root or as a user. User processes were limited to acting on the user's directories and files. root processes could create directories and files anywhere on the system. A process that required creation of a directory outside the user's directory would run with a UID=0, that is, as root. Security policy relied on DAC, discretionary access control, to protect system files. Device nodes were protected by DAC. For example, devices owned by group sys could be opened only by members of group sys.

However, setuid programs, file permissions, and administrative accounts are vulnerable to misuse. The actions that a setuid process is permitted are more numerous than the process requires to complete its operation. A setuid program can be compromised by an intruder who then runs as the all-powerful root user. Similarly, any user with access to the root password can compromise the entire system.

In contrast, a system that enforces policy with privileges allows a gradation between user capabilities and root capabilities. A user can be granted privileges to perform activities that are beyond the capabilities of ordinary users, and root can be limited to fewer privileges than root currently possesses. With RBAC, a command that runs with privileges can be isolated in a rights profile and assigned to one user or role. Table 8–1 summarizes the gradation between user capabilities and root capabilities that the RBAC plus privileges model provides.

The privilege model provides greater security than the superuser model. Privileges that have been removed from a process cannot be exploited. Process privileges prevent a program or administrative account from gaining access to all capabilities. Process privileges can provide an additional safeguard for sensitive files, where DAC protections alone can be exploited to gain access.

Privileges, then, can restrict programs and processes to just the capabilities that the program requires. This capability is called the principle of least privilege. On a system that implements least privilege, an intruder who captures a process has access to only those privileges that the process has. The rest of the system cannot be compromised.

**Privilege Descriptions**

Privileges are logically grouped on the basis of the area of the privilege.

- **FILE privileges** – Privileges that begin with the string file operate on file system objects. For example, the file_dac_write privilege overrides discretionary access control when writing to files.
IPC privileges – Privileges that begin with the string `ipc` override IPC object access controls. For example, the `ipc_dac_read` privilege enables a process to read remote shared memory that is protected by DAC.

NET privileges – Privileges that begin with the string `net` give access to specific network functionality. For example, the `net_rawaccess` privilege enables a device to connect to the network.

PROC privileges – Privileges that begin with the string `proc` allow processes to modify restricted properties of the process itself. PROC privileges include privileges that have a very limited effect. For example, the `proc_clock_highres` privilege enables a process to use high resolution timers.

SYS privileges – Privileges that begin with the string `sys` give processes unrestricted access to various system properties. For example, the `sys_linkdir` privilege enables a process to make and break hard links to directories.

Some privileges have a limited effect on the system, and some have a broad effect. The definition of the `proc_taskid` privilege indicates its limited effect:

```
proc_taskid
   Allows a process to assign a new task ID to the calling process.
```

The definition of the `file_setid` privilege indicates its broad effect:

```
net_rawaccess
   Allow a process to have direct access to the network layer.
```

The `privileges(5)` man page provides descriptions of every privilege. The command `ppriv -lv` prints a description of every privilege to standard out.

**Administrative Differences on a System With Privileges**

A system that has privileges has several visible differences from a system that does not have privileges. The following table lists some of the differences.

**TABLE 8–2 Visible Differences Between a System With Privileges and a System Without Privileges**

<table>
<thead>
<tr>
<th>Feature</th>
<th>No Privileges</th>
<th>Privileges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Daemons</td>
<td>Daemons run as root.</td>
<td>Daemons run as the user daemon.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For example, the following daemons have been</td>
</tr>
<tr>
<td></td>
<td></td>
<td>assigned appropriate privileges and run as</td>
</tr>
<tr>
<td></td>
<td></td>
<td>daemon: <code>lockd</code>, <code>nfsd</code>, and <code>rpcbind</code>.</td>
</tr>
<tr>
<td>Log File</td>
<td>Log files are owned by root.</td>
<td>Log files are now owned by daemon, who created</td>
</tr>
<tr>
<td>Ownership</td>
<td></td>
<td>the log file. The root user does not own the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>file.</td>
</tr>
</tbody>
</table>
### Visible Differences Between a System With Privileges and a System Without Privileges (Continued)

<table>
<thead>
<tr>
<th>Feature</th>
<th>No Privileges</th>
<th>Privileges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error Messages</td>
<td>Error messages refer to superuser.</td>
<td>Error messages reflect the use of privileges.</td>
</tr>
<tr>
<td></td>
<td>For example, chroot: not superuser.</td>
<td>For example, the equivalent error message for chroot failure is chroot: exec failed.</td>
</tr>
<tr>
<td>setuid Programs</td>
<td>Programs use setuid to complete tasks that ordinary users are not allowed to perform.</td>
<td>Many setuid programs have been changed to run with privileges. For example, the following utilities use privileges: rsh, rlogin, rcp, rdist, ping, traceroute, and newtask.</td>
</tr>
<tr>
<td>File Permissions</td>
<td>Device permissions are controlled by DAC.</td>
<td>File permissions (DAC) do not predict who can open a device. Devices are protected with DAC and device policy. For example, the /dev/ip file has 666 permissions, but the device can only be opened by a process with the appropriate privileges. Raw sockets are still protected by DAC.</td>
</tr>
<tr>
<td>Audit Events</td>
<td>Auditing the use of the su command covers many administrative functions.</td>
<td>Auditing the use of privileges covers most administrative functions. The pm and as audit classes include audit events that configure device policy and audit events that set privileges.</td>
</tr>
<tr>
<td>Processes</td>
<td>Processes are protected by who owns the process.</td>
<td>Processes are protected by privileges. Process privileges and process flags are visible as a new entry in the /proc/&lt;pid&gt; directory, priv. The ELF note section of core dumps includes information about process privileges and flags in the NT_PRPRIV and NT_PRPRIVINFO notes.</td>
</tr>
<tr>
<td>Debugging</td>
<td>No reference to privileges in core dumps.</td>
<td>The ppriv utility and other utilities show the proper number of properly sized sets. The utilities correctly map the bits in the bit sets to privilege names.</td>
</tr>
</tbody>
</table>

### Privileges and System Resources

In the Solaris Express Community Edition, the `project.max-locked-memory` and `zone.max-locked-memory` resource controls can be used to limit the memory consumption of processes that are assigned the PRIV_PROC_LOCK_MEMORY privilege. This privilege allows a process to lock pages in physical memory.

If you assign the PRIV_PROC_LOCK_MEMORY privilege to a rights profile, you can give the processes that have this privilege the ability to lock all memory. As a safeguard, set a resource control to prevent the user of the privilege from locking all memory. For privileged processes that run in a non-global zone, set the `zone.max-locked-memory` resource control. For privileged processes that run on a system, create a project and set the `project.max-locked-memory` resource control. For information about these resource controls, see Chapter 6, "Resource Controls (Overview)," in *System Administration Guide: Virtualization*.
How Privileges Are Implemented

Every process has four sets of privileges that determine whether a process can use a particular privilege. The kernel automatically calculates the effective set of privileges. You can modify the initial inheritable set of privileges. A program that is coded to use privileges can reduce the program’s permitted set of privileges. You can shrink the limit set of privileges.

- **Effective privilege set, or E** – Is the set of privileges that is currently in effect. A process can add privileges that are in the permitted set to the effective set. A process can also remove privileges from E.

- **Permitted privilege set, or P** – Is the set of privileges that is available for use. Privileges can be available to a program from inheritance or through assignment. An execution profile is one way to assign privileges to a program. The setuid command assigns all privileges that root has to a program. Privileges can be removed from the permitted set, but privileges cannot be added to the set. Privileges that are removed from P are automatically removed from E.

  A privilege-aware program removes the privileges that a program never uses from the program’s permitted set. In this way, unnecessary privileges cannot be exploited by the program or a malicious process. For more information on privilege-aware programs, see Chapter 2, “Developing Privileged Applications,” in Solaris Security for Developers Guide.

- **Inheritable privilege set, or I** – Is the set of privileges that a process can inherit across a call to exec. After the call to exec, the permitted and the effective sets are equal, except in the special case of a setuid program.

  For a setuid program, after the call to exec, the inheritable set is first restricted by the limit set. Then, the set of privileges that were inherited (I), minus any privileges that were in the limit set (L), are assigned to P and E for that process.

- **Limit privilege set, or L** – Is the outside limit of what privileges are available to a process and its children. By default, the limit set is all privileges. Processes can shrink the limit set but can never extend the limit set. L is used to restrict I. Consequently, L restricts P and E at the time of exec.

  If a user has been assigned a profile that includes a program that has been assigned privileges, the user can usually run that program. On an unmodified system, the program’s assigned privileges are within the user’s limit set. The privileges that have been assigned to the program become part of the user’s permitted set. To run the program that has been assigned privileges, the user must run the program from a profile shell.

The kernel recognizes a basic privilege set. On an unmodified system, each user’s initial inheritable set equals the basic set at login. You can modify the user’s initial inheritable set. You cannot modify the basic set.
On an unmodified system, a user's privilege sets at login would appear similar to the following:

E (Effective): basic
I (Inheritable): basic
P (Permitted): basic
L (Limit): all

Therefore, at login, all users have the basic set in their inheritable set, their permitted set, and their effective set. A user’s limit set contains all privileges. To put more privileges in the user’s effective set, you must assign a rights profile to the user. The rights profile would include commands to which you have added privileges. You can also assign privileges directly to the user or role, though such privilege assignment can be risky. For a discussion of the risks, see “Security Considerations When Directly Assigning Security Attributes” on page 163.

How Processes Get Privileges

Processes can inherit privileges. Or, processes can be assigned privileges. A process inherits privileges from its parent process. At login, the user’s initial inheritable set of privileges determines what privileges are available to the user’s processes. All child processes of the user’s initial login inherit that set.

You can also directly assign privileges to programs, users, and roles. When a program requires privileges, you assign the privileges to the program’s executable in a rights profile. Users or roles that are permitted to run the program are assigned the profile that includes the program. At login or when a profile shell is entered, the program runs with privilege when the program’s executable is typed in the profile shell. For example, a role that includes the Object Access Management profile is able to run the chmod command with the file_chown privilege.

When a role or user runs a program that has been directly assigned an additional privilege, the assigned privilege is added to the role or user’s inheritable set. Child processes of the program that was assigned privileges inherit the privileges of the parent. If the child process requires more privileges than the parent process, the child process must be directly assigned those privileges.

Programs that are coded to use privileges are called privilege-aware programs. A privilege-aware program turns on the use of privilege and turns off the use of privilege during program execution. To succeed in a production environment, the program must be assigned the privileges that the program turns on and off.

For examples of privilege-aware code, see Chapter 2, “Developing Privileged Applications,” in Solaris Security for Developers Guide. To assign privileges to a program that requires privileges, see “How to Add Privileges to a Command” on page 225.
Assigning Privileges

You, in your capacity as system administrator, are responsible for assigning privileges. Typically, you assign the privilege to a command in a rights profile. The rights profile is then assigned to a role or to a user. The Solaris Management Console provides the graphical user interface (GUI) to assign privileges. Privileges can also be assigned by using commands such as `smuser` and `smrole`. For more information on how to use the GUI to assign privileges, see Chapter 9, "Using Role-Based Access Control (Tasks)."

Privileges can also be assigned directly to a user. If you trust a subset of users to use a privilege responsibly throughout their sessions, you can assign the privilege directly. Good candidates for direct assignment are privileges that have a limited effect, such as `proc_clock_highres`. Poor candidates for direct assignment are privileges that have far-reaching effects, such as `file_dac_write`.

Privileges can also be denied to a user or to a system. Care must be taken when removing privileges from the initial inheritable set or the limit set of a user or a system.

Expanding a User or Role's Privileges

Users and roles have an inheritable set of privileges, and a limit set of privileges. The limit set cannot be expanded, since the limit set is initially all privileges. The initial inheritable set can be expanded for users, roles, and systems. A privilege that is not in the inheritable set can also be assigned to a process.

The assignment of privileges per process is the most precise way to add privileges. You can expand the number of privileged operations that a user can perform by enabling the user to assume a role. The role would be assigned profiles that include commands with added privileges. When the user assumes the role, the user gets the role's profile shell. By typing in the role's shell, the commands in the role's profiles execute with the added privileges.

You can also assign a profile to the user rather than to a role that the user assumes. The profile would include commands with added privileges. When the user opens a profile shell, such as `pfksh`, the user can execute the commands in the user's profile with privilege. In a regular shell, the commands do not execute with privilege. The privileged process can only execute in a privileged shell.

To expand the initial inheritable set of privileges for users, roles, or systems is a riskier way to assign privileges. All privileges in the inheritable set are in the permitted and effective sets. All commands that the user or role types in a shell can use the directly assigned privileges. Directly assigned privileges enable a user or role to easily perform operations that can be outside the bounds of their administrative responsibilities.

When you add to the initial inheritable set of privileges on a system, all users who log on to the system have a larger set of basic privileges. Such direct assignment enables all users of the system to easily perform operations that are probably outside the bounds of ordinary users.
Restricting a User or Role’s Privileges

By removing privileges, you can prevent users and roles from performing particular tasks. You can remove privileges from the initial inheritable set, and from the limit set. You should carefully test removal of privileges before you distribute an initial inheritable set or a limit set that is smaller than the default set. By removing privileges from the initial inheritable set, you might prevent users from logging in. When privileges are removed from the limit set, a legacy setuid program might fail because the program requires a privilege that was removed.

Assigning Privileges to a Script

Scripts are executables, like commands. Therefore, in a rights profile, you can add privileges to a script just as you can add privileges to a command. The script runs with the added privileges when a user or role who has been assigned the profile executes the script in a profile shell. If the script contains commands that require privileges, the commands with added privileges should also be in the profile.

Privilege-aware programs can restrict privileges per process. Your job with a privilege-aware program is to assign the executable just the privileges that the program needs. You then test the program to see that the program succeeds in performing its tasks. You also check that the program does not abuse its use of privileges.

Privileges and Devices

The privilege model uses privileges to protect system interfaces that are protected by file permissions alone in the superuser model. In a system with privileges, file permissions are too weak to protect the interfaces. A privilege such as proc_owner could override file permissions and then give full access to all of the system.

Therefore, ownership of the device directory is not sufficient to open a device. For example, members of the group sys are no longer automatically allowed to open the /dev/ip device. The file permissions on /dev/ip are 0666, but the net_rawaccess privilege is required to open the device.

Device policy is controlled by privileges. The getdevpolicy command displays the device policy for every device. The device configuration command, devfsadm, installs the device policy. The devfsadm command binds privilege sets with open for reading or writing of devices. For more information, see the getdevpolicy(1M) and devfsadm(1M) man pages.

Device policy allows you more flexibility in granting permission to open devices. You can require different privileges or more privileges than the default device policy. The privilege requirements can be modified for the device policy and for the driver proper. You can modify the privileges when installing, adding, or updating a device driver.

The add_drv and update_drv commands can modify device policy entries and driver-specific privileges. You must be running a process with the full set of privileges to change the device policy. For more information, see the add_drv(1M) and update_drv(1M) man pages.
Privileges and Debugging

The Solaris OS provides tools to debug privilege failure. The `ppriv` command and the `truss` command provide debugging output. For examples, see the `ppriv(1)` man page. For a procedure, see “How to Determine Which Privileges a Program Requires” on page 224.
Using Role-Based Access Control (Tasks)

This chapter covers tasks for distributing the capabilities of superuser by using discrete roles. The mechanisms that roles can use include rights profiles, authorizations, and privileges. The following is a list of the task maps in this chapter.

- “Using RBAC (Task Map)” on page 173
- “Configuring RBAC (Task Map)” on page 174
- “Using Roles (Task Map)” on page 188
- “Managing RBAC (Task Map)” on page 192

For an overview of RBAC, see “Role-Based Access Control (Overview)” on page 155. For reference information, see Chapter 10, “Role-Based Access Control (Reference).” To use privileges with RBAC or without RBAC, see Chapter 11, “Privileges (Tasks).”

Using RBAC (Task Map)

To use RBAC requires planning, configuring RBAC, and knowing how to assume a role. Once roles become familiar, you might further customize RBAC to handle new operations. The following task map points to these major tasks.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plan and configure RBAC</td>
<td>Configure RBAC at your site.</td>
<td>“Configuring RBAC (Task Map)” on page 174</td>
</tr>
<tr>
<td>Use roles</td>
<td>Assume roles from the command line and in the Solaris Management Console GUI.</td>
<td>“Using Roles (Task Map)” on page 188</td>
</tr>
<tr>
<td>Customize RBAC</td>
<td>Customize RBAC for your site.</td>
<td>“Managing RBAC (Task Map)” on page 192</td>
</tr>
</tbody>
</table>
To use RBAC effectively requires planning. Use the following task map to plan and initially implement RBAC at your site.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Plan for RBAC</td>
<td>Involves examining your site’s security needs, and deciding how to use RBAC at your site.</td>
<td>“How to Plan Your RBAC Implementation” on page 175</td>
</tr>
<tr>
<td>2. Learn to use the Solaris Management Console</td>
<td>Involves becoming familiar with the Solaris Management Console.</td>
<td>Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration</td>
</tr>
<tr>
<td>3. Configure the first user and role</td>
<td>Uses the RBAC configuration tools in the Solaris Management Console to create a user and a role, and to assign the role to the user.</td>
<td>“Using the Solaris Management Tools With RBAC (Task Map)” in System Administration Guide: Basic Administration</td>
</tr>
<tr>
<td>4. (Optional) Create other users who can assume roles</td>
<td>Ensures that users who can assume an administrative role exist.</td>
<td>“Using the Solaris Management Tools With RBAC (Task Map)” in System Administration Guide: Basic Administration</td>
</tr>
<tr>
<td>5. (Recommended) Create other roles and assign them to users</td>
<td>Uses the RBAC tools to create roles for particular administrative areas, and to assign the roles to users.</td>
<td>“How to Create and Assign a Role by Using the GUI” on page 177</td>
</tr>
<tr>
<td></td>
<td>Uses the command line to create roles, and to assign the roles to users</td>
<td>Example 9-5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>“How to Create a Role From the Command Line” on page 180</td>
</tr>
<tr>
<td></td>
<td></td>
<td>“How to Assign a Role to a Local User” on page 183</td>
</tr>
<tr>
<td>6. (Recommended) Audit role actions</td>
<td>Preselect an audit class that includes the audit event that records role actions.</td>
<td>“How to Audit Roles” on page 184</td>
</tr>
<tr>
<td>7. (Optional) Make root user a role</td>
<td>Prevents anonymous root login, which is a security hole.</td>
<td>“How to Make root User Into a Role” on page 184</td>
</tr>
</tbody>
</table>
Configuring RBAC

RBAC can be configured with the following utilities:

- **Solaris Management Console GUI** – The preferred method for performing RBAC-related tasks is through the GUI. The console tools for managing the RBAC elements are contained in the Users Tool collection.

- **Solaris Management Console commands** – With the Solaris Management Console command-line interfaces, such as `smprole`, you can operate on any name service. The Solaris Management Console commands require authentication to connect to the server. As a result, these commands are not practical for use in scripts.

- **Local commands** – With the `user*` and `role*` set of command-line interfaces, such as `useradd`, you can operate on local files only. The commands that operate on local files must be run by superuser or by a role with the appropriate privileges.

▼ How to Plan Your RBAC Implementation

RBAC can be an integral part of how an organization manages its information resources. Planning requires a thorough knowledge of the RBAC capabilities as well as the security requirements of your organization.

1. **Learn the basic RBAC concepts.**
   
   Read “Role-Based Access Control (Overview)” on page 155. Using RBAC to administer a system is very different from using conventional UNIX administrative practices. You should be familiar with the RBAC concepts before you start your implementation. For greater detail, see Chapter 10, “Role-Based Access Control (Reference).”

2. **Examine your security policy.**
   
   Your organization’s security policy should detail the potential threats to your system, measure the risk of each threat, and have a strategy to counter these threats. Isolating the security-relevant tasks through RBAC can be a part of the strategy. Although you can install the recommended roles and their configurations as is, you might need to customize your RBAC configuration to adhere to your security policy.

3. **Decide how much RBAC your organization needs.**
Depending on your security needs, you can use varying degrees of RBAC, as follows:

- **No RBAC** – You can perform all tasks as root user. In this configuration, you log in as yourself. Then, you type root as the user when you select a Solaris Management Console tool.

- **Single Role Only** – This method adds one role. The one role is assigned the Primary Administrator rights profile. This method is similar to the superuser model, in that the role has superuser capabilities. However, this method enables you to track the user who has assumed the role.

- **Recommended Roles** – This method creates three roles that are based on the following rights profiles: Primary Administrator, System Administrator, and Operator. The roles are suitable for organizations with administrators at different levels of responsibility.

- **Custom Roles** – You can create your own roles to meet the security requirements of your organization. The new roles can be based on existing or customized rights profiles. To customize rights profiles that enforce separation of duty, see "Creating Roles and Users in Trusted Extensions" in Solaris Trusted Extensions Administrator's Procedures.

- **Root User as a Role** – This method prevents any user from logging in as root. Instead, users must log in as ordinary users prior to assuming the root role. For details, see "How to Make root User Into a Role" on page 184.

4 **Decide which recommended roles are appropriate for your organization.**

Review the capabilities of the recommended roles and default rights profiles. Default rights profiles enable administrators to configure a recommended role by using a single profile.

Three default rights profiles are available for configuring the recommended roles:

- **Primary Administrator rights profile** – For configuring a role that can perform all administrative tasks, can grant rights to others, and can edit rights that are associated with administrative roles. A user in this role can assign this role to other users, and can grant rights to other users.

- **System Administrator rights profile** – For configuring a role that can perform most administrative tasks that are not related to security. For example, the System Administrator can add new user accounts, but cannot set passwords or grant rights to other users.

- **Operator rights profile** – For configuring a role that can perform simple administrative tasks, such as media backup and printer maintenance.

To further examine rights profiles, read one of the following:

- In the /etc/security directory, read the contents of the prof_attr database and the exec_attr database.

- In the Solaris Management Console, use the Rights tool to display the contents of a rights profile.

- In this book, refer to “Contents of Rights Profiles” on page 205 for summaries of some typical rights profiles.
5 Decide if any additional roles or rights profiles are appropriate for your organization.
Look for other applications or families of applications at your site that might benefit from restricted access. Applications that affect security, that can cause denial-of-service problems, or that require special administrator training are good candidates for RBAC. You can customize roles and rights profiles to handle the security requirements of your organization.

a. Determine which commands are needed for the new task.

b. Decide which rights profile is appropriate for this task.
Check if an existing rights profile can handle this task or if a separate rights profile needs to be created.

c. Determine which role is appropriate for this rights profile.
Decide if the rights profile for this task should be assigned to an existing role or if a new role should be created. If you use an existing role, check that the other rights profiles are appropriate for users who are assigned to this role.

6 Decide which users should be assigned to the available roles.
According to the principle of least privilege, you should assign users to roles that are appropriate to their level of trust. When you prevent users from access to tasks that the users do not need to perform, you reduce potential problems.

▼ How to Create and Assign a Role by Using the GUI

To create a new role, you can be superuser, or you can use the Primary Administrator role. In this procedure, the creator of the new role has assumed the role of Primary Administrator.

Before You Begin

- You have already created users who can assume a role at your site. If the users are not yet created, create them by following the instructions in “How to Add a User With the Solaris Management Console’s Users Tool” in System Administration Guide: Basic Administration “Using the Solaris Management Tools With RBAC (Task Map)” in System Administration Guide: Basic Administration.

- You have been assigned the Primary Administrator role by following the procedures in “Using the Solaris Management Tools With RBAC (Task Map)” in System Administration Guide: Basic Administration.

1 Start the Solaris Management Console.
# /usr/sbin/smc &
For login instructions, see “How to Assume a Role in the Solaris Management Console” on page 190.

2 Click the Administrative Roles icon.
3 Select Add Administrative Role from the Action menu.

4 Create a new role by filling in the fields in the series of dialog boxes.
   For possible roles, see Example 9–1 to Example 9–4.

   Tip – All tools in the Solaris Management Console display information in the bottom section of the page or at the left side of a wizard panel. Choose Help at any time to find additional information about performing tasks in this interface.

5 Assign the role to a user.

   Tip – After filling in the properties of the role, the last dialog box prompts you for a user for the role.

6 In a terminal window, restart the name service cache daemon.
   
   `# svcadm restart system/name-service-cache`

   For more information, see the `svcadm(1M)` and `nscd(1M)` man pages.

Example 9–1 Creating a Role for the System Administrator Rights Profile

In this example, the new role can do system administration tasks that are not connected to security. The role is created by performing the preceding procedure with the following parameters:

- Role name: sysadmin
- Role full name: System Administrator
- Role description: Performs non-security admin tasks
- Rights profile: System Administrator
  
  This rights profile is at the top of the list of profiles that are included in the role.

Example 9–2 Creating a Role for the Operator Rights Profile

The Operator rights profile can manage printers and back up the system to offline media. You might want to assign the role to one user on each shift. To do so, you would select the role mailing list option in the Step 1: Enter a Role Name dialog box. The role is created by performing the preceding procedure with the following parameters:

- Role name: operadm
- Role full name: Operator
- Role description: Backup operator
- Rights profile: Operator
This rights profile must be at the top of the list of profiles that are included in the role.

**Example 9–3  Creating a Role for a Security-Related Rights Profile**

By default, the only rights profile that contains security-related commands and rights is the Primary Administrator profile. If you want to create a role that is not as powerful as Primary Administrator, but can handle some security-related tasks, you must create the role.

In the following example, the role protects devices. The role is created by performing the preceding procedure with the following parameters:

- Role name: deviceSec
- Role full name: Device Security
- Role description: Configures Devices
- Rights profile: Device Security

In the following example, the role secures systems and hosts on the network. The role is created by performing the preceding procedure with the following parameters:

- Role name: netsec
- Role full name: Network Security
- Role description: Handles IPsec, IKE, and SSH
- Rights profile: Network Security

**Example 9–4  Creating a Role for a Rights Profile With Limited Scope**

A number of rights profiles are of limited scope. In this example, the sole task of the role is to manage DHCP. The role is created by performing the preceding procedure with the following parameters:

- Role name: dhcpmgmt
- Role full name: DHCP Management
- Role description: Manages Dynamic Host Config Protocol
- Rights profile: DHCP Management

**Example 9–5  Modifying a User’s Role Assignment**

In this example, a role is added to an existing user. The user’s role assignment is modified by clicking the User Accounts icon in the Users tool in the Solaris Management Console, double-clicking the user, and following the online help to add a role to the user’s capabilities.

**Troubleshooting**

Check the following if the role does not have the capabilities that it should:

- Are the role’s rights profiles listed in the GUI from most to least powerful?

For example, if the All rights profile is at the top of the list, then no commands are run with security attributes. A profile that contains commands with security attributes must precede the All rights profile in the list.
Do the commands in the role's rights profiles have the appropriate security attributes?
For example, when the policy is \texttt{suser}, some commands require \texttt{uid=0} rather than \texttt{euid=0}.

Is the rights profile defined in the appropriate name service scope? Is the role operating in the name service scope where the rights profile is defined?

Has the name service cache, \texttt{svc:/system/name-service-cache}, been restarted?
The \texttt{nscd} daemon can have a lengthy time-to-live interval. By restarting the daemon, you update the name service with current data.

\section*{How to Create a Role From the Command Line}
The Solaris Management Console GUI is the preferred method for managing RBAC. To use the GUI, see \textit{“How to Create and Assign a Role by Using the GUI” on page 177}. You can also use the command-line interfaces, as described in this procedure.

\textbf{Note} – Do not attempt to administer RBAC with the command line and the graphical user interface at the same time. Conflicting changes could be made to the configuration, and the behavior would be unpredictable. You can use both tools to administer RBAC, but you cannot use both concurrently.

\section*{Before You Begin}
To create a role, you must either assume a role that includes the Primary Administrator rights profile, or switch to the user \texttt{root}.

1 \textbf{Assume the Primary Administrator role, or become superuser.}
The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see \textit{Chapter 2, “Working With the Solaris Management Console (Tasks),”} in \textit{System Administration Guide: Basic Administration}.

2 \textbf{Choose one of the following commands to create a role on the command line.}

- For roles in the local name service scope, \textbf{use the \texttt{roleadd} command.}

\textbf{Note} – The \texttt{roleadd} command is more limited than the Solaris Management Console GUI or command-line interfaces. After running the \texttt{roleadd} command, you must run the \texttt{usermod} command to assign the role to a user. And, the user then must set the password for the role, as shown in \textit{“How to Assign a Role to a Local User” on page 183}.

\begin{verbatim}
# roleadd -c comment \
- g group - m homedir - u UID - s shell \
- P profile rolename \
- c comment        Is a comment that describes rolename.
\end{verbatim}
Use the smrole add command.

This command creates a role in a distributed name service, such as NIS, NIS+, or LDAP. This command runs as a client of the Solaris Management Console server.

```
$ /usr/sadm/bin/smrole -D domain-name \
  -r admin-role -l <Type admin-role password> \
  add -- -n rolename -a rolename -d directory\ 
  -F full-description -p profile
```

- `-D domain-name` Is the name of the domain that you want to manage.
- `-r admin-role` Is the name of the administrative role that can modify the role. The administrative role must have the `solaris.role.assign` authorization. If you are modifying a role that you have assumed, the role must have the `solaris.role.delegate` authorization.
- `-l` Is the prompt for the password of `admin-role`.
- `--` Is the required separator between authentication options and subcommand options.
- `-n rolename` Is the name of the new role.
- `-c comment` Is the comment that describes the capabilities of the role.
- `-a username` Is the name of the user who can assume `rolename`.
- `-d directory` Is the home directory for `rolename`.
- `-F full-description` Is the full description for `rolename`. This description is displayed in the Solaris Management Console GUI.
- `-p profile` Is a rights profile that is included in the capabilities of `rolename`. This option gives commands with administrative capabilities to the role. You can specify multiple `-p profile` options.

To put the changes into effect, see “How to Assign a Role to a Local User” on page 183.
Creating a Custom Operator Role by Using the \texttt{smrole} Command

The \texttt{smrole} command specifies a new role and its attributes in a name service. In the following example, the Primary Administrator creates a new version of the Operator role. The role includes the standard Operator rights profile as well as the Media Restore rights profile. Note that the command prompts you for a password for the new role.

\begin{verbatim}
% su - primaryadm
Password: <Type primaryadm password>
$ /usr/sadm/bin/smrole add -H myHost -c "Backup and Restore Operator" \
-a operadm2 -a janedoe -d /export/home/operadm \
-F "Backup/Restore Operator" -p "Operator" -p "Media Restore"
Authenticating as user: primaryadm

Type /? for help, pressing <enter> accepts the default denoted by [ ]
Please enter a string value for: password :: <Type primaryadm password>

Loading Tool: com.sun.admin.usermgr.cli.role.UserMgrRoleCli from myHost
Login to myHost as user primaryadm was successful.
Download of com.sun.admin.usermgr.cli.role.UserMgrRoleCli from myHost was successful.

Type /? for help, pressing <enter> accepts the default denoted by [ ]
Please enter a string value for: password :: <Type operadm2 password>

$ svcadm restart system/name-service-cache
\end{verbatim}

The \texttt{smrole} command with the \texttt{list} subcommand is used to display the new role:

\begin{verbatim}
$ /usr/sadm/bin/smrole list --
Authenticating as user: primaryadm

Type /? for help, pressing <enter> accepts the default denoted by [ ]
Please enter a string value for: password :: <Type primaryadm password>

Loading Tool: com.sun.admin.usermgr.cli.role.UserMgrRoleCli from myHost
Login to myHost as user primaryadm was successful.
Download of com.sun.admin.usermgr.cli.role.UserMgrRoleCli from myHost was successful.

root 0 Superuser
primaryadm 100 Most powerful role
sysadmin 101 Performs non-security admin tasks
operadm 102 Backup Operator
operadm2 103 Backup/Restore Operator
\end{verbatim}

\textbf{Example 9–6} Configuring RBAC
How to Assign a Role to a Local User

This procedure assigns a local role to a local user, restarts the name cache daemon, and then shows how the user can assume the role.

To assign a role to a user in a distributed name service, see “How to Create a Role From the Command Line” on page 180 and “How to Change the Properties of a Role” on page 195.

Before You Begin

You have added a local role, as described in “How to Create a Role From the Command Line” on page 180. You must have assumed the role of Primary Administrator or have switched to superuser.

1 Assign the role to a local user.

If you added a local role with the roleadd command, this step is required. This step is optional when you use the smrole command and the Solaris Management Console to create a role.

# usermod -u UID -R rolename login-name

- u UID Is the UID of the user.
- R rolename Is the role that is being assigned to the user.
- login-name Is the user’s login name.

2 To put the changes into effect, restart the name service cache daemon.

# svcadm restart system/name-service-cache

If you added a role with a Solaris Management Console interface, go to “Using Roles (Task Map)” on page 188. Otherwise, continue with the next step.

3 (Optional) To unlock the role account, the user must create a password.

If you added a local role with the roleadd command, this step is required.

% su - rolename
Password: <Type rolename password>
Confirm Password: <Retype rolename password>
$

Example 9–7 Creating and Assigning a Local Role From the Command Line

In this example, a role is created to administer the Solaris Cryptographic Framework. The Crypto Management rights profile contains the cryptoadm command for administering hardware and software cryptographic services on a local system.

# roleadd -c "Cryptographic Services manager" \
- g 14 -m /export/home/cryptoadm -u 104 -s pfksh \

-P "Crypto Management" cryptomgt
# usermod -u 1111 -R cryptomgt
# svcadm restart system/name-service-cache
% su - cryptomgt
Password: <Type cryptomgt password>
Confirm Password: <Retype cryptomgt password>
$ /usr/ucb/whoami
cryptomgt
$

For information about the Solaris Cryptographic Framework, see Chapter 13, “Solaris Cryptographic Framework (Overview).” To administer the framework, see “Administering the Cryptographic Framework (Task Map)” on page 263.

How to Audit Roles

The actions that a role performs can be audited. Included in the audit record is the login name of the user who assumed the role, the role name, and the action that the role performed. The 6180:AUE_prof_cmd:profile command:ua, as audit event collects the information. By preselecting the as class or the ua class, you can audit role actions.

1 Plan for auditing and edit the audit configuration files.
For more information, see “Solaris Auditing (Task Map)” on page 575.

2 Include the ua class or the as class in the flags line of the audit_control file.

```bash
## audit_control file
flags:lo,as
naflags:lo
plugin:name=audit_binfile.so; p_dir=/var/audit

The ua class and the as class include other audit events. To see the audit events that are included in a class, read the audit_event file. You can also use the auditrecord command, as shown in Example 30–24.

3 Finish configuring the auditing service, then enable auditing.
For more information, see “Configuring and Enabling the Audit Service (Tasks)” on page 587.

How to Make root User Into a Role

This procedure shows how to change root from a login user to a role. When you complete this procedure, you can no longer directly log in to the system as root, except in single-user mode. You must be assigned the root role and su to root.
By changing the root user into a role, you prevent anonymous root login. Because a user must log in and then assume the root role, the user’s login ID is provided to the auditing service and is in the su-log file.

In this procedure, you create a local user and assign the root role to the user. To prevent users from assuming the role, see Example 9–8.

1 **As a regular user, log in to the target system.**

2 **Assume the Primary Administrator role, or become superuser.**
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see “Using the Solaris Management Tools With RBAC (Task Map)” in System Administration Guide: Basic Administration.

3 **Create a local user who can assume the root role.**
   For safety, at least one local user should be assigned the root role.
   
   ```
   $ useradd -c comment -u uid -d homedir username
   
   -c comment     Is the comment that describes the user.
   -d homedir     Is the home directory of the user. This directory should be on the local system.
   -u uid         Is the user identification number.
   username       Is the name of the new local user.
   
   # useradd -c "JDoe’s local account" -u 123 -d /export/home1 jdoe-local
   ```

4 **Give the user a password.**
   
   ```
   # passwd -r files jdoe-local
   New Password:     <Type password>
   Re-enter new Password:  <Retype password>
   passwd: password successfully changed for jdoe-local
   #
   ```

5 **Make sure that you are not logged in as root.**
   
   ```
   # who
   jdoe    console   May 24 13:51   (:0)
   jdoe    pts/5     May 24 13:51   (:0.0)
   jdoe    pts/4     May 24 13:51   (:0.0)
   jdoe    pts/10    May 24 13:51   (:0.0)
   ```

6 **Change root user into a role.**
   
   ```
   # usermod -K type=role root
   ```
Verify that root is a role.
The root entry in the user_attr file should appear similar to the following:

```bash
# grep root /etc/user_attr
root::::type=role;auths=solaris.*,solaris.grant;profiles=...
```

Assign the root role to your local account.

```bash
# usermod -R root jdoe-local
```

Caution – If you do not assign the root role to a user, no one can become superuser, except in single-user mode. You must type a root password to enter single-user mode.

Configure the name service to return in case of failure.

a. Open a new terminal window and assume the root role.

```bash
% whoami
jdoe
% su - jdoe-local
Enter password: <Type jdoe-local password>
% roles
root
% su - root
Enter password: <Type root password>
#
```

b. Edit the nsswitch.conf file.
For example, the following entries in the nsswitch.conf file would enable the name service to return:

```bash
passwd: files nis [TRYAGAIN=0 UNAVAIL=return NOTFOUND=return]
group: files nis [TRYAGAIN=0 UNAVAIL=return NOTFOUND=return]
```

(Optional) Assign the root role to selected user accounts in the name service.
For the procedure, see “How to Change the RBAC Properties of a User” on page 200.

Example 9–8 Preventing the root Role From Being Used to Configure a System

In this example, site security policy requires that several discrete roles configure the system. These discrete roles have been created and tested. To prevent the root account from being used to configure the system, the security administrator changes root into a role, but does not assign the role. The root role retains a password to enter the system in single-user mode.

First, the administrator verifies that root is not an assigned role.
% whoami
jdoe-local
% su - root
Password: a12@3#4$5%6^7
# grep roles /etc/user_attr
jdoe-local:::type=normal;roles=secadmin
kdoe-local:::type=normal;roles=sysadmin

Still in the root account, the administrator changes root into a role.

# usermod -K type=role root

Then, the administrator verifies the change in the root entry in the user_attr file.

# grep root /etc/user_attr
root:::type=role;auths=solaris.*,solaris.grant;profiles=...

Example 9–9 Changing the root Role Back Into the root User

In this example, the administrator is decommissioning a system and wants to log in to the desktop as superuser. The system has been removed from the network.

First, the administrator assumes the root role to remove all root role assignments.

% whoami
jdoe-local
% su - root
Password: a12@3#4$5%6^7
# grep roles /etc/user_attr
jdoe-local:::type=normal;roles=root
kdoe-local:::type=normal;roles=root
# usermod -R "" jdoe-local
# usermod -R "" kdoe-local
# grep roles /etc/user_attr
#

Still in the root role, the administrator changes root into a user.

# rolemod -K type=normal root

Then, the administrator verifies the change in the root entry in the user_attr file.

# grep root /etc/user_attr
root:::type=normal;auths=solaris.*,solaris.grant;profiles=...
In a desktop environment, you cannot directly log in as root when root is a role. A diagnostic message indicates that root is a role on your system. If you do not have a local account that can assume the root role, create one. As root, log in to the system in single-user mode, create a local user account, and assign the root role to the new account. Then, log in as the new user and assume the root role.

No one can become superuser if you change the root user into a role and fail to make one of the following assignments:

- Assign the root role to a valid user.
- Assign a rights profile that is equivalent to root’s rights profile to a valid user. The Primary Administrator profile is an equivalent rights profile for root capabilities.
- Create a role that has the capabilities of root and assign the role to a valid user. A role that is assigned the Primary Administrator profile is equivalent to the root role.

### Using Roles (Task Map)

The following task map points to procedures for using your role after roles have been assigned.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the Solaris Management Console</td>
<td>Authenticate yourself as a role to perform administrative tasks in the Solaris Management Console.</td>
<td>&quot;How to Assume a Role in the Solaris Management Console&quot; on page 190</td>
</tr>
<tr>
<td>Assume a role in a terminal window</td>
<td>Perform command-line administrative tasks in a profile shell.</td>
<td>&quot;How to Assume a Role in a Terminal Window&quot; on page 188</td>
</tr>
</tbody>
</table>

### Using Roles

After you have set up roles with default Solaris rights profiles, and assigned the roles to users, the roles can be used. A role can be assumed on the command line. In the Solaris Management Console, a role can also be used for administering the system locally and over the network.

#### How to Assume a Role in a Terminal Window

**Before You Begin**

The role must already be assigned to you. The name service must be updated with that information.

1. **In a terminal window, determine which roles you can assume.**
   
   `% roles
   Comma-separated list of role names is displayed`
2 Use the su command to assume a role.

   % su - rolename
   Password: <Type rolename password>
   $

   The su - rolename command changes the shell to a profile shell for the role. A profile shell recognizes security attributes (authorizations, privileges, and set ID bits).

3 Verify that you are now in a role.

   $ /usr/ucb/whoami
   rolename

   You can now perform role tasks in this terminal window.

4 (Optional) View the capabilities of your role.

   For the procedure, see “How to Determine the Privileged Commands That a Role Can Run” on page 233.

Example 9–10 Assuming the Primary Administrator Role

In the following example, the user assumes the role of Primary Administrator. In the default configuration, this role is equivalent to superuser. The role then checks to see which privileges are available to any command that is typed in the profile shell for the role.

   % roles
   sysadmin, oper, primaryadm
   % su - primaryadm
   Password: <Type primaryadm password>
   $ /usr/ucb/whoami
   rolename
   Prompt has changed to role prompt
   primaryadm
   $ ppriv $ $
   1200: pfksh
   flags = <none>
     E (Effective): all
     I (Inheritable): basic
     P (Permitted): all
     L (Limit): all

   For information about privileges, see “Privileges (Overview)” on page 164.

Example 9–11 Assuming the root Role

In the following example, the user assumes the root role. The role was created in “How to Make root User Into a Role” on page 184.
Using Roles

% roles
root
% su - root
Password: <Type root password>
# /usr/ucb/whoami  Prompt has changed to role prompt
root
$ ppriv $$
1200: pfksh
flags = <none>
   E: all
   I: basic
   P: all
   L: all

For information about privileges, see “Privileges (Overview)” on page 164.

Example 9–12  Assuming the System Administrator Role

In the following example, the user assumes the role of System Administrator. In contrast to the Primary Administrator role, the System Administrator has the basic set of privileges in its effective set.

% roles
sysadmin,oper,primaryadm
% su - sysadmin
Password: <Type sysadmin password>
$ /usr/ucb/whoami  Prompt has changed to role prompt
sysadmin
$ ppriv $$
1200: pfksh
flags = <none>
   E: basic
   I: basic
   P: basic
   L: all

For information about privileges, see “Privileges (Overview)” on page 164. For a short description of the capabilities of the role, see “System Administrator Rights Profile” on page 206.

▼ How to Assume a Role in the Solaris Management Console

To change information in the Solaris Management Console GUI requires administrative capabilities. A role gives you administrative capabilities. If you want to view information, you must have the solaris.admin.usermgr.read authorization. The Basic Solaris User rights profile includes this authorization.
An administrative role that can change the properties of users or roles must have already been assigned to you. For example, the Primary Administrator role can change the properties of users or roles.

Before You Begin

1. Start the Solaris Management Console.

   % /usr/sbin/smc &

   For detailed instructions, see “Using the Solaris Management Tools With RBAC (Task Map)” in System Administration Guide: Basic Administration.

2. Select the toolbox for your task.

   Navigate to the toolbox that contains the tool or collection in the appropriate name service scope and click the icon. The scopes are files (local), NIS, NIS+, and LDAP. If the appropriate toolbox is not displayed in the navigation pane, choose Open Toolbox from the Console menu and load the relevant toolbox.

3. Select the tool that you want to use.

   Navigate to the tool or collection and click the icon. The tools for managing the RBAC elements are in the Users tool, as shown in the following figure.

4. Type your user name and password in the Login: User Name dialog box.
5 Authenticate yourself in the Login: Role dialog box.
The Role option menu in the dialog box displays the roles that are assigned to you. Choose a role and type the role password.

Managing RBAC (Task Map)

The following task map points to procedures for customizing role-based access control (RBAC) after RBAC has been initially implemented.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change the role password</td>
<td>An authorized user or role changes the password of another role.</td>
<td>“How to Change the Password of a Role” on page 193</td>
</tr>
<tr>
<td>Modify the properties of a role</td>
<td>Modifies the capabilities (privileges, privileged commands, profiles, or authorizations) of a role.</td>
<td>“How to Change the Properties of a Role” on page 195</td>
</tr>
<tr>
<td>Create or change rights profiles</td>
<td>Creates a rights profile. Or modifies the authorizations, privileged commands, or supplementary rights profiles in a rights profile.</td>
<td>“How to Create or Change a Rights Profile” on page 196</td>
</tr>
<tr>
<td>Change a user’s administrative capabilities</td>
<td>Adds a role, a rights profile, an authorization, or privileges to an ordinary user.</td>
<td>“How to Change the RBAC Properties of a User” on page 200</td>
</tr>
<tr>
<td>Secure legacy applications</td>
<td>Turns on the set ID permissions for legacy applications. Scripts can contain commands with set IDs. Legacy applications can check for authorizations, if appropriate.</td>
<td>“How to Add RBAC Properties to Legacy Applications” on page 201</td>
</tr>
</tbody>
</table>

These procedures manage the elements that are used in RBAC. For user management procedures, refer to Chapter 5, “Managing User Accounts and Groups (Tasks),” in System Administration Guide: Basic Administration.

Managing RBAC

The Solaris Management Console GUI is the preferred method for managing RBAC.

Note – Do not attempt to administer RBAC with the command line and the graphical user interface at the same time. Conflicting changes could be made to the configuration, and the behavior would be unpredictable. Both tools can administer RBAC, but you cannot use both tools concurrently.
How to Change the Password of a Role

Before You Begin
You must have assumed a role that includes the User Security profile or have switched to superuser. You cannot be in the role whose password you want to change. A role cannot change its own password.

Use one of the following methods to change a role's password.

- As superuser or in a role that includes the User Security rights profile, run the `passwd` command.
  
  ```
  $ passwd -r naming-service target-rolename
  ```
  
  `-r naming-service` Applies the password change to one of the following repositories `files`, `nis`, `nisplus`, or `ldap`. If a repository is not specified, the password is changed in `files`.

  `target-rolename` Is the name of an existing role that you want to modify.
  
  For more command options, see the `passwd(1)` man page.

- Change the password in the Solaris Management Console.
  
  To start the console, see “How to Assume a Role in the Solaris Management Console” on page 190.

  ```a. Log into the console as superuser or in a role that includes the User Security rights profile.

  The login role cannot be the target role.

  b. Choose the appropriate scope.

  The `Files` scope modifies the role password on the local system. The `LDAP` scope modifies the role password in the LDAP naming service.

  c. Navigate to Administrative Roles and follow the instructions in the left-hand pane.

  For more extensive information, see the online help.

- As superuser or in a role that includes the User Security rights profile, run the `smrole` command with the `modify` subcommand.

  ```
  $ /usr/sadm/bin/smrole -D domain-name -r admin-role -l <Type admin-role password> \ modify -- -n target-rolename -P password
  ```
  
  `-D domain-name` Is the name of the domain that you want to manage.

  `-r admin-role` Is the name of the administrative role that can modify the target role.

  The administrative role must have the
solaris.admin.usermgr.pswd authorization. The administrative role and the target role cannot be the same role.

-1

Is the prompt for the password of admin-role.

- -

Is the required separator between authentication options and subcommand options.

-n target-rolename

Is the name of the target role.

-P password

Is the new password for target-rolename.

For the full list of command options, see the smrole(1M) man page.

Example 9–13  Changing a Local Role’s Password With the passwd Command

In this example, superuser changes the password of the local operadm role.

```
# passwd -r files operadm
New password:  Type new password
Re-enter new password:   Retype new password
```

Example 9–14  Changing a Role’s Password in an LDAP Repository

In this example, the Primary Admin role changes the password of the operadm role in the LDAP directory service.

```
$ passwd -r ldap operadm
New password:  Type new password
Re-enter new password:   Retype new password
```

Example 9–15  Changing a Role’s Password With the smrole modify Command

In this example, the administrator contacts the Solaris Management Console server to change the operadm password in the NIS domain. When the administrator does not provide the password before pressing the Return key, the New Password: prompt appears.

```
$ /usr/sadm/bin/smrole -D nis:/examplehost/example.domain \
-r primaryadm -l <Type primaryadm password> \
modify -- -n operadm -P  Press the Return key
New Password: a!2@3#4$5%6*7
```

$
How to Change the Properties of a Role

Before You Begin
You must have assumed the role of Primary Administrator or have switched to superuser to change the properties of a role. Role properties include password, rights profiles, and authorizations.

Note – To change a role’s password property, see “How to Change the Password of a Role” on page 193.

Use one of the following methods to change the properties of a role.

- Use the Users tool in the Solaris Management Console.
  To start the console, see “How to Assume a Role in the Solaris Management Console” on page 190. Follow the instructions in the left-hand pane to modify a role in Administrative Roles. For more extensive information, see the online help.

- Use the `rolemod` command.
  This command modifies the attributes of a role that is defined in the local name service.
  
  ```sh
  $ rolemod -c comment -P profile-list rolename
  
  -c comment Is the new comment that describes the capabilities of the role.
  -P profile-list Is the list of the profiles that are included in the role. This list replaces the current list of profiles.
  rolename Is the name of an existing, local role that you want to modify.
  
  For more command options, see the `rolemod(1M)` man page.
  
  - Use the `smrole` command with the `modify` subcommand.
  This command modifies the attributes of a role in a distributed name service, such as NIS, NIS+, or LDAP. This command runs as a client of the Solaris Management Console server.
  
  ```sh
  $ /usr/sadm/bin/smrole -D domain-name \ 
  -r admin-role -l <Type admin-role password> \ 
  modify -- -n rolename -r username -u username
  
  -D domain-name Is the name of the domain that you want to manage.
  -r admin-role Is the name of the administrative role that can modify the role. The administrative role must have the `solaris.role.assign` authorization. If you are modifying a role that you have assumed, the role must have the `solaris.role.delegate` authorization.
  -l Is the prompt for the password of admin-role.
- Is the required separator between authentication options and subcommand options.
-rolename Is the name of the new role.
-username Is the name of the user who can no longer assume rolename.
-username Is the name of the user who can now assume rolename.

For more command options, see the `smrole(1M)` man page.

**Example 9–16** Changing a Local Role’s Properties With the `rolemod` Command

In this example, the `operaadm` role is modified to include the Media Restore rights profile.

```bash
$ rolemod -c "Handles printers, backup, AND restore" \ 
-P "Printer Management,Media Backup,Media Restore,All" operaadm
```

**Example 9–17** Changing a Local Role’s Properties With the `smrole modify` Command

In the following example, the `operaadm` role is modified to add the Media Restore rights profile.

```bash
$ /usr/sadm/bin/smrole -r primaryadm -l <Type primaryadm password> \ 
modify -- -n operaadm -c "Handles printers, backup, AND restore" \ 
-p "Media Restore"
```

**Example 9–18** Changing a Role in a Domain With the `smrole modify` Command

In the following example, the `clockmgr` role is changed. The NIS user whose ID is 108 cannot no longer assume the role. The NIS user whose ID is 110 can assume the role `clockmgr`.

```bash
$ /usr/sadm/bin/smrole -D nis:/examplehost/example.domain \ 
-r primaryadm -l <Type primaryadm password> \ 
modify -- -n clockmgr -r 108 -u 110
```

**How to Create or Change a Rights Profile**

A rights profile is a property of a role. You should create or change a rights profile when the prof_attr database does not contain a rights profile that fulfills your needs. To learn more about rights profiles, see “RBAC Rights Profiles” on page 162.

**Before You Begin** To create or change a rights profile, you must have assumed the role of Primary Administrator or have switched to superuser.
Use one of the following methods to create or change a rights profile.

■ **Use the Users tool in the Solaris Management Console.**

To start the console, see “How to Assume a Role in the Solaris Management Console” on page 190. Follow the instructions in the left-hand pane to create or change a rights profile in Rights. For more extensive information, see the online help.

■ **Use the `smprofile` command.**

This command enables you to add, modify, list, or delete a rights profile. The command works on files, and in a distributed name service, such as NIS, NIS+, or LDAP. The `smprofile` command runs as a client of the Solaris Management Console server.

```bash
$ /usr/sadm/bin/smprofile -D domain-name \
-r admin-role -l <Type admin-role password> \nadd | modify -- -n profile-name \n-d description -m help-file -p supplementary-profile
```

- `-D domain-name` is the name of the domain that you want to manage.
- `-r admin-role` is the name of the administrative role that can modify the role. The administrative role must have the `solaris.role.assign` authorization. If you are modifying a role that you have assumed, the role must have the `solaris.role.delegate` authorization.
- `-l` is the prompt for the password of `admin-role`.
- `--` is the required separator between authentication options and subcommand options.
- `-n profile-name` is the name of the new profile.
- `-d description` is a short description of the profile.
- `-m help-file` is the name of the HTML help file that you have created and placed in the `/usr/lib/help/profiles/locale/C` directory.
- `-p supplementary-profile` is the name of an existing rights profile that is included in this rights profile. You can specify multiple `-p supplementary-profile` options.

For more command options, see the `smprofile(1M)` man page.

**Example 9–19  ** Modifying a Rights Profile From the Command Line

In the following example, the Network Management rights profile is made a supplementary profile of the Network Security rights profile. The role that contains the Network Security profile can now configure the network and hosts, as well as run security–relevant commands.
Example 9–20  Modifying an Existing Rights Profile

In the following example, the security administrator of MyCompany customizes the Console User rights profile. Another goal is to retain the customized rights profile when the Solaris OS is updated to a later version.

First, the administrator closes the Solaris Management Console.

Then, the administrator opens the prof_attr file, copies the Console User rights profile to the next line, and renames the second entry. The administrator uses the existing help file, RtConsUser.html.

```
vi /etc/security/prof_attr
```

```
Console User:::Manage System as the Console User:help=RtConsUser.html
MyCompany Console User:::Manage System as the Console User:help=RtConsUser.html
```

The administrator assumes the secadmin role. The secadmin role can modify the security features of a system. The secadmin role opens the Solaris Management Console, clicks the System Configuration and the Users tool, types the role password, and double-clicks the Rights tool.

The administrator double-clicks the MyCompany Console User rights profile. Under the Authorizations tab, the administrator adds two authorizations to the Authorizations Included list and saves the changes. When the system is patched or updated to a later version of the Solaris OS, the Console User rights profile is updated and the MyCompany Console User rights profile is not changed.

Example 9–21  Creating a New Rights Profile With the Rights Tool

The following table shows sample data for a hypothetical rights profile that is called “Build Administrator”. This rights profile includes the commands in the subdirectory /usr/local/swctrl/bin. These commands have an effective UID of 0. The Build Administrator rights profile would be useful for administrators who manage the builds and versioning for software development.
### Troubleshooting

Check the following if the rights profile does not provide the role with the capabilities that you expect:

- Are the rights profiles for the role listed in the GUI from most to least powerful?
  For example, if the All rights profile is at the top of the list, then no commands are run with security attributes. A profile that contains commands with security attributes must precede the All rights profile in the list.

- Is a command listed more than once in the role’s rights profiles? If so, does the first instance of the command have all the security attributes that are required?
  For example, a command can require privileges for particular options to the command. For the options that require privileges to succeed, the first instance of the command in the highest rights profile in the list must have the assigned privileges.

- Do the commands in the role’s rights profiles have the appropriate security attributes?
  For example, when the policy is suser, some commands require uid=0 rather than euid=0 to succeed.

- Has the name service cache, svc:/system/name-service-cache, been restarted?
  The nscd daemon can have a lengthy time-to-live interval. By restarting the daemon, you update the name service with current data.
How to Change the RBAC Properties of a User

User properties include password, rights profiles, and authorizations. The most secure method of giving a user administrative capabilities is to assign a role to the user. For a discussion, see “Security Considerations When Directly Assigning Security Attributes” on page 163.

Before You Begin

You must have assumed the role of Primary Administrator or have switched to superuser to change the properties of a user.

Use one of the following methods to change the RBAC properties of a user.

- Use the Users tool in the Solaris Management Console.
  
  To start the console, see “How to Assume a Role in the Solaris Management Console” on page 190. Follow the instructions in the left-hand pane to modify a user in User Accounts. For more extensive information, see the online help.

  Tip – It is not good practice to assign authorizations, privileges, or rights profiles directly to users. The preferred approach is to assign a role to users. Users then assume a role to perform privileged operations.

- Use the usermod command.
  
  This command modifies the attributes of a user that is defined in the local name service.

  `$ usermod -R rolename username`

  `-R rolename` Is the name of an existing local role.

  `username` Is the name of an existing, local user that you want to modify.

  For more command options, see the `usermod(1M)` man page.

- Use the smuser command with the modify subcommand.
  
  This command modifies the attributes of a user in a distributed name service, such as NIS, NIS+, or LDAP. This command runs as a client of the Solaris Management Console server.

  `$ /usr/sadm/bin/smuser -D domain-name \ 
  -r admin-role -l <Type admin-role password> \ 
  modify -- -n username -a rolename`

  `-D domain-name` Is the name of the domain that you want to manage.

  `-r admin-role` Is the name of the administrative role that can modify the role. The administrative role must have the `solaris.role.assign` authorization. If you are modifying a role that you have assumed, the role must have the `solaris.role.delegate` authorization.

  `-l` Is the prompt for the password of `admin-role`. 
- -  Is the required separator between authentication options and subcommand options.

- \texttt{n} \textit{username}  Is the name of the user who is being assigned \textit{rolename}.

- \texttt{a} \textit{rolename}  Is the name of the role that you are assigning to \textit{username}. You can specify multiple - \texttt{a} \textit{rolename} options.

For more command options, see the \texttt{smuser(1M)} man page.

\textbf{Example 9–22}  \textbf{Modifying a Local User's RBAC Properties From the Command Line}

In this example, the user \textit{jdoe} can now assume the role of System Administrator.

\texttt{$ \textnormal{usermod -R sysadmin jdoe}$}

\textbf{Example 9–23}  \textbf{Modifying a User's RBAC Properties With the \texttt{smuser} Command}

In this example, the user \textit{jdoe} is assigned two roles, System Administrator and Operator. Because the user and the roles are defined locally, the - \texttt{D} option is not necessary.

\texttt{$ /usr/sadm/bin/smuser -r primaryadm -l <Type primaryadm password> \ 
modify -- -n jdoe -a sysadmin -a operadm$}

In the following example, the user is defined in the NIS name service. Therefore, the - \texttt{D} option is required. Two roles are defined in the name service. One role, root, is defined locally.

\texttt{$ /usr/sadm/bin/smuser -D nis:/examplehost/example.domain \ 
-r primaryadm -l <Type primaryadm password> \ 
modify -- -n jdoe -a sysadmin -a operadm -a root$}

```
\section*{How to Add RBAC Properties to Legacy Applications}

A legacy application is a command or set of commands. The security attributes are set for each command in a rights profile. The rights profile is then included in a role. A user who assumes the role can run the legacy application with the security attributes.

To add legacy applications to the Solaris Management Console, see “Adding Tools to the Solaris Management Console” in \textit{System Administration Guide: Basic Administration}.

\section*{Before You Begin}

You must have assumed the role of Primary Administrator or have switched to superuser to change the security attributes of a command in a rights profile.
Use the Users tool in the Solaris Management Console.
To start the console, see "How to Assume a Role in the Solaris Management Console" on page 190. Follow the instructions in the left-hand pane to modify a rights profile in Rights. For more extensive information, see the online help.

2 Add security attributes to the commands that implement the legacy application.
You add security attributes to a legacy application in the same way that you would for any command. You must add the command with security attributes to a rights profile. For a legacy command, give the command euid=0 or uid=0 security attributes. For details of the procedure, see "How to Create or Change a Rights Profile" on page 196.

3 After adding the legacy application to a rights profile, include the rights profile in a role's list of profiles.
To add a rights profile to a role, see "How to Change the Properties of a Role" on page 195.

Example 9–24 Adding Security Attributes to Commands in a Script
If a command in a script needs to have the setuid bit or setgid bit set to succeed, the script executable and the command must have the security attributes added in a rights profile. Then, the rights profile is included in a role, and the role is assigned to a user. When the user assumes the role and executes the script, the command runs with the security attributes.

To add security attributes to a command or shell script, see "How to Create or Change a Rights Profile" on page 196.

Example 9–25 Checking for Authorizations in a Script or Program
To have a script for authorizations, you need to add a test that is based on the auths command. For detailed information about this command, see the auths(1) man page.

For example, the following line tests if the user has the authorization that is supplied as the $1 argument:

```bash
if [ '/usr/bin/auths'/usr/xpg4/bin/grep $1' ]; then
    echo Auth granted
else
    echo Auth denied
fi
```
To be more complete, the test should include logic that checks for other authorizations that use wildcards. For example, to test if the user has the `solaris.admin.usermgr.write` authorization, you would need to check for the following strings:

- `solaris.admin.usermgr.write`
- `solaris.admin.usermgr.*`
- `solaris.admin.*`
- `solaris.*`

If you are writing a program, use the function `getauthattr()` to test for the authorization.
This chapter provides reference material about RBAC. The following is a list of the reference information in this chapter:

- “Contents of Rights Profiles” on page 205
- “Authorization Naming and Delegation” on page 210
- “Databases That Support RBAC” on page 211
- “RBAC Commands” on page 218

For information on using RBAC, see Chapter 9, "Using Role-Based Access Control (Tasks)." For overview information, see “Role-Based Access Control (Overview)” on page 155.

### Contents of Rights Profiles

This section describes some typical rights profiles. Rights profiles can include authorizations, commands with security attributes, and supplementary rights profiles. The rights profiles are listed from most to least powerful. For suggestions on how to distribute rights profiles to roles at your site, see “How to Plan Your RBAC Implementation” on page 175.

- **Primary Administrator rights profile** – Provides the capabilities of superuser in one profile.
- **System Administrator rights profile** – Provides a profile that can do most tasks that are not connected with security. This profile includes several other profiles to create a powerful role.
- **Operator rights profile** – Provides limited capabilities to manage files and offline media. This profile includes supplementary rights profiles to create a simple role.
- **Printer Management rights profile** – Provides a limited number of commands and authorizations to handle printing. This profile is one of several profiles that cover a single area of administration.
- **Basic Solaris User rights profile** – Enables users to use the system within the bounds of security policy. This profile is listed by default in the `policy.conf` file.
**Primary Administrator Rights Profile**

The Primary Administrator rights profile is assigned to the most powerful role on the system. The role that includes the Primary Administrator rights profile has superuser capabilities.

- The `solaris.*` authorization effectively assigns all of the authorizations that are provided by the Solaris software.
- The `solaris.grant` authorization lets a role assign any authorization to any rights profile, role, or user.
- The command assignment `*:uid=0;gid=0` provides the ability to run any command with UID=0 and GID=0.

You can customize the help file `RtPriAdmin.html` for your site, if necessary. Help files are stored in the `/usr/lib/help/profiles/locale/C` directory.

Note also that if the Primary Administrator rights profile is not consistent with a site's security policy, the profile can be modified or not assigned at all. However, the security capabilities in the Primary Administrator rights profile would need to be handled in one or more other rights profiles. Those other rights profiles would then be assigned to roles.

**TABLE 10–1  Contents of Primary Administrator Rights Profile**

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>To perform all administrative tasks</td>
<td><strong>Commands:</strong> <code>*:uid=0;gid=0</code></td>
</tr>
<tr>
<td></td>
<td><strong>Authorizations:</strong> <code>solaris.*</code>, <code>solaris.grant</code></td>
</tr>
<tr>
<td></td>
<td><strong>Help File:</strong> <code>RtPriAdmin.html</code></td>
</tr>
</tbody>
</table>

**System Administrator Rights Profile**

The System Administrator rights profile is intended for the System Administrator role. Because the System Administrator does not have the broad capabilities of the Primary Administrator, no wildcards are used. Instead, this profile is a set of discrete, supplementary administrative rights profiles that do not deal with security. The commands with security attributes from one of the supplementary rights profiles are shown.
Note that the All rights profile is assigned at the end of the list of supplementary rights profiles.

### TABLE 10–2  Contents of System Administrator Rights Profile

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>To perform most nonsecurity</td>
<td>Supplementary rights profiles: Audit Review, Printer Management, Cron</td>
</tr>
<tr>
<td>administrative tasks</td>
<td>Management, Device Management, File System Management, Mail</td>
</tr>
<tr>
<td></td>
<td>Management, Maintenance and Repair, Media Backup, Media Restore, Name</td>
</tr>
<tr>
<td></td>
<td>Service Management, Network Management, Object Access Management,</td>
</tr>
<tr>
<td></td>
<td>Process Management, Software Installation, Project Management, User</td>
</tr>
<tr>
<td></td>
<td>Management, All</td>
</tr>
<tr>
<td></td>
<td>Help File: RtSysAdmin.html</td>
</tr>
</tbody>
</table>

Commands from one of the supplementary profiles

- **Object Access Management rights profile**, solaris policy:
  - /usr/bin/chgrp:privs=file_chown,
  - /usr/bin/chmod:privs=file_chown,
  - /usr/bin/chown:privs=file_chown,
  - /usr/bin/setfacl:privs=file_chown

- suser policy:
  - /usr/bin/chgrp:euid=0,
  - /usr/bin/chmod:euid=0,
  - /usr/bin/chown:euid=0,
  - /usr/bin/setfacl:euid=0

### Operator Rights Profile

The Operator rights profile is a less powerful profile that provides the ability to do backups and printer maintenance. The ability to restore files has more security consequences. Therefore, in this profile, the default is to not include the ability to restore files.

### TABLE 10–3  Contents of Operator Rights Profile

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>To perform simple administrative tasks</td>
<td>Supplementary rights profiles: Printer Management, Media Backup, All</td>
</tr>
<tr>
<td></td>
<td>Help File: RtOperator.html</td>
</tr>
</tbody>
</table>

### Printer Management Rights Profile

Printer Management is a typical rights profile that is intended for a specific task area. This profile includes authorizations and commands. The following table shows a partial list of commands.
### Basic Solaris User Rights Profile

By default, the Basic Solaris User rights profile is assigned automatically to all users through the `policy.conf` file. This profile provides basic authorizations that are useful in normal operations. Note that the convenience that is offered by the Basic Solaris User rights profile must be balanced against site security requirements. Sites that need stricter security might prefer to remove this profile from the `policy.conf` file.

#### TABLE 10–5  Contents of Basic Solaris User Rights Profile

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>To automatically assign rights to all users</td>
<td><strong>Authorizations:</strong> solaris.profmgr.read, solaris.jobs.user, solaris.mail.mailq, solaris.device.mount.removable, solaris.admin.usermgr.read, solaris.admin.logsvc.read, solaris.admin.fsmgr.read, solaris.admin.serialmgr.read, solaris.admin.diskmgr.read, solaris.admin.procmgr.user, solaris.compsys.read, solaris.adminprinter.read, solaris.admin.proddreg.read, solaris.admin.dcmgr.read, solaris.snmp.read, solaris.project.read, solaris.admin.patchmg.read, solaris.network.hosts.read, solaris.admin.volgmr.read</td>
</tr>
<tr>
<td></td>
<td><strong>Commands:</strong> /usr/bin/cdda2wav.bin:privs=file_dac_read,sys_devices, proc_priocntl,net_privaddr, /usr/bin/cdrecord.bin:privs=file_dac_read,sys_devices, proc_lock_memory,proc_priocntl,net_privaddr, /usr/bin/readcd.com:privs=file_dac_read,sys_devices,net_privaddr, /usr/lib/osspm/lp-queue-helper:euid=lp;gid=lp,</td>
</tr>
<tr>
<td></td>
<td><strong>Supplementary rights profiles:</strong> All</td>
</tr>
<tr>
<td></td>
<td><strong>Help File:</strong> RtDefault.html</td>
</tr>
</tbody>
</table>
**Console User Rights Profile**

The Console User rights profile is intended for the console user, that is, the person who is seated in front of the system. This profile is delivered with a convenient set of authorizations for the console user. You can customize the Console User rights profile to satisfy your site security requirements. For an example, see Example 9–20.

**All Rights Profile**

The All rights profile uses the wildcard to include all commands. This profile provides a role with access to all commands that are not explicitly assigned in other rights profiles. Without the All rights profile or other rights profiles that use wildcards, a role has access to explicitly assigned commands only. Such a limited set of commands is not very practical.

The All rights profile, if used, should be the final rights profile that is assigned. This last position ensures that explicit security attribute assignments in other rights profiles are not inadvertently overridden.

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Contents</th>
</tr>
</thead>
</table>
| To execute any command as the user or role | **Commands:** *  
**Help File:** RtAll.html |

**Order of Rights Profiles**

The commands in rights profiles are interpreted in order. The first occurrence of a command is the only version of the command that is used for that role or user. Different rights profiles can include the same command. Therefore, the order of rights profiles in a list of profiles is important. The rights profile with the most capabilities should be listed first.

Rights profiles are listed in the Solaris Management Console GUI and in the prof_attr file. In the Solaris Management Console GUI, the rights profile with the most capabilities should be the top profile in a list of assigned rights profiles. In the prof_attr file, the rights profile with the most capabilities should be the first in a list of supplementary profiles. This placement ensures that a command with security attributes is listed before that same command without security attributes.

**Viewing the Contents of Rights Profiles**

The Solaris Management Console Rights tool provides one way of inspecting the contents of the rights profiles.
The prof_attr and exec_attr files offer a more fragmented view. The prof_attr file contains the name of every rights profile that is defined on the system. The file also includes the authorizations and the supplementary rights profiles for each profile. The exec_attr file contains the names of rights profiles and their commands with security attributes.

Authorization Naming and Delegation

An RBAC authorization is a discrete right that can be granted to a role or a user. Authorizations are checked by RBAC-compliant applications before a user gets access to the application or specific operations within the application. This check replaces the tests in conventional UNIX applications for UID=0.

Authorization Naming Conventions

An authorization has a name that is used internally and in files. For example, solaris.admin.usermgr.pswd is the name of an authorization. An authorization has a short description, which appears in the graphical user interfaces (GUIs). For example, Change Passwords is the description of the solaris.admin.usermgr.pswd authorization.

By convention, authorization names consist of the reverse order of the Internet name of the supplier, the subject area, any subareas, and the function. The parts of the authorization name are separated by dots. An example would be com.xyzcorp.device.access. Exceptions to this convention are the authorizations from Sun Microsystems, Inc., which use the prefix solaris instead of an Internet name. The naming convention enables administrators to apply authorizations in a hierarchical fashion. A wildcard (*) can represent any strings to the right of a dot.

Example of Authorization Granularity

As an example of how authorizations are used, consider the following: A user in the Operator role might be limited to the solaris.admin.usermgr.read authorization, which provides read but not write access to user configuration files. The System Administrator role naturally has the solaris.admin.usermgr.read and the solaris.admin.usermgr.write authorizations for making changes to user files. However, without the solaris.admin.usermgr.pswd authorization, the System Administrator cannot change passwords. The Primary Administrator has all three of these authorizations.

The solaris.admin.usermgr.pswd authorization is required to make password changes in the Solaris Management Console User tool. This authorization is also required for using the password modification options in the smuser, smmultiuser, and smrole commands.
Delegation Authority in Authorizations

An authorization that ends with the suffix grant enables a user or a role to delegate to other users any assigned authorizations that begin with the same prefix.

For example, a role with the authorizations `solaris.admin.usermgr.grant` and `solaris.admin.usermgr.read` can delegate the `solaris.admin.usermgr.read` authorization to another user. A role with the `solaris.admin.usermgr.grant` and `solaris.admin.usermgr.*` authorizations can delegate any of the authorizations with the `solaris.admin.usermgr` prefix to other users.

Databases That Support RBAC

The following four databases store the data for the RBAC elements:

- **Extended user attributes database** (`user_attr`) – Associates users and roles with authorizations and rights
- **Rights profile attributes database** (`prof_attr`) – Defines rights profiles, lists the profiles’ assigned authorizations, and identifies the associated help file
- **Authorization attributes database** (`auth_attr`) – Defines authorizations and their attributes, and identifies the associated help file
- **Execution attributes database** (`exec_attr`) – Identifies the commands with security attributes that are assigned to specific rights profiles

The `policy.conf` database contains authorizations, privileges, and rights profiles that are applied to all users. For more information, see “policy.conf File” on page 217.

RBAC Database Relationships

Each RBAC database uses a `key=value` syntax for storing attributes. This method accommodates future expansion of the databases. The method also enables a system to continue to operate if the system encounters a keyword that is unknown to its policy. The `key=value` contents link the files. The following linked entries from the four databases illustrate how the RBAC databases work together.

**EXAMPLE 10-1  Showing RBAC Database Connections**

In the following example, the user `jdoe` gets the capabilities of the File System Management profile through being assigned the role `filemgr`.

1. The user `jdoe` is assigned the role `filemgr` in the `jdoe` user entry in the `user_attr` database.

   ```
   # user_attr - user definition
   jdoe::::type=normal;roles=filemgr
   ```
2. The role filemgr is assigned the rights profile File System Management in the role's entry in the user_attr database.

   # user_attr - role definition
   filemgr::::profiles=File System Management;type=role

   The user and the role are uniquely defined in the passwd and shadow files on the local system, or in equivalent databases in a distributed name service.

3. The File System Management rights profile is defined in the prof_attr database. This database also assigns three sets of authorizations to the File System Management entry.

   # prof_attr - rights profile definitions and assigned authorizations
   File System Management:::Manage, mount, share file systems:
   help=RtFileSysMngmnt.html;
   auths=solaris.admin.fsmgr.*,solaris.admin.diskmgr.*,solaris.admin.volmgr.*

4. The authorizations are defined in the auth_attr database.

   # auth_attr - authorization definitions
   solaris.admin.fsmgr.:::Mounts and Shares::help=AuthFsmgrHeader.html
   solaris.admin.fsmgr.read:::View Mounts and Shares::help=AuthFsmgrRead.html
   solaris.admin.fsmgr.write:::Mount and Share Files::help=AuthFsmgrWrite.html

5. The File System Management rights profile is assigned commands with security attributes in the exec_attr database.

   # exec_attr - rights profile names with secured commands
   File System Management:suser:cmd:::/usr/sbin/mount:uid=0
   File System Management:suser:cmd:::/usr/sbin/dfshares:euid=0
   ... File System Management:solaris:cmd:::/usr/sbin/mount:privs=sys_mount
   ...

**RBAC Databases and the Name Service**

The name service scope of the RBAC databases can apply to the local host only. The scope can also include all hosts that are served by a name service such as NIS, NIS+, or LDAP. Which name service has precedence is set for each of the databases in the /etc/nsswitch.conf file.

- **auth_attr entry** – Sets the name service precedence for the auth_attr database.
- **passwd entry** – Sets the name service precedence for the user_attr database.
- **prof_attr entry** – Sets the name service precedence for the prof_attr database. Also sets the name service precedence for the exec_attr database.

For example, if a command with security attributes is assigned to a rights profile that exists in two name service scopes, only the entry in the first name service scope is used.
user_attr Database

The user_attr database contains user and role information that supplements the passwd and shadow databases. The user_attr database contains extended user attributes such as authorizations, rights profiles, and assigned roles. The fields in the user_attr database are separated by colons, as follows:

user:qualifier:res1:res2:attr

The fields have the following meanings:

user
The name of the user or role as specified in the passwd database.

qualifier:res1:res2
These fields are reserved for future use.

attr
An optional list of semicolon-separated (;) key-value pairs that describes the security attributes to be applied when the user runs commands. The four valid keys are type, auths, profiles, and roles.

- The type keyword can be set to normal, if this account is for a normal user. The type is role if this account is for a role.
- The auths keyword specifies a comma-separated list of authorization names that are chosen from names that are defined in the auth_attr database. Authorization names can include the asterisk (*) character as a wildcard. For example, solaris.device.* means all of the Solaris device authorizations.
- The profiles keyword specifies an ordered, comma-separated list of rights profile names from the prof_attr database. The order of rights profiles works similarly to UNIX search paths. The first profile in the list that contains the command to be executed defines which (if any) security attributes are to be applied to the command.
- The roles keyword can be assigned to the user through a comma-separated list of role names. Note that roles are defined in the same user_attr database. Roles are indicated by setting the type value to role. Roles cannot be assigned to other roles.

The following example demonstrates how the Operator role is defined in a typical user_attr database. The example shows how the role is assigned to user jdoe. Roles and users are differentiated by the type keyword.

% grep operator /etc/user_attr
jdoe:::type=normal;roles=operator
operator:::profiles=Operator;type=role
auth_attr Database

All authorizations are stored in the auth_attr database. Authorizations can be assigned to users, to roles, or to rights profiles. The preferred method is to place authorizations in a rights profile, to include the profile in a role’s list of profiles, and then to assign the role to a user.

The fields in the auth_attr database are separated by colons, as follows:

authname:res1:res2:short_desc:long_desc:attr

The fields have the following meanings:

authname A unique character string that is used to identify the authorization in the format prefix.[suffix]. Authorizations for the Solaris OS use solaris as a prefix. All other authorizations should use a prefix that begins with the reverse-order Internet domain name of the organization that creates the authorization (for example, com.xyzcompany). The suffix indicates what is being authorized, which is typically the functional area and operation.

When the authname consists of a prefix and functional area and ends with a period, the authname serves as a heading to be used by applications in their GUls. A two-part authname is not an actual authorization. The authname of solaris.printmgr. is an example of a heading.

When authname ends with the word “grant,” the authname serves as a grant authorization. A grant authorization enables the user to delegate to other users authorizations with the same prefix and functional area. The authname of solaris.printmgr.grant is an example of a grant authorization. solaris.printmgr.grant gives the user the right to delegate to other users such authorizations as solaris.printmgr.admin and solaris.printmgr.nobanner.

res1:res2 Reserved for future use.

short_desc A short name for the authorization. This short name is suitable for display in user interfaces, such as in a scrolling list in a GUI.

long_desc A long description. This field identifies the purpose of the authorization, the applications in which the authorization is used, and the type of user who might use the authorization. The long description can be displayed in the help text of an application.

attr An optional list of semicolon-separated (;) key-value pairs that describe the attributes of an authorization. Zero or more keys can be specified.

The keyword help identifies a help file in HTML. Help files can be accessed from the index.html file in the /usr/lib/help/auths/locale/C directory.
The following example shows an auth_attr database with some typical values:

```
% grep printer /etc/security/auth_attr
solaris.admin.printer.:::Printer Information::help=AuthPrinterHeader.html
solaris.admin.printer.delete:::Delete Printer Information::help=AuthPrinterDelete.html
solaris.admin.printer.modify:::Update Printer Information::help=AuthPrinterModify.html
solaris.admin.printer.read:::View Printer Information::help=AuthPrinterRead.html
```

Note that `solaris.admin.printer.` is defined as a heading, because the authorization name ends in a dot (.). Headings are used by the GUIs to organize families of authorizations.

### prof_attr Database

The prof_attr database stores the name, description, help file location, and authorizations that are assigned to rights profiles. The commands and security attributes that are assigned to rights profiles are stored in the exec_attr database. For more information, see "exec_attr Database" on page 216. The fields in the prof_attr database are separated by colons, as follows:

```
profname:res1:res2:desc:attr
```

The fields have the following meanings:

- **profname**: The name of the rights profile. Rights profile names are case-sensitive. This name is also used by the user_attr database to indicate the profiles that are assigned to roles and users.
- **res1:res2**: Reserved for future use.
- **desc**: A long description. This field should explain the purpose of the rights profile, including what type of user would be interested in using the profile. The long description should be suitable for display in the help text of an application.
- **attr**: An optional list of key-value pairs that are separated by semicolons (;) that describes the security attributes to apply to the object on execution. Zero or more keys can be specified. The two valid keys are `help` and `auths`.

The keyword `help` identifies a help file in HTML. Help files can be accessed from the `index.html` file in the `/usr/lib/help/profiles/locale/C` directory.

The keyword `auths` specifies a comma-separated list of authorization names that are chosen from those names that are defined in the auth_attr database. Authorization names can be specified with the asterisk (*) character as a wildcard.
The following examples show two typical prof_attr database entries. Note that the Printer Management rights profile is a supplementary rights profile of the Operator rights profile. The example is wrapped for display purposes.

```
% grep 'Printer Management' /etc/security/prof_attr
Printer Management::: Name of rights profile
Manage printers, daemons, spooling: Description
help=RtPrntAdmin.html; Help file
auths=solaris.admin.printer.read, Authorizations
solaris.admin.printer.modify,solaris.admin.printer.delete...
Operator::: Name of rights profile
Can perform simple administrative tasks: Description
profiles=Printer Management, Supplementary rights profiles
Media Backup,All;
help=RtOperator.html Help file
```

**exec_attr Database**

The exec_attr database defines commands that require security attributes to succeed. The commands are part of a rights profile. A command with its security attributes can be run by roles to whom the profile is assigned.

The fields in the exec_attr database are separated by colons, as follows:

```
name:policy:type:res1:res2:id:attr
```

The fields have the following meanings.

- **profname**: The name of the rights profile. Rights profile names are case-sensitive. The name refers to a profile in the prof_attr database.
- **policy**: The security policy that is associated with this entry. Currently, suser and solaris are the valid entries. The solaris policy recognizes privileges. The suser policy does not.
- **type**: The type of entity that is specified. Currently, the only valid entity type is cmd (command).
- **res1:res2**: Reserved for future use.
- **id**: A string that identifies the entity. Commands should have the full path or a path with a wildcard (*). To specify arguments, write a script with the arguments and point the id to the script.
An optional list of semicolon (;) separated key-value pairs that describes the security attributes to apply to the entity on execution. Zero or more keys can be specified. The list of valid keywords depends on the policy that is enforced.

For the suser policy, the four valid keys are euid, uid, egid, and gid.

- The euid and uid keywords contain a single username or a numeric user ID (UID). Commands that are designated with euid run with the supplied UID, which is similar to setting the setuid bit on an executable file. Commands that are designated with uid run with both the real UID and the effective UID.

- The egid and gid keywords contain a single group name or numeric group ID (GID). Commands that are designated with egid run with the supplied GID, which is similar to setting the setgid bit on an executable file. Commands that are designated with gid run with both the real GID and the effective GID.

For the solaris policy, the valid keyword is privs. The value consists of a list of privileges that are separated by commas.

The following example shows some typical values from an exec_attr database:

```bash
% grep 'File System Management' /etc/security/exec_attr
File System Management:suser:cmd:::/usr/sbin/ff:euid=0
File System Management:solaris:cmd:::/usr/sbin/mount:privs=sys_mount
...
```

**policy.conf File**

The policy.conf file provides a way of granting specific rights profiles, specific authorizations, and specific privileges to all users. The relevant entries in the file consist of key=value pairs:

- **AUTHS_GRANTED=authorizations** – Refers to one or more authorizations.
- **PROFS_GRANTED=rights profiles** – Refers to one or more rights profiles.
- **CONSOLE_USER=Console User** – Refers to the Console User rights profile. This profile is delivered with a convenient set of authorizations for the console user. You can customize this profile.
- **PRIV_DEFAULT=privileges** – Refers to one or more privileges.
- **PRIV_LIMIT=privileges** – Refers to all privileges.

The following example shows some typical values from a policy.conf database:
RBAC Commands

This section lists commands that are used to administer RBAC. Also provided is a table of commands whose access can be controlled by authorizations.

Commands That Manage RBAC

While you can edit the local RBAC databases manually, such editing is strongly discouraged. The following commands are available for managing access to tasks with RBAC.

<table>
<thead>
<tr>
<th>Man Page for Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auths(1)</td>
<td>Displays authorizations for a user.</td>
</tr>
<tr>
<td>nakedbm(1M)</td>
<td>Makes a dbm file.</td>
</tr>
<tr>
<td>nscd(1M)</td>
<td>Name service cache daemon, useful for caching the user_attr, prof_attr, and exec_attr databases. Use the svcadm command to restart the daemon.</td>
</tr>
<tr>
<td>pam_roles(5)</td>
<td>Role account management module for PAM. Checks for the authorization to assume role.</td>
</tr>
<tr>
<td>pfexec(1)</td>
<td>Used by profile shells to execute commands with security attributes that are specified in the exec_attr database.</td>
</tr>
<tr>
<td>policy.conf(4)</td>
<td>Configuration file for system security policy. Lists granted authorizations, granted privileges, and other security information.</td>
</tr>
<tr>
<td>profiles(1)</td>
<td>Displays rights profiles for a specified user.</td>
</tr>
<tr>
<td>roles(1)</td>
<td>Displays roles that a specified user can assume.</td>
</tr>
</tbody>
</table>
### RBAC Administration Commands (Continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>roleadd(1M)</td>
<td>Adds a role to a local system.</td>
</tr>
<tr>
<td>roledel(1M)</td>
<td>Deletes a role from a local system.</td>
</tr>
<tr>
<td>rolemod(1M)</td>
<td>Modifies a role’s properties on a local system.</td>
</tr>
<tr>
<td>smattrpop(1M)</td>
<td>Merges the source security attribute database into the target database. For use in situations where local databases need to be merged into a name service. Also for use in upgrades where conversion scripts are not supplied.</td>
</tr>
<tr>
<td>smexec(1M)</td>
<td>Manages entries in the exec_attr database. Requires authentication.</td>
</tr>
<tr>
<td>smmultiuser(1M)</td>
<td>Manages bulk operations on user accounts. Requires authentication.</td>
</tr>
<tr>
<td>smprofile(1M)</td>
<td>Manages rights profiles in the prof_attr and exec_attr databases. Requires authentication.</td>
</tr>
<tr>
<td>smrole(1M)</td>
<td>Manages roles and users in role accounts. Requires authentication.</td>
</tr>
<tr>
<td>smuser(1M)</td>
<td>Manages user entries. Requires authentication.</td>
</tr>
<tr>
<td>useradd(1M)</td>
<td>Adds a user account to the system. The -P option assigns a role to a user’s account.</td>
</tr>
<tr>
<td>userdel(1M)</td>
<td>Deletes a user’s login from the system.</td>
</tr>
<tr>
<td>usermod(1M)</td>
<td>Modifies a user’s account properties on the system.</td>
</tr>
</tbody>
</table>

### Commands That Require Authorizations

The following table provides examples of how authorizations are used to limit command options on a Solaris system. For more discussion of authorizations, see "Authorization Naming and Delegation" on page 210.

<table>
<thead>
<tr>
<th>Command</th>
<th>Authorization Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>at(1)</td>
<td>solaris.jobs.user required for all options (when neither at.allow nor at.deny files exist)</td>
</tr>
<tr>
<td>atq(1)</td>
<td>solaris.jobs.admin required for all options</td>
</tr>
<tr>
<td>cdrw(1)</td>
<td>solaris.device.cdrw required for all options, and is granted by default in the policy.conf file</td>
</tr>
<tr>
<td>crontab(1)</td>
<td>solaris.jobs.user required for the option to submit a job (when neither crontab.allow nor crontab.deny files exist)</td>
</tr>
<tr>
<td></td>
<td>solaris.jobs.admin required for the options to list or modify other users' crontab files</td>
</tr>
</tbody>
</table>

---

TABLE 10–8  Commands and Associated Authorizations
<table>
<thead>
<tr>
<th>Man Page for Command</th>
<th>Authorization Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>allocate(1)</td>
<td><code>solaris.device.allocate</code> (or other authorization as specified in <code>device_allocate</code> file) required to allocate a device</td>
</tr>
<tr>
<td></td>
<td><code>solaris.device.revoke</code> (or other authorization as specified in <code>device_allocate</code> file) required to allocate a device to another user (-F option)</td>
</tr>
<tr>
<td>deallocate(1)</td>
<td><code>solaris.device.allocate</code> (or other authorization as specified in <code>device_allocate</code> file) required to deallocate another user's device</td>
</tr>
<tr>
<td></td>
<td><code>solaris.device.revoke</code> (or other authorization as specified in <code>device_allocate</code>) required to force deallocation of the specified device (-F option) or all devices (-I option)</td>
</tr>
<tr>
<td>list_devices(1)</td>
<td><code>solaris.device.revoke</code> required to list another user's devices (-U option)</td>
</tr>
<tr>
<td>sendmail(1M)</td>
<td><code>solaris.mail</code> required to access mail subsystem functions; <code>solaris.mail.mailq</code> required to view mail queue</td>
</tr>
</tbody>
</table>
This chapter provides step-by-step instructions for managing privileges and using privileges on your system. The following is a list of the information in this chapter.

- “Managing and Using Privileges (Task Map)” on page 221
- “Managing Privileges (Task Map)” on page 221
- “Determining Your Privileges (Task Map)” on page 229

For an overview of privileges, see “Privileges (Overview)” on page 164. For reference information, see Chapter 12, “Privileges (Reference).”

### Managing and Using Privileges (Task Map)

The following task map points to task maps for managing privileges and for using privileges.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use privileges at your site</td>
<td>Involves assigning, removing, adding, and debugging the use of privileges.</td>
<td>“Managing Privileges (Task Map)” on page 221</td>
</tr>
<tr>
<td>Use privileges when you run a command</td>
<td>Involves using the privileges that have been assigned to you.</td>
<td>“Determining Your Privileges (Task Map)” on page 229</td>
</tr>
</tbody>
</table>

### Managing Privileges (Task Map)

The following task map points to procedures for viewing privileges, assigning privileges, and running a script that contains privileged commands.
Managing Privileges

The most secure way to manage privileges for users and roles is to confine use of privilege to commands in a rights profile. The rights profile is then included in a role. The role is assigned to a user. When the user assumes the assigned role, the privileged commands are available to be run in a profile shell. The following procedures show how to assign privileges, remove privileges, and debug privilege use.

### How to Determine the Privileges on a Process

This procedure shows how to determine which privileges are available to your processes. The listing does not include privileges that have been assigned to particular commands.

- **List the privileges that are available to your shell’s process.**

  ```
  % ppriv pid
  $ ppriv -v pid
  
  pid  Is the process number. Use a double dollar sign ($$) to pass the process number of the parent shell to the command.
  -v   Provides a verbose listing of the privilege names.
  ```
Determining the Privileges in Your Current Shell

In the following example, the privileges in the parent process of the user's shell process are listed. In the second example, the full names of the privileges are listed. The single letters in the output refer to the following privilege sets:

- **E** is the effective privilege set.
- **I** is the inheritable privilege set.
- **P** is the permitted privilege set.
- **L** is the limit privilege set.

```
% ppriv $$
1200: -csh
flags = <none>
  E: basic
  I: basic
  P: basic
  L: all
```

```
% ppriv -v $$
1200: -csh
flags = <none>
  E: file_link_any,proc_exec,proc_fork,proc_info,proc_session
  I: file_link_any,proc_exec,proc_fork,proc_info,proc_session
  P: file_link_any,proc_exec,proc_fork,proc_info,proc_session
  L: cpc_cpu,dtrace_kernel,dtrace_proc,dtrace_user,...,sys_time
```

Determining the Privileges of a Role That You Can Assume

Roles use an administrative shell, or profile shell. You must assume a role and use the role's shell to list the privileges that have been directly assigned to the role. In the following example, the role `sysadmin` has no directly assigned privileges.

```
% su - sysadmin
Password: <Type sysadmin password>
$ /usr/ucb/whoami
sysadmin
```

```
% ppriv -v $$
1400: pfksh
flags = <none>
  E: file_link_any,proc_exec,proc_fork,proc_info,proc_session
  I: file_link_any,proc_exec,proc_fork,proc_info,proc_session
  P: file_link_any,proc_exec,proc_fork,proc_info,proc_session
  L: cpc_cpu,dtrace_kernel,dtrace_proc,dtrace_user,...,sys_time
```
How to Determine Which Privileges a Program Requires

This procedure determines which privileges a command or process requires to succeed.

Before You Begin

The command or process must have failed for this procedure to work.

1 Type the command that is failing as an argument to the ppriv debugging command.

   % ppriv -eD touch /etc/acct/yearly

touch[5245]: missing privilege "file_dac_write"
   (euid = 130, syscall = 224) needed at zfs_zaccess+0x258
   touch: cannot create /etc/acct/yearly: Permission denied

2 Determine which system call is failing by finding the syscall number in the
   /etc/name_to_sysnum file.

   % grep 224 /etc/name_to_sysnum
   creat64 224

Example 11–3 Using the truss Command to Examine Privilege Use

The truss command can debug privilege use in a regular shell. For example, the following
command debugs the failing touch process:

   % truss -t creat touch /etc/acct/yearly
   creat64("/etc/acct/yearly", 0666)
   Err#13 EACCES [file_dac_write]
   touch: /etc/acct/yearly cannot create

The extended /proc interfaces report the missing privilege after the error code in truss output.

Example 11–4 Using the ppriv Command to Examine Privilege Use in a Profile Shell

The ppriv command can debug privilege use in a profile shell. If you assign a rights profile to a
user, and the rights profile includes commands with privileges, the commands must be typed in
a profile shell. When the privileged commands are typed in a regular shell, the commands do
not execute with privilege.

In this example, the jdoe user can assume the role objadmin. The objadmin role includes the
Object Access Management rights profile. This rights profile allows the objadmin role to change
permissions on files that objadmin does not own.

In the following excerpt, jdoe fails to change the permissions on the useful.script file:
When jdoe assumes the objadmin role, the permissions on the file are changed:

```
$ su - objadmin
Password: <Type objadmin password>
$ ls -l useful.script
-rw-r--r-- 1 aloe staff 2303 Apr 10 10:10 useful.script
$ chown objadmin useful.script
$ ls -l useful.script
-rw-r--r-- 1 objadmin staff 2303 Apr 10 10:10 useful.script
$ chgrp admin useful.script
$ ls -l useful.script
-rw-r--r-- 1 objadmin admin 2303 Apr 10 10:11 useful.script
```

### Example 11–5 Changing a File Owned by the root User

This example illustrates the protections against privilege escalation. For a discussion, see “Prevention of Privilege Escalation” on page 239. The file is owned by the root user. The less powerful role, objadmin role needs all privileges to change the file's ownership, so the operation fails.

```
jdoe% su - objadmin
Password: <Type objadmin password>
$ cd /etc; ls -l system
-rw-r--r-- 1 root sys 1883 Oct 10 10:20 system
$ chown objadmin system
chown: system: Not owner
$ ppriv -eD chown objadmin system
chown[11481]: missing privilege "ALL"
    (euid = 101, syscall = 16) needed at zfs_zaccess+0x258
chown: system: Not owner
```

### How to Add Privileges to a Command

You add privileges to a command when you are adding the command to a rights profile. The privileges enable the role that includes the rights profile to run the administrative command, while not gaining any other superuser capabilities.
The command or program must be privilege-aware. For a fuller discussion, see “How Processes Get Privileges” on page 169.

1 Become superuser or assume an equivalent role.
Roles contain authorizations and privileged commands. For more information about roles, see “Configuring RBAC (Task Map)” on page 174.

2 Open the Solaris Management Console GUI.
For instructions, see “How to Assume a Role in the Solaris Management Console” on page 190.

3 Use the Rights tool to update an appropriate profile.
Select the command to include. For each included command, add the privileges that the command requires.

Caution – When you include commands in a rights profile and add privileges to the commands, the commands execute with those privileges when the commands are run in a profile shell.

The order of profiles is important. The profile shell executes a command or action with the security attributes that are specified in the earliest profile in the account's list of profiles. For example, if the chgrp command is in the Object Access Management rights profile with privileges, and Object Access Management is the first profile in which the chgrp command is found, then the chgrp command executes with the privileges specified in the Object Access Management profile.

▼ How to Assign Privileges to a User or Role

You might trust some users with a particular privilege all the time. Very specific privileges that affect a small part of the system are good candidates for assigning to a user. For a discussion of the implications of directly assigned privileges, see “Security Considerations When Directly Assigning Security Attributes” on page 163.

The following procedure enables user jdoe to use high resolution timers.

1 Assume the Primary Administrator role, or become superuser.
The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2 Add the privilege that affects high resolution times to the user's initial inheritable set of privileges.

```bash
$ usermod -K defaultpriv=basic,proc_clock_highres jdoe
```
3 Read the resulting `user_attr` entry.

```
$ grep jdoe /etc/user_attr
jdoe::::type=normal;defaultpriv=basic,proc_clock_highres
```

**Example 11–6** Creating a Role With Privileges to Configure System Time

In this example, a role is created whose only task is to handle time on the system.

```
$ /usr/sadm/bin/smrole -D nisplus:/examplehost/example.domain \ 
  -r primaryadm -l <Type primaryadm password> \ 
  add -- -n clockmgr \ 
  -c "Role that sets system time" \ 
  -F "Clock Manager" \ 
  -s /bin/pfksh \ 
  -u 108 \ 
  -P <Type clockmgr password> \ 
  -K defaultpriv=basic,proc_prioctl,sys_cpu_config, \ 
  proc_clock_highres,sys_time
```

The `-K` line is wrapped for display purposes.

If the role was created locally, the `user_attr` entry for the role would appear similar to the following:

```
clockmgr::::Role that sets system time: 
type=role;defaultpriv=basic,proc_prioctl,sys_cpu_config, \ 
proc_clock_highres,sys_time
```

▼ **How to Limit a User's or Role's Privileges**

You can limit the privileges that are available to a user or role by reducing the basic set, or by reducing the limit set. You should have good reason to limit the user's privileges in this way, because such limitations can have unintended side effects.

**Caution** – You should thoroughly test any user’s capabilities where the basic set or the limit set has been modified for a user.

- When the basic set is less than the default, users can be prevented from using the system.
- When the limit set is less than all privileges, processes that need to run with an effective `UID=0` might fail.

1 Determine the privileges in a user's basic set and limit set.

For the procedure, see "How to Determine the Privileges on a Process" on page 222.
2  (Optional) Remove one of the privileges from the basic set.

$ usermod -K defaultpriv=basic,!priv-name username

By removing the proc_session privilege, you prevent the user from examining any processes outside the user’s current session. By removing the file_link_any privilege, you prevent the user from making hard links to files that are not owned by the user.

---

Caution – Do not remove the proc_fork or the proc_exec privilege. Without these privileges, the user would not be able to use the system. In fact, these two privileges are only reasonably removed from daemons that should not fork() or exec() other processes.

3  (Optional) Remove one of the privileges from the limit set.

$ usermod -K limitpriv=all,!priv-name username

4  Test the capabilities of username.

Log in as username and try to perform the tasks that username must perform on the system.

---

Example 11–7  Removing Privileges From a User’s Limit Set

In the following example, all sessions that originate from jdoe’s initial login are prevented from using the sys_linkdir privilege. That is, the user cannot make hard links to directories, nor can the user unlink directories, even after the user runs the su command.

$ usermod -K limitpriv=all,!sys_linkdir jdoe
$ grep jdoe /etc/user_attr
jdoe:::type=normal;defaultpriv=basic;limitpriv=all,!sys_linkdir

Example 11–8  Removing Privileges From a User’s Basic Set

In the following example, all sessions that originate from jdoe’s initial login are prevented from using the proc_session privilege. That is, the user cannot examine any processes outside the user’s session, even after the user runs the su command.

$ usermod -K defaultpriv=basic,!proc_session jdoe

$ grep jdoe /etc/user_attr
jdoe:::type=normal;defaultpriv=basic,!proc_session;limitpriv=all
How to Run a Shell Script With Privileged Commands

Note – When you create a shell script that runs commands with inherited privileges, the appropriate rights profile must contain the commands with privileges assigned to them.

1 Start the script with /bin/pfsh, or any other profile shell, on the first line.

```
#!/bin/pfsh
# Copyright (c) 2003 by Sun Microsystems, Inc.
```

2 Determine the privileges that the commands in the script need.

```
% ppriv -eD script-full-path
```

3 Open the Solaris Management Console GUI.

For instructions, see "How to Assume a Role in the Solaris Management Console" on page 190. Choose a role, such as Primary Administrator, that can create a rights profile.

4 Use the Rights tool to create or update an appropriate profile.

Select the script, and include in the rights profile each of the commands in the shell script that need privileges to run. For each included command, add the privileges that the command requires.

Caution – The order of rights profiles is important. The profile shell executes the earliest instance of a command in the list of profiles. For example, if the chgrp command is in the Object Access Management rights profile, and Object Access Management is the first profile in which the chgrp command is found, then the chgrp command executes with the privileges that are specified in the Object Access Management profile.

5 Add the rights profile to a role and assign the role to a user.

To execute the profile, the user assumes the role and runs the script in the role's profile shell.

Determining Your Privileges (Task Map)

The following task map points to procedures for using the privileges that have been assigned to you.
## Determining Your Assigned Privileges

When a user is directly assigned privileges, the privileges are in effect in every shell. When a user is not directly assigned privileges, then the user must open a profile shell. For example, when commands with assigned privileges are in a rights profile that is in the user’s list of rights profiles, then the user must execute the command in a profile shell.

### ▼ How to Determine the Privileges That You Have Been Directly Assigned

The following procedure shows how to determine if you have been directly assigned privileges.

**Caution** – Inappropriate use of directly assigned privileges can result in unintentional breaches of security. For a discussion, see “Security Considerations When Directly Assigning Security Attributes” on page 163.

1. **List the privileges that your processes can use.**
   See “How to Determine the Privileges on a Process” on page 222 for the procedure.

2. **Invoke actions and run commands in any shell.**
   The privileges that are listed in the effective set are in effect throughout your session. If you have been directly assigned privileges in addition to the basic set, the privileges are listed in the effective set.

### Example 11–9 Determining Your Directly-Assigned Privileges

If you have been directly assigned privileges, then your basic set contains more than the default basic set. In this example, the user always has access to the `proc_clock_highres` privilege.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>View your privileges as a user in any shell</td>
<td>Shows the privileges that have been directly assigned to you. All of your processes run with these privileges.</td>
<td>“How to Determine the Privileges That You Have Been Directly Assigned” on page 230</td>
</tr>
<tr>
<td>Determine which commands you can run with privilege</td>
<td>When privileges are assigned to executables in a rights profile, the executable must be typed in a profile shell.</td>
<td>“How to Determine the Privileged Commands That You Can Run” on page 231</td>
</tr>
<tr>
<td>Determine which commands a role can run with privileges</td>
<td>Assumes the role to determine which commands the role can run with privileges.</td>
<td>“How to Determine the Privileged Commands That a Role Can Run” on page 233</td>
</tr>
</tbody>
</table>
Determining a Role's Directly-Assigned Privileges

Roles use an administrative shell, or profile shell. Users who assume a role can use the role's shell to list the privileges that have been directly assigned to the role. In the following example, the role `realtime` has been directly assigned privileges to handle date and time programs.

```
% su - realtime
Password: <Type realtime password>
```

```
$ /usr/ucb/whoami
realtime
```

```
$ ppriv -v $$
1600: pfksh
flags = <none>
    E: file_link_any,...,proc_clock_highres,proc_session,sys_time
    I: file_link_any,...,proc_clock_highres,proc_session,sys_time
    P: file_link_any,...,proc_clock_highres,proc_session,sys_time
    L: cpc_cpu,dtrace_kernel,dtrace_proc,dtrace_user,...,sys_time
```

```
% ppriv -vl proc_clock_highres
Allows a process to use high resolution timers.
```

▼ How to Determine the Privileged Commands That You Can Run

When a user is not directly assigned privileges, then the user gets access to privileged commands through a rights profile. Commands in a rights profile must be executed in a profile shell.

Before You Begin

The user or role who authenticates to the Solaris Management Console must have the `solaris.admin.usermgr.read` authorization. The Basic Solaris User rights profile includes this authorization.

1. Determine the rights profiles that you have been assigned.

```
$ /usr/sadm/bin/smuser list -- -n username -l
```

Authenticating as user: admin

... Please enter a string value for: password ::
Determining Your Assigned Privileges

... 
User name: username
User ID (UID): 130
Primary group: staff
Secondary groups:
Comment: object mgt jobs
Login Shell: /bin/sh
Home dir server: system
Home directory: /export/home/username
AutoHome setup: True
Mail server: system
Rights: Object Access Management
Assigned Roles:

2 Locate the line that begins with “Rights:”.
The “Rights” line lists the names of the rights profiles that have been directly assigned to you.

3 Find the names of the rights profiles in the exec_attr database.
   $ cd /etc/security
   $ grep "Object Access Management" exec_attr
   Object Access Management:solaris:cmd:::/usr/bin/chgrp:privs=file_chown
   Object Access Management:solaris:cmd:::/usr/bin/chown:privs=file_chown
   Object Access Management:suser:cmd:::/usr/bin/chgrp:euid=0
   Object Access Management:suser:cmd:::/usr/bin/chmod:euid=0
   ...
   The commands with added privileges are listed at the end of solaris policy entries.

4 Type the commands that require privileges in a profile shell.
   When the commands are typed in a regular shell, the commands do not run with privilege, and do not succeed.
   % pfsh
   $

Example 11–11 Running Privileged Commands in a Profile Shell

In the following example, the user jdoe cannot change the group permissions on a file from his regular shell. However, jdoe can change the permissions when typing the command in a profile shell.

   % whoami
   jdoe
   % ls -l useful.script
   -rwxr-xr-x 1 jdoe  eng 262 Apr 2 10:52 useful.script
   chgrp staff useful.script
   chgrp: useful.script: Not owner

   % pfsh
   $
Determining Your Assigned Privileges

How to Determine the Privileged Commands That a Role Can Run

A role gets access to privileged commands through a rights profile that contains commands with assigned privileges. The most secure way to provide a user with access to privileged commands is to assign a role to them. After assuming the role, the user can execute all the privileged commands that are included in the rights profiles for that role.

Before You Begin

The user or role who authenticates to the Solaris Management Console must have the solaris.admin.usermgr.read authorization. The Basic Solaris User rights profile includes this authorization.

1 Determine the roles that you can assume.

$ /usr/sadm/bin/smuser list -- -n username -l
Authenticating as user: primadmin

...  
User name:  username
User ID (UID):  110
Primary group:  staff
Secondary groups:
  Comment:  Has admin roles
Login Shell:  /bin/sh
...
Rights:
  Assigned Roles:  primadmin, admin

2 Locate the line that begins with “Assigned Roles:”.

The “Assigned Roles” line lists the roles that you can assume.

3 Determine the rights profiles that are included in one of your roles.

$ /usr/sadm/bin/smuser list -- -n admin -l
Authenticating as user: primadmin

...  
User name:  admin
User ID (UID):  101
Primary group: sysadmin
Secondary groups:
Comment: system administrator
Login Shell: /bin/pfksh

Rights: System Administrator

4 Locate the names of the rights profiles for the role in the “Rights:” line.

5 Find the rights profiles in the prof_attr database.
Because the System Administrator profile is a collection of profiles, you need to list the profiles in the System Administrator profile.

$ cd /etc/security
$ grep "System Administrator" prof_attr
System Administrator:::Can perform most non-security administrative
tasks:profiles=Audit Review,Printer Management,Cron Management,
Device Management,File System Management,Mail Management,Maintenance
and Repair,Media Backup,Media Restore,Name Service Management,Network
Management,Object Access Management,Process Management,Software
Installation,User Management,All;help=RtSysAdmin.html

6 For each rights profile, find the rights profiles in the exec_attr database.
For example, the Network Management profile is a supplementary profile of the System Administrator profile. The Network Management profile includes a number of privileged commands.

$ cd /etc/security
$ grep "Network Management" exec_attr
Network Management:solaris:cmd:::/usr/sbin/ifconfig:privs=sys_net_config
Network Management:solaris:cmd:::/usr/sbin/route:privs=sys_net_config

The commands and their assigned privileges are the final two fields of solaris policy entries. You can run these commands in the profile shell of your role.

Example 11–12 Running the Privileged Commands in Your Role
When a user assumes a role, the shell becomes a profile shell. Therefore, the commands are executed with the privileges that were assigned to the commands. In the following example, the admin role can change the permissions on the useful.script file.

% whoami
jdoe
% ls -l useful.script
-rw-r-xr-x 1 elsee eng 262 Apr 2 10:52 useful.script
% chgrp admin useful.script
chgrp: useful.script: Not owner
% su - admin
Password: <Type admin password>
$ /usr/ucb/whoami
admin
$ chgrp admin useful.script
$ chown admin useful.script
$ ls -l useful.script
-rwxr-xr-- 1 admin admin 262 Apr 2 10:53 useful.script
The following is a list of the reference information in this chapter:

- “Administrative Commands for Handling Privileges” on page 237
- “Files With Privilege Information” on page 238
- “Privileges and Auditing” on page 239
- “Prevention of Privilege Escalation” on page 239
- “Legacy Applications and the Privilege Model” on page 240

To use privileges, see Chapter 11, “Privileges (Tasks).” For overview information, see “Privileges (Overview)” on page 164.

## Administrative Commands for Handling Privileges

The following table lists the commands that are available to handle privileges.

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Command</th>
<th>Man Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Examine process privileges</td>
<td><code>ppriv -V pid</code></td>
<td><code>ppriv(1)</code></td>
</tr>
<tr>
<td>Set process privileges</td>
<td><code>ppriv -s spec</code></td>
<td></td>
</tr>
<tr>
<td>List the privileges on the system</td>
<td><code>ppriv -l</code></td>
<td></td>
</tr>
<tr>
<td>List a privilege and its description</td>
<td><code>ppriv -lv priv</code></td>
<td></td>
</tr>
<tr>
<td>Debug privilege failure</td>
<td><code>ppriv -ed failed-operation</code></td>
<td></td>
</tr>
<tr>
<td>Assign privileges to a new local user</td>
<td><code>useradd</code></td>
<td><code>useradd(1M)</code></td>
</tr>
<tr>
<td>Add privileges to an existing local user</td>
<td><code>usermod</code></td>
<td><code>usermod(1M)</code></td>
</tr>
<tr>
<td>Assign privileges to a user in a name service</td>
<td><code>smuser</code></td>
<td><code>smuser(1M)</code></td>
</tr>
</tbody>
</table>
The Solaris Management Console GUI is the preferred tool for assigning privileges to commands, users, and roles. For more information, see "How to Assume a Role in the Solaris Management Console" on page 190.

**Files With Privilege Information**

The following files contain information about privileges.

**TABLE 12-2  Files That Contain Privilege Information**

<table>
<thead>
<tr>
<th>File and Man Page</th>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/security/policy.conf</td>
<td>PRIV_DEFAULT</td>
<td>Inheritable set of privileges for the system</td>
</tr>
<tr>
<td></td>
<td>PRIV_LIMIT</td>
<td>Limit set of privileges for the system</td>
</tr>
<tr>
<td>/etc/user_attr</td>
<td>defaultpriv</td>
<td>Inheritable set of privileges for a user or role</td>
</tr>
<tr>
<td></td>
<td>limitpriv</td>
<td>Limit set of privileges for a user or role</td>
</tr>
<tr>
<td></td>
<td>privs</td>
<td>List of privileges that are assigned to a command in a rights profile</td>
</tr>
</tbody>
</table>

**TABLE 12-1  Commands for Handling Privilege (Continued)**

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Command</th>
<th>Man Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assign privileges to a new local role</td>
<td>roleadd</td>
<td>roleadd(1M)</td>
</tr>
<tr>
<td>Add privileges to an existing local role</td>
<td>rolemod</td>
<td>rolemod(1M)</td>
</tr>
<tr>
<td>Assign privileges to a role in a name service</td>
<td>smrole</td>
<td>smrole(1M)</td>
</tr>
<tr>
<td>View device policy</td>
<td>getdevpolicy</td>
<td>getdevpolicy(1M)</td>
</tr>
<tr>
<td>Set device policy</td>
<td>devfsadm</td>
<td>devfsadm(1M)</td>
</tr>
<tr>
<td>Update device policy on open devices</td>
<td>update_drv -p policy driver</td>
<td>update_drv(1M)</td>
</tr>
<tr>
<td>Add device policy to a device</td>
<td>add_drv -p policy driver</td>
<td>add_drv(1M)</td>
</tr>
</tbody>
</table>
Privileges and Auditing

Privilege use can be audited. Any time that a process uses a privilege, the use of privilege is recorded in the audit trail in the upriv audit token. When privilege names are part of the record, their textual representation is used. The following audit events record use of privilege:

- **AUE_SETPPRIV audit event** – The event generates an audit record when a privilege set is changed. The AUE_SETPPRIV audit event is in the pm class.
- **AUE_MODALLOCPRI Priv audit event** – The audit event generates an audit record when a privilege is added from outside the kernel. The AUE_MODALLOCPRI Priv audit event is in the ad class.
- **AUE_MODDEVPLCY audit event** – The audit event generates an audit record when the device policy is changed. The AUE_MODDEVPLCY audit event is in the ad class.
- **AUE_prof_cmd audit event** – The audit event generates an audit record when a command is executed in a profile shell. The AUE_prof_cmd audit event is in the as and ua audit classes. The names of the privileges are included in the audit record.

The successful use of privileges that are in the basic set is not audited. The attempt to use a basic privilege that has been removed from a user’s basic set is audited.

Prevention of Privilege Escalation

The Solaris kernel prevents *privilege escalation*. Privilege escalation is when a privilege enables a process to do more than the process should be able to do. To prevent a process from gaining more privileges than the process should have, certain system modifications require the full set of privileges. For example, a file or process that is owned by root (UIT=0) can only be changed by a process with the full set of privileges. The root user does not require privileges to change a file that root owns. However, a non-root user must have all privileges in order to change a file that is owned by root.

Similarly, operations that provide access to devices require all privileges in the effective set.
The `file_chown_self` and `proc_owner` privileges are subject to privilege escalation. The `file_chown_self` privilege allows a process to give away its files. The `proc_owner` privilege allows a process to inspect processes that the process does not own.

The `file_chown_self` privilege is limited by the `rstchown` system variable. When the `rstchown` variable is set to zero, the `file_chown_self` privilege is removed from the initial inheritable set of the system and of all users. For more information on the `rstchown` system variable, see the `chown(1)` man page.

The `file_chown_self` privilege is most safely assigned to a particular command, placed in a profile, and assigned to a role for use in a profile shell.

The `proc_owner` privilege is not sufficient to switch a process UID to 0. To switch a process from any UID to UID=0 requires all privileges. Because the `proc_owner` privilege gives unrestricted read access to all files on the system, the privilege is most safely assigned to a particular command, placed in a profile, and assigned to a role for use in a profile shell.

Caution – A user’s account can be modified to include the `file_chown_self` privilege or the `proc_owner` privilege in the user’s initial inheritable set. You should have overriding security reasons for placing such powerful privileges in the inheritable set of privileges for any user, role, or system.

For details of how privilege escalation is prevented for devices, see “Privileges and Devices” on page 171.

Legacy Applications and the Privilege Model

To accommodate legacy applications, the implementation of privileges works with both the superuser and the privilege models. The kernel automatically tracks the `PRIV_AWARE` flag, which indicates that a program has been designed to work with privileges. Consider a child process that is not aware of privileges. Any privileges that were inherited from the parent process are available in the child’s permitted and effective sets. If the child process sets a UID to 0, the child process might not have full superuser capabilities. The process’s effective and permitted sets are restricted to those privileges in the child’s limit set. Thus, the limit set of a privilege-aware process restricts the root privileges of child processes that are not aware of privileges.
Solaris Cryptographic Services

This section describes the centralized cryptographic and public key technology services that the Solaris OS provides.
This chapter describes the Solaris™ Cryptographic Framework. The following is a list of the information in this chapter.

- “Solaris Cryptographic Framework” on page 243
- “Terminology in the Solaris Cryptographic Framework” on page 244
- “Scope of the Solaris Cryptographic Framework” on page 246
- “Administrative Commands in the Solaris Cryptographic Framework” on page 246
- “User-Level Commands in the Solaris Cryptographic Framework” on page 247
- “Plugins to the Solaris Cryptographic Framework” on page 247
- “Cryptographic Services and Zones” on page 248

To administer and use the Solaris Cryptographic Framework, see Chapter 14, “Solaris Cryptographic Framework (Tasks).”

**Solaris Cryptographic Framework**

The Solaris Cryptographic Framework provides a common store of algorithms and PKCS #11 libraries to handle cryptographic requirements. The PKCS #11 libraries are implemented according to the following standard: RSA Security Inc. PKCS #11 Cryptographic Token Interface (Cryptoki).

At the kernel level, the framework currently handles cryptographic requirements for Kerberos and IPsec. User-level consumers include libasl and IKE.

Export law in the United States requires that the use of open cryptographic interfaces be restricted. The Solaris Cryptographic Framework satisfies the current law by requiring that kernel cryptographic providers and PKCS #11 cryptographic providers be signed. For further discussion, see “Binary Signatures for Third-Party Software” on page 247.
The framework enables providers of cryptographic services to have their services used by many consumers in the Solaris Operating System. Another name for providers is plugins. The framework allows three types of plugins:

- **User-level plugins** – Shared objects that provide services by using PKCS #11 libraries, such as pkcs11_softtoken.so.1.
- **Kernel-level plugins** – Kernel modules that provide implementations of cryptographic algorithms in software, such as AES. Many of the algorithms in the framework are optimized for x86 with the SSE2 instruction set and for SPARC hardware.
- **Hardware plugins** – Device drivers and their associated hardware accelerators. The Niagara chips, the ncp and n2cp device drivers, are one example. A hardware accelerator offloads expensive cryptographic functions from the operating system. The Sun Crypto Accelerator 6000 board is one example.

The framework implements a standard interface, the PKCS #11, v2.11 library, for user-level providers. The library can be used by third-party applications to reach providers. Third parties can also add signed libraries, signed kernel algorithm modules, and signed device drivers to the framework. These plugins are added when the pkgadd utility installs the third-party software. For a diagram of the major components of the framework, see Chapter 8, “Introduction to the Solaris Cryptographic Framework,” in Solaris Security for Developers Guide.

### Terminology in the Solaris Cryptographic Framework

The following list of definitions and examples is useful when working with the cryptographic framework.

- **Algorithms** – Cryptographic algorithms. These are established, recursive computational procedures that encrypt or hash input. Encryption algorithms can be symmetric or asymmetric. Symmetric algorithms use the same key for encryption and decryption. Asymmetric algorithms, which are used in public-key cryptography, require two keys. Hashing functions are also algorithms.

  Examples of algorithms include:
  - Symmetric algorithms, such as AES and ARCFOUR
  - Asymmetric algorithms, such as Diffie-Hellman and RSA
  - Hashing functions, such as MD5

- **Consumers** – Are users of the cryptographic services that come from providers. Consumers can be applications, end users, or kernel operations.

  Examples of consumers include:
  - Applications, such as IKE
  - End users, such as an ordinary user who runs the encrypt command
Kernel operations, such as IPsec

**Mechanism** – Is the application of a mode of an algorithm for a particular purpose. For example, a DES mechanism that is applied to authentication, such as CKM_DES_MAC, is a separate mechanism from a DES mechanism that is applied to encryption, CKM_DES_CBC_PAD.

**Metaslot** – Is a single slot that presents a union of the capabilities of other slots which are loaded in the framework. The metaslot eases the work of dealing with all of the capabilities of the providers that are available through the framework. When an application that uses the metaslot requests an operation, the metaslot figures out which actual slot should perform the operation. Metaslot capabilities are configurable, but configuration is not required. The metaslot is on by default. To configure the metaslot, see the `cryptoadm(1M)` man page.

**Mode** – Is a version of a cryptographic algorithm. For example, CBC (Cipher Block Chaining) is a different mode from ECB (Electronic Code Book). The AES algorithm has two modes, CKM_AES_ECB and CKM_AES_CBC.

**Policy** – Is the choice, by an administrator, of which mechanisms to make available for use. By default, all providers and all mechanisms are available for use. The disabling of any mechanism would be an application of policy. The enabling of a disabled mechanism would also be an application of policy.

**Providers** – Are cryptographic services that consumers use. Providers plug in to the framework, so are also called plugins.

Examples of providers include:

- PKCS #11 libraries, such as `pkcs11_softoken.so`
- Modules of cryptographic algorithms, such as `aes` and `arcfour`
- Device drivers and their associated hardware accelerators, such as the `mca` driver for the Sun Crypto 6000 accelerator

**Slot** – Is an interface to one or more cryptographic devices. Each slot, which corresponds to a physical reader or other device interface, might contain a token. A token provides a logical view of a cryptographic device in the framework.

**Token** – In a slot, a token provides a logical view of a cryptographic device in the framework.
Scope of the Solaris Cryptographic Framework

The framework provides commands for administrators, for users, and for developers who supply providers:

- **Administrative commands** – The cryptoadm command provides a list subcommand to list the available providers and their capabilities. Ordinary users can run the cryptoadm list and the cryptoadm --help commands.

  All other cryptoadm subcommands require you to assume a role that includes the Crypto Management rights profile, or to become superuser. Subcommands such as disable, install, and uninstall are available for administering the framework. For more information, see the cryptoadm(1M) man page.

  The svcadm command is used to manage the kcfd daemon, and to refresh cryptographic policy in the kernel. For more information, see the svcadm(1M) man page.

- **User-level commands** – The digest and mac commands provide file integrity services. The encrypt and decrypt commands protect files from eavesdropping. To use these commands, see “Protecting Files With the Solaris Cryptographic Framework (Task Map)” on page 250.

- **Binary signatures for third-party providers** – The elfsign command enables third parties to sign binaries for use within the framework. Binaries that can be added to the framework are PKCS #11 libraries, kernel algorithm modules, and hardware device drivers. To use the elfsign command, see Appendix F, “Packaging and Signing Cryptographic Providers,” in Solaris Security for Developers Guide.

Administrative Commands in the Solaris Cryptographic Framework

The cryptoadm command administers a running cryptographic framework. The command is part of the Crypto Management rights profile. This profile can be assigned to a role for secure administration of the cryptographic framework. The cryptoadm command manages the following:

- Displaying cryptographic provider information
- Disabling or enabling provider mechanisms
- Solaris 10 1/06: Disabling or enabling the metaslot

The svcadm command is used to enable, refresh, and disable the cryptographic services daemon, kcfd. This command is part of the Solaris service management facility, smf. svc:/system/cryptosvcs is the service instance for the cryptographic framework. For more information, see the smf(5) and svcadm(1M) man pages.
User-Level Commands in the Solaris Cryptographic Framework

The Solaris Cryptographic Framework provides user-level commands to check the integrity of files, to encrypt files, and to decrypt files. A separate command, `elfsign`, enables providers to sign binaries for use with the framework.

- **digest command** – Computes a message digest for one or more files or for stdin. A digest is useful for verifying the integrity of a file. SHA1 and MD5 are examples of digest functions.

- **mac command** – Computes a message authentication code (MAC) for one or more files or for stdin. A MAC associates data with an authenticated message. A MAC enables a receiver to verify that the message came from the sender and that the message has not been tampered with. The `sha1_mac` and `md5_hmac` mechanisms can compute a MAC.

- **encrypt command** – Encrypts files or stdin with a symmetric cipher. The `encrypt -l` command lists the algorithms that are available. Mechanisms that are listed under a user-level library are available to the `encrypt` command. The framework provides AES, DES, 3DES (Triple-DES), and ARCFOUR mechanisms for user encryption.

- **decrypt command** – Decrypts files or stdin that were encrypted with the `encrypt` command. The `decrypt` command uses the identical key and mechanism that were used to encrypt the original file.

Binary Signatures for Third-Party Software

The `elfsign` command provides a means to sign providers to be used with the Solaris Cryptographic Framework. Typically, this command is run by the developer of a provider.

The `elfsign` command has subcommands to request a certificate from Sun and to sign binaries. Another subcommand verifies the signature. Unsigned binaries cannot be used by the Solaris Cryptographic Framework. To sign one or more providers requires the certificate from Sun and the private key that was used to request the certificate. For more information, see Appendix F, “Packaging and Signing Cryptographic Providers,” in Solaris Security for Developers Guide.

Plugins to the Solaris Cryptographic Framework

Third parties can plug their providers into the Solaris Cryptographic Framework. A third-party provider can be one of the following objects:

- **PKCS #11 shared library**
- **Loadable kernel software module, such as an encryption algorithm, MAC function, or digest function**
Kernel device driver for a hardware accelerator

The objects from a provider must be signed with a certificate from Sun. The certificate request is based on a private key that the third party selects, and a certificate that Sun provides. The certificate request is sent to Sun, which registers the third party and then issues the certificate. The third party then signs its provider object with the certificate from Sun.

The loadable kernel software modules and the kernel device drivers for hardware accelerators must also register with the kernel. Registration is through the Solaris Cryptographic Framework SPI (service provider interface).

To install the provider, the third party provides a package that installs the signed object and the certificate from Sun. The package must include the certificate, and enable the administrator to place the certificate in a secure directory. For more information, see the Appendix F, "Packaging and Signing Cryptographic Providers," in Solaris Security for Developers Guide.

Cryptographic Services and Zones

The global zone and each non-global zone has its own /system/cryptosvc service. When the cryptographic service is enabled or refreshed in the global zone, the kcfd daemon starts in the global zone, user-level policy for the global zone is set, and kernel policy for the system is set. When the service is enabled or refreshed in a non-global zone, the kcfd daemon starts in the zone, and user-level policy for the zone is set. Kernel policy was set by the global zone.

For more information on zones, see Part II, “Zones,” in System Administration Guide: Virtualization Using the Solaris Operating System. For more information on the service management facility that manages persistent applications, see Chapter 16, "Managing Services (Overview)," in System Administration Guide: Basic Administration and the smf(5) man page.
Solaris Cryptographic Framework (Tasks)

This chapter describes how to use the Solaris Cryptographic Framework. The following is a list of information in this chapter.
- “Using the Cryptographic Framework (Task Map)” on page 249
- “Protecting Files With the Solaris Cryptographic Framework (Task Map)” on page 250
- “Administering the Cryptographic Framework (Task Map)” on page 263

Using the Cryptographic Framework (Task Map)

The following task map points to tasks for using the cryptographic framework.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protect individual files or sets of files</td>
<td>Ensures that file content has not been tampered with. Prevents files from being read by intruders. These procedures can be done by ordinary users.</td>
<td>“Protecting Files With the Solaris Cryptographic Framework (Task Map)” on page 250</td>
</tr>
<tr>
<td>Administer the framework</td>
<td>Adds, configures, and removes software providers. Disables and enables hardware provider mechanisms. These procedures are administrative procedures.</td>
<td>“Administering the Cryptographic Framework (Task Map)” on page 263</td>
</tr>
<tr>
<td>Sign a provider</td>
<td>Enables a provider to be added to the Solaris Cryptographic Framework. These procedures are developer procedures.</td>
<td>Appendix F, “Packaging and Signing Cryptographic Providers,” in Solaris Security for Developers Guide.</td>
</tr>
</tbody>
</table>
Protecting Files With the Solaris Cryptographic Framework
(Task Map)

The Solaris Cryptographic Framework can help you protect your files. The following task map points to procedures for listing the available algorithms, and for protecting your files cryptographically.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate a symmetric key</td>
<td>Generates a random key for use with algorithms that the user specifies.</td>
<td>&quot;How to Generate a Symmetric Key by Using the dd Command&quot; on page 250</td>
</tr>
<tr>
<td></td>
<td>Generates a key of user-specified length. Optionally, stores the key in a file, a PKCS #11 keystore, or an NSS keystore.</td>
<td>&quot;How to Generate a Symmetric Key by Using the pktool Command&quot; on page 252</td>
</tr>
<tr>
<td>Provide a checksum that ensures the integrity of a file</td>
<td>Verifies that the receiver’s copy of a file is identical to the file that was sent.</td>
<td>&quot;How to Compute a Digest of a File&quot; on page 257</td>
</tr>
<tr>
<td>Protect a file with a message authentication code (MAC)</td>
<td>Verifies to the receiver of your message that you were the sender.</td>
<td>&quot;How to Compute a MAC of a File&quot; on page 258</td>
</tr>
<tr>
<td>Encrypt a file, and then decrypt the encrypted file</td>
<td>Protects the content of files by encrypting the file. Provides the encryption parameters to decrypt the file.</td>
<td>&quot;How to Encrypt and Decrypt a File&quot; on page 260</td>
</tr>
</tbody>
</table>

Protecting Files With the Solaris Cryptographic Framework

This section describes how to generate symmetric keys, how to create checksums for file integrity, and how to protect files from eavesdropping. The commands in this section can be run by regular users. Developers can write scripts that use these commands.

▼ How to Generate a Symmetric Key by Using the dd Command

A key is needed to encrypt files and to generate the MAC of a file. The key should be derived from a random pool of numbers. To create the key, you have three options:

- If your site has a random number generator, use the generator.
- If you want to generate the key and store it, see "How to Generate a Symmetric Key by Using the pktool Command" on page 252.
- Otherwise, use this procedure. This procedure requires that you provide the key size in bites. In contrast, the pktool command determines the correct key size according to the algorithm that you specify.
1. **Determine the key length that your algorithm requires.**

   a. **List the available algorithms.**

   ```
   % encrypt -l
   Algorithm Keysize: MinMax (bits)
   -------------------------------
aes 128 128
arcfour 8 128
des 64 64
3des 192 192
   
   % mac -l
   Algorithm Keysize: MinMax (bits)
   -------------------------------
des_mac 64 64
sha1_hmac 8 512
md5_hmac 8 512
sha256_hmac 8 1024
sha384_hmac 8 1024
   ```

   b. **Determine the key length in bytes to pass to the dd command.**

   Divide the minimum and maximum key sizes by 8. When the minimum and maximum key sizes are different, intermediate key sizes are possible. For example, the value 8, 16, or 64 can be passed to the `dd` command for the `sha1_hmac` and `md5_hmac` functions.

2. **Generate the symmetric key.**

   ```
   % dd if=/dev/urandom of=keyfile bs=n count=n
   
   if=file Is the input file. For a random key, use the /dev/urandom file.
of=keyfile Is the output file that holds the generated key.
bs=n Is the key size in bytes. For the length in bytes, divide the key length in bits by 8.
count=n Is the count of the input blocks. The number for n should be 1.
   ```

3. **Store your key in a protected directory.**

   The key file should not be readable by anyone but the user.

   ```
   % chmod 400 keyfile
   ```

**Example 14–1 Creating a Key for the AES Algorithm**

In the following example, a secret key for the AES algorithm is created. The key is also stored for later decryption. AES mechanisms use a 128-bit key. The key is expressed as 16 bytes in the `dd` command.
Example 14–2 Creating a Key for the DES Algorithm

In the following example, a secret key for the DES algorithm is created. The key is also stored for later decryption. DES mechanisms use a 64-bit key. The key is expressed as 8 bytes in the dd command.

```
% dd if=/dev/urandom of=$HOME/keyf/05.07.aes16 bs=16 count=1
% chmod 400 ~/keyf/05.07.aes16
```

Example 14–3 Creating a Key for the 3DES Algorithm

In the following example, a secret key for the 3DES algorithm is created. The key is also stored for later decryption. 3DES mechanisms use a 192-bit key. The key is expressed as 24 bytes in the dd command.

```
% dd if=/dev/urandom of=$HOME/keyf/05.07.3des24 bs=24 count=1
% chmod 400 ~/keyf/05.07.3des24
```

Example 14–4 Creating a Key for the MD5 Algorithm

In the following example, a secret key for the MD5 algorithm is created. The key is also stored for later decryption. The key is expressed as 64 bytes in the dd command.

```
% dd if=/dev/urandom of=$HOME/keyf/05.07.mack64 bs=64 count=1
% chmod 400 ~/keyf/05.07.mack64
```

How to Generate a Symmetric Key by Using the pktool Command

Some applications require a symmetric key for encryption and decryption of communications. In this procedure, you create a symmetric key and store it.

- If your site has a random number generator, you can use the generator to create a random number for the key. This procedure does not use your site’s random number generator.
- You can instead use the dd command with the Solaris /dev/urandom device as input. The dd command does not store the key. For the procedure, see “How to Generate a Symmetric Key by Using the dd Command” on page 250.
1  **(Optional) If you plan to use a keystore, create it.**
   - To create and initialize a PKCS #11 keystore, see "How to Generate a Passphrase by Using the pktool setpin Command" on page 286.
   - To create and initialize an NSS database, see Example 15–5.

2  **Generate a random number for use as a symmetric key.**
   Use one of the following methods.
   - **Generate a key and store it in a file.**
     The advantage of a file-stored key is that you can extract the key from this file for use in an application's key file, such as the /etc/inet/secret/ipseckeys file or IPsec.
     
     ```
     \% pktool genkey keytype=file outkey=key-fn \\
     [keytype=generic|specific-symmetric-algorithm] [keylen=size-in-bits] \\
     [dir=directory] [print=n]
     ```
     
     **keystore**
     
     The value file specifies the file type of storage location for the key.
     
     **outkey=key-fn**
     
     Is the filename when keytype=file.
     
     **keytype=specific-symmetric-algorithm**
     
     For a symmetric key of any length, the value is generic. For a particular algorithm, specify aes, arcfour, des, or 3des.
     
     **keylen=size-in-bits**
     
     Is the length of the key in bits. The number must be divisible by 8. Do not specify for des or 3des.
     
     **dir=directory**
     
     Is the directory path to key-fn. By default, directory is the current directory.
     
     **print=n**
     
     Prints the key to the terminal window. By default, the value of print is n.
   
   - **Generate a key and store it in a PKCS #11 keystore.**
     The advantage of the PKCS #11 keystore is that you can retrieve the key by its label. This method is useful for keys that encrypt and decrypt files. You must complete Step 1 before using this method.
     
     ```
     \% pktool genkey label=key-label \\
     [keytype=generic|specific-symmetric-algorithm] [keylen=size-in-bits] \\
     [token=token] [sensitive=n] [extractable=y] [print=n]
     ```
     
     **label=key-label**
     
     Is a user-specified label for the key. The key can be retrieved from the keystore by its label.
keytype=specific-symmetric-algorithm
For a symmetric key of any length, the value is generic. For a particular algorithm,
specify aes, arcfour, des, or 3des.

keylen=size-in-bits
Is the length of the key in bits. The number must be divisible by 8. Do not specify for des
or 3des.

token=token
Is the token name. By default, the token is Sun Software PKCS#11 softtoken.
sensitive=n
Specifies the sensitivity of the key. When the value is y, the key cannot be printed by using
the print=y argument. By default, the value of sensitive is n.
extractable=y
Specifies that the key can be extracted from the keystore. Specify n to prevent the key
from being extracted.
print=n
Prints the key to the terminal window. By default, the value of print is n.

Generate a key and store it in an NSS keystore.
You must complete Step 1 before using this method.
% pktool keystore=nss genkey label=key-label \[keytype=\{keytype=generic|specific-symmetric-algorithm\} [keylen=size-in-bits] [token=token] \[dir=directory-path] [prefix=database-prefix]\] keystore
The value nss specifies the NSS type of storage location for the key.

label=key-label
Is a user-specified label for the key. The key can be retrieved from the keystore by its label.

keytype=specific-symmetric-algorithm
For a symmetric key of any length, the value is generic. For a particular algorithm,
specify aes, arcfour, des, or 3des.

keylen=size-in-bits
Is the length of the key in bits. The number must be divisible by 8. Do not specify for des
or 3des.

token=token
Is the token name. By default, the token is the NSS internal token.

dir=directory
Is the directory path to the NSS database. By default, directory is the current directory.

prefix=directory
Is the prefix to the NSS database. The default is no prefix.
print=n
Prints the key to the terminal window. By default, the value of print is n.

3 (Optional) Verify that the key exists.
Use one of the following commands, depending on where you stored the key.

- **Verify the key in the key-fn file.**
  
  % pktool list keystore=file objtype=key infile=key-fn
  Found n keys.
  Key #1 - keytype:location (keylen)

- **Verify the key in the PKCS #11 or the NSS keystore.**
  
  $ pktool list objtype=key
  Enter PIN for keystore:
  Found n keys.
  Key #1 - keytype:location (keylen)

**Example 14–5 Creating a Symmetric Key by Using the pktool Command**

In the following example, a user creates a PKCS #11 keystore for the first time, and then generates a large symmetric key for an application. Finally, the user verifies that the key is in the keystore.

```bash
# pktool setpin
Create new passphrase: easily-remembered-hard-to-detect-password
Re-enter new passphrase: Retype password
Passphrase changed.
% pktool genkey label=specialappkey keytype=generic keylen=1024
Enter PIN for Sun Software PKCS#11 softtoken : Type password

% pktool list objtype=key
Enter PIN for Sun Software PKCS#11 softtoken : Type password
Found 1 keys.
Key #1 - symmetric: specialappkey (1024 bits)
```

**Example 14–6 Creating a DES Key by Using the pktool Command**

In the following example, a secret key for the DES algorithm is created. The key is stored in a local file for later decryption. The command protects the file with 400 permissions. When the key is created, the print=y option displays the generated key in the terminal window.

DES mechanisms use a 64-bit key. The user who owns the keyfile retrieves the key by using the od command.
Example 14-7  Creating a Symmetric Key for IPsec Security Associations

In the following example, the administrator manually creates the keying material for IPsec SAs and stores them in files. Then, the administrator copies the keys to the /etc/inet/secret/ipseckeys file and destroys the original files.

First, the administrator creates and displays the keys that the IPsec policy requires:

```
# pktool genkey keystore=file outkey=ipencrin1 keytype=generic keylen=192 print=y
  Key Value ="294979e512cb8e79370dabec3fcfb849e78d2d6bd2049"
# pktool genkey keystore=file outkey=ipencrout1 keytype=generic keylen=192 print=y
  Key Value ="9678f80e33406c86e3d1686e50406bd0434819c20d09d204"  
# pktool genkey keystore=file outkey=ipspi1 keytype=generic keylen=32 print=y
  Key Value ="acbeaa20"  
# pktool genkey keystore=file outkey=ipspi2 keytype=generic keylen=32 print=y
  Key Value ="19174215" 
# pktool genkey keystore=file outkey=ipmd51 keytype=generic keylen=64 print=y
  Key Value ="438c3ad2cecc9a3621e90462d11ca7d2f" 
# pktool genkey keystore=file outkey=ipmd52 keytype=generic keylen=64 print=y
  Key Value ="a61319630cf2abde7609ce24de3d029f" 
```

Then, the administrator creates the following /etc/inet/secret/ipseckeys file:

```
## SPI values require a leading 0x.
## Backslashes indicate command continuation.
##
## for outbound packets on this system
add esp spi 0xacbeaa20 \
  src 192.168.1.1 dst 192.168.2.1 \
  encr_alg 3des auth_alg md5 \
  encrkey 294979e512cb8e79370dabec3fcfb849e78d2d6bd2049 \
  authkey 438c3ad2cecc9a3621e90462d11ca7d2f 
##
## for inbound packets
add esp spi 0x19174215 \
  src 192.168.2.1 dst 192.168.1.1 \
  encr_alg 3des auth_alg md5 \
  encrkey 9678f80e33406c86e3d1686e50406bd0434819c20d09d204 \
  authkey a61319630cf2abde7609ce24de3d029f
```

After verifying that the syntax of the ipseckeys file is valid, the administrator destroys the original key files.

```
# ipseckey -c /etc/inet/secret/ipseckeys  
# rm ipencrin1 ipencrout1 ipspi1 ipspi2 ipmds1 ipmds2
```
The administrator copies the ipseckey file to the communicating system by using the ssh command or another secure mechanism. On the communicating system, the protections are reversed. The first entry in the ipseckey file protects inbound packets, and the second entry protects outbound packets. No keys are generated on the communicating system.

How to Compute a Digest of a File

When you compute a digest of a file, you can check to see that the file has not been tampered with by comparing digest outputs. A digest does not alter the original file.

1. List the available digest algorithms.
   ```bash
   % digest -l
   md5
   sha1
   sha256
   sha384
   sha512
   ```

2. Compute the digest of the file and save the digest listing.
   Provide an algorithm with the digest command.
   ```bash
   % digest -v -a algorithm input-file > digest-listing
   -v: Displays the output in the following format:
   
   algorithm (input-file) = digest
   
   -a algorithm: Is the algorithm to use to compute a digest of the file. Type the algorithm as the algorithm appears in the output of Step 1.
   
   input-file: Is the input file for the digest command.
   
   digest-listing: Is the output file for the digest command.
   ```

Example 14–8  Computing a Digest With the MD5 Mechanism

In the following example, the digest command uses the MD5 mechanism to compute a digest for an email attachment.

```bash
% digest -v -a md5 email.attach >> $HOME/digest.emails.05.07
% cat ~/digest.emails.05.07
md5 (email.attach) = 85c0a53d1a5cc71ea34d9ee7b1b28b01
```

When the -v option is not used, the digest is saved with no accompanying information:
Computing a Digest With the SHA1 Mechanism

In the following example, the digest command uses the SHA1 mechanism to provide a directory listing. The results are placed in a file.

```
% digest -v -a sha1 docs/* > $HOME/digest.docs.legal.05.07
% more ~/digest.docs.legal.05.07
sha1 (docs/legal1) = 1df50e8ad219e34f0b911e097b7b588e31f9b435
sha1 (docs/legal2) = 68efa5a636291bde8f33e046eb33508c94842c38
sha1 (docs/legal3) = 085d991238d61bd08fa2946c183e8e32ccc6c9
sha1 (docs/legal4) = f3085eae7e2c8d008816564df28027d10e1d983
```

How to Compute a MAC of a File

A message authentication code, or MAC, computes a digest for the file and uses a secret key to further protect the digest. A MAC does not alter the original file.

1. **List the available mechanisms.**

   ```
   % mac -l
   Algorithm    Keysize:  Min  Max
   ---------------------
   des_mac      64   64
   sha1_hmac    8    512
   md5_hmac     8    512
   sha256_hmac  8    512
   sha384_hmac  8   1024
   sha512_hmac  8   1024
   ```

2. **Generate a symmetric key of the appropriate length.**

   You have two options. You can provide a passphrase from which a key will be generated. Or you can provide a key.

   - If you provide a passphrase, you must store or remember the passphrase. If you store the passphrase online, the passphrase file should be readable only by you.
   - If you provide a key, it must be the correct size for the mechanism. For the procedure, see “How to Generate a Symmetric Key by Using the dd Command” on page 250. You can also use the pktool command. For the procedure and some examples, see “How to Generate a Symmetric Key by Using the pktool Command” on page 252.
3 Create a MAC for a file.

Provide a key and use a symmetric key algorithm with the `mac` command.

```bash
% mac [-v] -a algorithm [-k keyfile | -K key-label [-T token]] input-file
```

- `-v` Displays the output in the following format:

  ```
  algorithm (input-file) = mac
  ```

- `-a algorithm` Is the algorithm to use to compute the MAC. Type the algorithm as the algorithm appears in the output of the `mac -l` command.

- `-k keyfile` Is the file that contains a key of algorithm-specified length.

- `-K key-label` Is the label of a key in the PKCS#11 keystore.

- `-T token` Is the token name. By default, the token is Sun Software PKCS#11 softtoken. Is used only when the `-K key-label` option is used.

- `input-file` Is the input file for the MAC.

### Example 14–10 Computing a MAC With `DES_MAC` and a Passphrase

In the following example, the email attachment is authenticated with the `DES_MAC` mechanism and a key that is derived from a passphrase. The MAC listing is saved to a file. If the passphrase is stored in a file, the file should not be readable by anyone but the user.

```bash
% mac -v -a des_mac email.attach
Enter passphrase: <Type passphrase>
der_mac (email.attach) = dd27870a
% echo "des_mac (email.attach) = dd27870a" >> ~/desmac.daily.05.07
```

### Example 14–11 Computing a MAC With `MD5_HMAC` and a Key File

In the following example, the email attachment is authenticated with the `MD5_HMAC` mechanism and a secret key. The MAC listing is saved to a file.

```bash
% mac -v -a md5_hmac -k $HOME/keyf/05.07.mack64 email.attach
md5_hmac (email.attach) = 02df6eb6c123ff25d78877eb1d55710c
% echo "md5_hmac (email.attach) = 02df6eb6c123ff25d78877eb1d55710c" \
>> ~/mac.daily.05.07
```

### Example 14–12 Computing a MAC With `SHA1_HMAC` and a Key File

In the following example, the directory manifest is authenticated with the `SHA1_HMAC` mechanism and a secret key. The results are placed in a file.
Computing a MAC With SHA1_HMAC and a Key Label

In the following example, the directory manifest is authenticated with the SHA1_HMAC mechanism and a secret key. The results are placed in the user’s PKCS #11 keystore. The user initially created the keystore and the password to the keystore by using the pktool setpin command.

```
% mac -a sha1_hmac -K legaldocs0507 docs/*
Enter pin for Sun Software PKCS#11 softtoken: Type password
```

To retrieve the MAC from the keystore, the user uses the verbose option, and provides the key label and the name of the directory that was authenticated.

```
% mac -v -a sha1_hmac -K legaldocs0507 docs/*
Enter pin for Sun Software PKCS#11 softtoken: Type password
```

How to Encrypt and Decrypt a File

When you encrypt a file, the original file is not removed or changed. The output file is encrypted.

For solutions to common errors from the encrypt command, see the section that follows the examples.

1. Create a symmetric key of the appropriate length.

You have two options. You can provide a passphrase from which a key will be generated. Or you can provide a key.

- If you provide a passphrase, you must store or remember the passphrase. If you store the passphrase online, the passphrase file should be readable only by you.
If you provide a key, it must be the correct size for the mechanism. For the procedure, see "How to Generate a Symmetric Key by Using the dd Command" on page 250. You can also use the pktool command. For the procedure and some examples, see “How to Generate a Symmetric Key by Using the pktool Command” on page 252.

2 Encrypt a file.

Provide a key and use a symmetric key algorithm with the encrypt command.

```
% encrypt -a algorithm [-v] \ 
[-k keyfile | -K key-label [-T token]] [-i input-file] [-o output-file]
```

- `-a algorithm` Is the algorithm to use to encrypt the file. Type the algorithm as the algorithm appears in the output of the encrypt -l command.
- `-k keyfile` Is the file that contains a key of algorithm-specified length. The key length for each algorithm is listed, in bits, in the output of the encrypt -l command.
- `-K key-label` Is the label of a key in the PKCS #11 keystore.
- `-T token` Is the token name. By default, the token is Sun Software PKCS#11 softtoken. Is used only when the `-K key-label` option is used.
- `-i input-file` Is the input file that you want to encrypt. This file is left unchanged by the command.
- `-o output-file` Is the output file that is the encrypted form of the input file.

### Example 14–14 Creating an AES Key for Encrypting Your Files

In the following example, a user creates and stores an AES key in an existing PKCS #11 keystore for use in encryption and decryption. The user can verify that the key exists and can use the key, but cannot view the key itself.

```
% pktool genkey label=MyAESkeynumber1 keytype=aes keylen=256
Enter PIN for Sun Software PKCS#11 softtoken : Type password

% pktool list objtype=key
Enter PIN for Sun Software PKCS#11 softtoken :<Type password>
Found 1 key
Key #1 - Sun Software PKCS#11 softtoken: MyAESkeynumber1 (256)

To use the key to encrypt a file, the user retrieves the key by its label.

% encrypt -a aes -K MyAESkeynumber1 -i encryptthisfile -o encryptedthisfile
To decrypt the encryptedthisfile file, the user retrieves the key by its label.

% decrypt -a aes -K MyAESkeynumber1 -i encryptedthisfile -o sameasencryptthisfile
```
Encrypting and Decrypting With AES and a Passphrase

In the following example, a file is encrypted with the AES algorithm. The key is generated from the passphrase. If the passphrase is stored in a file, the file should not be readable by anyone but the user.

```
% encrypt -a aes -i ticket.to.ride -o ~/enc/e.ticket.to.ride
Enter passphrase:     <Type passphrase>
Re-enter passphrase: Type passphrase again
```

The input file, ticket.to.ride, still exists in its original form.

To decrypt the output file, the user uses the same passphrase and encryption mechanism that encrypted the file.

```
% decrypt -a aes -i ~/enc/e.ticket.to.ride -o ~/d.ticket.to.ride
Enter passphrase:     <Type passphrase>
```

Encrypting and Decrypting With AES and a Key File

In the following example, a file is encrypted with the AES algorithm. AES mechanisms use a key of 128 bits, or 16 bytes.

```
% encrypt -a aes -k ~/keyf/05.07.aes16 \   
- i ticket.to.ride -o ~/enc/e.ticket.to.ride
```

The input file, ticket.to.ride, still exists in its original form.

To decrypt the output file, the user uses the same key and encryption mechanism that encrypted the file.

```
% decrypt -a aes -k ~/keyf/05.07.aes16 \   
- i ~/enc/e.ticket.to.ride -o ~/d.ticket.to.ride
```

Encrypting and Decrypting With ARCFOUR and a Key File

In the following example, a file is encrypted with the ARCFOUR algorithm. The ARCFOUR algorithm accepts a key of 8 bits (1 byte), 64 bits (8 bytes), or 128 bits (16 bytes).

```
% encrypt -a arcfour -i personal.txt \   
- k ~/keyf/05.07.rc4.8 -o ~/enc/e.personal.txt
```

To decrypt the output file, the user uses the same key and encryption mechanism that encrypted the file.
Encrypting and Decrypting With 3DES and a Key File

In the following example, a file is encrypted with the 3DES algorithm. The 3DES algorithm requires a key of 192 bits, or 24 bytes.

```
% encrypt -a 3des -k ~/keyf/05.07.des24 \
-i ~/personal2.txt -o ~/enc/e.personal2.txt
```

To decrypt the output file, the user uses the same key and encryption mechanism that encrypted the file.

```
% decrypt -a 3des -k ~/keyf/05.07.des24 \
-i ~/enc/e.personal2.txt -o ~/personal2.txt
```

Troubleshooting

The following messages indicate that the key that you provided to the encrypt command is not permitted by the algorithm that you are using.

- encrypt: unable to create key for crypto operation: CKR_ATTRIBUTE_VALUE_INVALID
- encrypt: failed to initialize crypto operation: CKR_KEY_SIZE_RANGE

If you pass a key that does not meet the requirements of the algorithm, you must supply a better key.

- One option is to use a passphrase. The framework then provides a key that meets the requirements.
- The second option is to pass a key size that the algorithm accepts. For example, the DES algorithm requires a key of 64 bits. The 3DES algorithm requires a key of 192 bits.

**Administering the Cryptographic Framework (Task Map)**

The following task map points to procedures for administering software and hardware providers in the Solaris Cryptographic Framework.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>List the providers in the Solaris</td>
<td>Lists the algorithms, libraries, and hardware devices that are available for use in the Solaris Cryptographic Framework.</td>
<td>“How to List Available Providers” on page 264</td>
</tr>
</tbody>
</table>

Chapter 14  •  Solaris Cryptographic Framework (Tasks) 263
Administering the Cryptographic Framework

This section describes how to administer the software providers and the hardware providers in the Solaris Cryptographic Framework. Software providers and hardware providers can be removed from use when desirable. For example, you can disable the implementation of an algorithm from one software provider. You can then force the system to use the algorithm from a different software provider.

### How to List Available Providers

The Solaris Cryptographic Framework provides algorithms for several types of consumers:

- User-level providers provide a PKCS #11 cryptographic interface to applications that are linked with the `libpkcs11` library
- Kernel software providers provide algorithms for IPSec, Kerberos, and other Solaris kernel components
- Kernel hardware providers provide algorithms that are available to kernel consumers and to applications through the `pkcs11_kernel` library

1. **List the providers in a brief format.**
Note – The contents and format of the providers list varies for different Solaris releases. Run the `cryptoadm list` command on your system to see the providers that your system supports.

Only those mechanisms at the user level are available for use by regular users.

```bash
% cryptoadm list
User-level providers:
Provider: /usr/lib/security/$ISA/pkcs11_kernel.so
Provider: /usr/lib/security/$ISA/pkcs11_softtoken.so

Kernel software providers:
  des
  aes
  arcfour
  blowfish
  ecc
  sha1
  sha2
  md4
  md5
  rsa
  swrand

Kernel hardware providers:
  ncp/0
```

2 List the providers and their mechanisms in the Solaris Cryptographic Framework.

All mechanisms are listed in the following output. However, some of the listed mechanisms might be unavailable for use. To list only the mechanisms that the administrator has approved for use, see Example 14–20.

The output is truncated for display purposes.

```bash
% cryptoadm list -m
User-level providers:
====================================================================
Provider: /usr/lib/security/$ISA/pkcs11_kernel.so
/usr/lib/security/$ISA/pkcs11_kernel.so: no slots presented.

Provider: /usr/lib/security/$ISA/pkcs11_softtoken.so
Mechanisms:
  CKM_DES_CBC
  CKM_DES_CBC_PAD
  CKM_DES_ECB
  CKM_DES_KEY_GEN
  CKM_DES_MAC_GENERAL
```
Kernel software providers:

des: CKM_DES_ECB, CKM_DES_CBC, CKM_DES3_ECB, CKM_DES3_CBC
aes: CKM_AES_ECB, CKM_AES_CBC, CKM_AES_CTR, CKM_AES_CCM, CKM_AES_GCM, CKM_AES_GMAC
arcfour: CKM_RC4
blowfish: CKM_BLOWFISH_ECB, CKM_BLOWFISH_CBC
ecc: CKM_EC_KEY_PAIR_GEN, CKM_ECDH1_DERIVE, CKM_ECDSA, CKM_ECDSA_SHA1
sha1: CKM_SHA1, CKM_SHA1_HMAC, CKM_SHA1_HMAC_GENERAL
sha2: CKM_SHA256, CKM_SHA256_HMAC, CKM_SHA256_HMAC_GENERAL, CKM_SHA384, CKM_SHA384_HMAC, CKM_SHA384_HMAC_GENERAL, CKM_SHA512, CKM_SHA512_HMAC, CKM_SHA512_HMAC_GENERAL
md4: CKM_MD4
md5: CKM_MD5, CKM_MD5_HMAC, CKM_MD5_HMAC_GENERAL
rsa: CKM_RSA_PKCS, CKM_RSA_X_509, CKM_MD5_RSA_PKCS, CKM_SHA1_RSA_PKCS, CKM_SHA256_RSA_PKCS, CKM_SHA384_RSA_PKCS, CKM_SHA512_RSA_PKCS
swrand: No mechanisms presented.

Kernel hardware providers:

ncp/0: CKM_DSA, CKM_RSA_X_509, CKM_RSA_PKCS, CKM_RSA_PKCS_KEY_PAIR_GEN, CKM_DH_PKCS_KEY_PAIR_GEN, CKM_DH_PKCS_DERIVE, CKM_EC_KEY_PAIR_GEN, CKM_ECDH1_DERIVE, CKM_ECDSA

Example 14–19 Finding the Existing Cryptographic Mechanisms

In the following example, all mechanisms that the user-level library, pkcs11_softtoken, offers are listed.

% cryptoadm list -m provider=/usr/lib/security/$ISA/pkcs11_softtoken.so
Mechanisms:
CKM_DES_CBC, CKM_DES_CBC_PAD, CKM_DES_ECB, CKM_DES_KEY_GEN,
CKM_DES3_CBC, CKM_DES3_CBC_PAD, CKM_DES3_ECB, CKM_DES3_KEY_GEN,
...
CKM_SSL3_KEY_AND_MAC_DERIVE, CKM_TLS_KEY_AND_MAC_DERIVE

Example 14–20 Finding the Available Cryptographic Mechanisms

Policy determines which mechanisms are available for use. The administrator sets the policy. An administrator can choose to disable mechanisms from a particular provider. The -p option displays the list of mechanisms that are permitted by the policy that the administrator has set.

% cryptoadm list -p
User-level providers:

waiting for input
How to Add a Software Provider

1. Assume the Primary Administrator role, or become superuser.

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in System Administration Guide: Basic Administration.

2. List the software providers that are available to the system.

   `% cryptoadm list`

   User-level providers:
   Provider: /usr/lib/security/$ISA/pkcs11_kernel.so
   Provider: /usr/lib/security/$ISA/pkcs11_softtoken.so

   Kernel software providers:
   des
   aes
   arcfour
   blowfish
   sha1
   sha2
   md4
   md5
   rsa
swrand

Kernel hardware providers:
  ncp/0

3  Add the provider's package by using the `pkgadd` command.

```bash
# pkgadd -d /path/to/package pkginst
```

The package must include software that has been signed by a certificate from Sun. To request a certificate from Sun and to sign a provider, see Appendix F, "Packaging and Signing Cryptographic Providers," in Solaris Security for Developers Guide.

The package should have scripts that notify the cryptographic framework that another provider with a set of mechanisms is available. For information about the packaging requirements, see Appendix F, "Packaging and Signing Cryptographic Providers," in Solaris Security for Developers Guide.

4  Refresh the providers.

You need to refresh providers if you added a software provider, or if you added hardware and specified policy for the hardware.

```bash
# svcadm refresh svc:/system/cryptosvc
```

5  Locate the new provider on the list.

In this case, a new kernel software provider was installed.

```bash
# cryptoadm list
...
Kernel software providers:
  des
  aes
  arcfour
  blowfish
  sha1
  sha2
  md4
  md5
  rsa
  swrand
  ecc <-- added provider
...
```

**Example 14–21  Adding a User-Level Software Provider**

In the following example, a signed PKCS #11 library is installed.

```bash
# pkgadd -d /cdrom/cdrom0/SolarisNew
Answer the prompts
```
svcadm refresh system/cryptosvc

# cryptoadm list

user-level providers:

/usr/lib/security/$ISA/pkcs11_kernel.so
/usr/lib/security/$ISA/pkcs11_softtoken.so
/opt/SUNWconn/lib/$ISA/libpkcs11.so.1  <-- added provider

Developers who are testing a library with the cryptographic framework can install the library manually.

# cryptoadm install provider=/opt/SUNWconn/lib/$ISA/libpkcs11.so.1

For information on getting your provider signed, see “Binary Signatures for Third-Party Software” on page 247.

▼ How to Prevent the Use of a User-Level Mechanism

If some of the cryptographic mechanisms from a library provider should not be used, you can remove selected mechanisms. This procedure uses the DES mechanisms in the pkcs11_softtoken library as an example.

1 Become superuser or assume a role that includes the Crypto Management rights profile.

To create a role that includes the Crypto Management rights profile and assign the role to a user, see Example 9–7.

2 List the mechanisms that are offered by a particular user-level software provider.

% cryptoadm list -m provider=/usr/lib/security/$ISA/pkcs11_softtoken.so

/usr/lib/security/$ISA/pkcs11_softtoken.so:

CKM_DES_CBC,CKM_DES_CBC_PAD,CKM_DES_ECB,CKM_DES_KEY_GEN,
CKM_DES3_CBC,CKM_DES3_CBC_PAD,CKM_DES3_ECB,CKM_DES3_KEY_GEN,
CKM_AES_CBC,CKM_AES_CBC_PAD,CKM_AES_ECB,CKM_AES_KEY_GEN,
...

3 List the mechanisms that are available for use.

$ cryptoadm list -p

user-level providers:

/usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled.
random is enabled.
...
4 Disable the mechanisms that should not be used.
   $ cryptoadm disable provider=/usr/lib/security/$ISA/pkcs11_softtoken.so \
   > mechanism=CKM_DES_CBC,CKM_DES_CBC_PAD,CKM_DES_ECB

5 List the mechanisms that are available for use.
   $ cryptoadm list -p provider=/usr/lib/security/$ISA/pkcs11_softtoken.so
   /usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled, 
   except CKM_DES_ECB,CKM_DES_CBC_PAD,CKM_DES_CBC. random is enabled.

Example 14–22 Enabling a User-Level Software Provider Mechanism

In the following example, a disabled DES mechanism is again made available for use.

   $ cryptoadm list -m provider=/usr/lib/security/$ISA/pkcs11_softtoken.so
   /usr/lib/security/$ISA/pkcs11_softtoken.so: 
   CKM_DES_CBC,CKM_DES_CBC_PAD,CKM_DES_ECB,CKM_DES_KEY_GEN, 
   CKM_DES3_CBC,CKM_DES3_CBC_PAD,CKM_DES3_ECB,CKM_DES3_KEY_GEN, 
   ...
   $ cryptoadm list -p provider=/usr/lib/security/$ISA/pkcs11_softtoken.so
   /usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled, 
   except CKM_DES_ECB,CKM_DES_CBC_PAD,CKM_DES_CBC. random is enabled.
   $ cryptoadm enable provider=/usr/lib/security/$ISA/pkcs11_softtoken.so \
   > mechanism=CKM_DES_ECB
   $ cryptoadm list -p provider=/usr/lib/security/$ISA/pkcs11_softtoken.so
   /usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled, 
   except CKM_DES_CBC_PAD,CKM_DES_CBC. random is enabled.

Example 14–23 Enabling All User-Level Software Provider Mechanisms

In the following example, all mechanisms from the user-level library are enabled.

   $ cryptoadm enable provider=/usr/lib/security/$ISA/pkcs11_softtoken.so all
   $ cryptoadm list -p provider=/usr/lib/security/$ISA/pkcs11_softtoken.so
   /usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled. 
   random is enabled.

Example 14–24 Permanently Removing User-Level Software Provider Availability

In the following example, the libpkcs11.so.1 library is removed.

   $ cryptoadm uninstall provider=/opt/SUNWconn/lib/$ISA/libpkcs11.so.1
   $ cryptoadm list
   user-level providers:
   /usr/lib/security/$ISA/pkcs11_kernel.so
   /usr/lib/security/$ISA/pkcs11_softtoken.so
How to Prevent the Use of a Kernel Software Provider

If the cryptographic framework provides multiple modes of a provider such as AES, you might remove a slow mechanism from use, or a corrupted mechanism. This procedure uses the AES algorithm as an example.

1. Become superuser or assume a role that includes the Crypto Management rights profile.
   To create a role that includes the Crypto Management rights profile and assign the role to a user, see Example 9–7.

2. List the mechanisms that are offered by a particular kernel software provider.
   ```bash
   cryptoadm list -m provider=aes
   aes: CKM_AES_ECB, CKM_AES_CBC
   ```

3. List the mechanisms that are available for use.
   ```bash
   cryptoadm list -p provider=aes
   aes: all mechanisms are enabled.
   ```

4. Disable the mechanism that should not be used.
   ```bash
   cryptoadm disable provider=aes mechanism=CKM_AES_ECB
   ```

5. List the mechanisms that are available for use.
   ```bash
   cryptoadm list -p provider=aes
   aes: all mechanisms are enabled, except CKM_AES_ECB.
   ```

Example 14–25 Enabling a Kernel Software Provider Mechanism

In the following example, a disabled AES mechanism is again made available for use.

```bash
cryptoadm list -m provider=aes
aes: CKM_AES_ECB, CKM_AES_CBC
$ cryptoadm list -p provider=aes
aes: all mechanisms are enabled, except CKM_AES_ECB.
$ cryptoadm enable provider=aes mechanism=CKM_AES_ECB
$ cryptoadm list -p provider=aes
aes: all mechanisms are enabled.
```
Example 14–26 Temporarily Removing Kernel Software Provider Availability

In the following example, the AES provider is temporarily removed from use. The unload subcommand is useful to prevent a provider from being loaded automatically while the provider is being uninstalled. For example, the unload subcommand would be used when installing a patch that affects the provider.

$ cryptoadm unload provider=aes

$ cryptoadm list

... Kernel software providers:
  des
  aes (inactive)
  arcfour
  blowfish
  ecc
  sha1
  sha2
  md4
  md5
  rsa
  swrand

The AES provider is unavailable until the cryptographic framework is refreshed.

$ svcadm refresh system/cryptosvc

$ cryptoadm list

... Kernel software providers:
  des
  aes
  arcfour
  blowfish
  ecc
  sha1
  sha2
  md4
  md5
  rsa
  swrand

If a kernel consumer is using the kernel software provider, the software is not unloaded. An error message is displayed and the provider continues to be available for use.
Example 14–27  Permanently Removing Software Provider Availability

In the following example, the AES provider is removed from use. Once removed, the AES provider does not appear in the policy listing of kernel software providers.

$ cryptoadm uninstall provider=aes

$ cryptoadm list
...
Kernel software providers:

des
arcfour
blowfish
ecc
sha1
sha2
md4
md5
rsa
swrand

If a kernel consumer is using the kernel software provider, an error message is displayed and the provider continues to be available for use.

Example 14–28  Reinstalling a Removed Kernel Software Provider

In the following example, the AES kernel software provider is reinstalled.

$ cryptoadm install provider=aes mechanism=CKM_AES_ECB,CKM_AES_CBC

$ cryptoadm list
...
Kernel software providers:

des
aes
arcfour
blowfish
ecc
sha1
sha2
md4
md5
rsa
swrand
How to List Hardware Providers

Hardware providers are automatically located and loaded. For more information, see `driver.conf(4)` man page.

Before You Begin

When you have hardware that expects to be used within the Solaris Cryptographic Framework, the hardware registers with the SPI in the kernel. The framework checks that the hardware driver is signed. Specifically, the framework checks that the object file of the driver is signed with a certificate that Sun issues.

For example, the Sun Crypto Accelerator 6000 board (`mca`), the `ncp` driver for the cryptographic accelerator on the UltraSPARC T1 and T2 processors (`ncp`), and the `n2cp` driver for the UltraSPARC T2 processors (`n2cp`) plug hardware mechanisms into the framework.

For information on getting your provider signed, see "Binary Signatures for Third-Party Software" on page 247.

1. List the hardware providers that are available on the system.
   ```
   % cryptoadm list
   ...
   kernel hardware providers:
   ncp@0
   ```

2. List the mechanisms that the chip or the board provides.
   ```
   % cryptoadm list -m provider=ncp@0
   ncp@0: CKM_DSA,CKM_RSA_X_509,CKM_RSA_PKCS,CKM_RSA_PKCS_KEY_PAIR_GEN,
   CKM_DH_PKCS_KEY_PAIR_GEN,CKM_DH_PKCS_DERIVE,CKM_EC_KEY_PAIR_GEN,
   CKM_ECDH1_DERIVE,CKM_ECDSA
   ```

3. List the mechanisms that are available for use on the chip or the board.
   ```
   % cryptoadm list -p provider=ncp@0
   ncp@0: all mechanisms are enabled.
   ```

How to Disable Hardware Provider Mechanisms and Features

You can selectively disable mechanisms and the random number feature from a hardware provider. To enable them again, see Example 14–29. The hardware in this example, the Sun Crypto Accelerator 1000 board, provides a random number generator.

1. Become superuser or assume a role that includes the Crypto Management rights profile.
   To create a role that includes the Crypto Management rights profile and assign the role to a user, see Example 9–7.
2 Choose the mechanisms or feature to disable.

List the hardware provider.

```
# cryptoadm list
...
Kernel hardware providers:
dca/0
```

- Disable selected mechanisms.

```
# cryptoadm list -m provider=dca/0
dca/0: CKM_RSA_PKCS, CKM_RSA_X_509, CKM_DSA, CKM DES_CBC, CKM DES3_CBC
random is enabled.
# cryptoadm disable provider=dca/0 mechanism=CKM DES_CBC,CKM DES3_CBC
# cryptoadm list -p provider=dca/0
   dca/0: all mechanisms are enabled except CKM DES_CBC,CKM DES3_CBC. random is enabled.
```

- Disable the random number generator.

```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled. random is enabled.
# cryptoadm disable provider=dca/0 random
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled. random is disabled.
```

- Disable all mechanisms. Do not disable the random number generator.

```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled. random is enabled.
# cryptoadm disable provider=dca/0 mechanism=all
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are disabled. random is enabled.
```

- Disable every feature and mechanism on the hardware.

```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled. random is enabled.
# cryptoadm disable provider=dca/0 all
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are disabled. random is disabled.
```

---

Example 14–29 Enabling Mechanisms and Features on a Hardware Provider

In the following examples, disabled mechanisms on a piece of hardware are selectively enabled.

```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled except CKM DES ECB,CKM DES3 ECB
```
random is enabled.
```
# cryptoadm enable provider=dca/0 mechanism=CKM_DES3_ECB
# cryptoadm list -p provider=dca/0
```
dca/0: all mechanisms are enabled except CKM_DES_ECB.
random is enabled.

In the following example, only the random generator is enabled.
```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled, except CKM_MD5,CKM_MD5_HMAC,....
random is disabled.
# cryptoadm enable provider=dca/0 random
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled, except CKM_MD5,CKM_MD5_HMAC,....
random is enabled.
```

In the following example, only the mechanisms are enabled. The random generator continues to be disabled.
```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled, except CKM_MD5,CKM_MD5_HMAC,....
random is disabled.
# cryptoadm enable provider=dca/0 mechanism=all
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled. random is disabled.
```

In the following example, every feature and mechanism on the board is enabled.
```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled, except CKM_DES_ECB,CKM_DES3_ECB.
random is disabled.
# cryptoadm enable provider=dca/0 all
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled. random is enabled.
```

▼ **How to Refresh or Restart All Cryptographic Services**

By default, the Solaris Cryptographic Framework is enabled. When the kcf daemon fails for any reason, the service management facility can be used to restart cryptographic services. For more information, see the `smf(5)` and `svcadm(1M)` man pages. For the effect on zones of restarting cryptographic services, see "Cryptographic Services and Zones" on page 248.

1. **Check the status of cryptographic services.**

   ```
   % svcs cryptosvc
   STATE STIME FMRI
   offline Dec_09 svc:/system/cryptosvc:default
   ```
2  Become superuser or assume an equivalent role to enable cryptographic services.
    Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" on page 174.

    # svcadm enable svc:/system/cryptosvc

**Example 14–30**  Refreshing Cryptographic Services

    In the following example, cryptographic services are refreshed in the global zone. Therefore, kernel-level cryptographic policy in every non-global zone is also refreshed.

    # svcadm refresh system/cryptosvc
In the Solaris Express Community Edition, the Key Management Framework (KMF) provides tools and programming interfaces for managing public key objects. Public key objects include X.509 certificates and public/private key pairs. The formats for storing these objects can vary. KMF also provides a tool for managing policies that define the use of X.509 certificates by applications. In the Solaris Express Community Edition, KMF supports third-party plugins.

- “Managing Public Key Technologies” on page 279
- “Key Management Framework Utilities” on page 280
- “Using the Key Management Framework (Tasks)” on page 282

Managing Public Key Technologies

The Key Management Framework (KMF) provides a unified approach to managing public key technologies (PKI). The Solaris OS has several different applications that make use of PKI technologies. Each application provides its own programming interfaces, key storage mechanisms, and administrative utilities. If an application provides a policy enforcement mechanism, the mechanism applies to that application only. With KMF, applications use a unified set of administrative tools, a single set of programming interfaces, and a single policy enforcement mechanism. These features manage the PKI needs of all applications that adopt these interfaces.

KMF unifies the management of public key technologies with the following interfaces:

- **pktool command** – This command manages PKI objects, such as certificates, in a variety of keystores.

- **kmcfog command** – This command manages the PKI policy database and third-party plugins.

  PKI policy decisions include operations such as the validation method for an operation. Also, PKI policy can limit the scope of a certificate. For example, PKI policy might assert that a certificate can be used only for specific purposes. Such a policy would prevent that certificate from being used for other requests.
Key Management Framework Utilities

KMF library – This library contains programming interfaces that abstract the underlying key store mechanism. Applications do not have to choose one particular key store mechanism, but can migrate from one mechanism to another mechanism. The supported keystores are PKCS #11, NSS, and OpenSSL. The library includes a pluggable framework so that new key store mechanisms can be added. Therefore, applications that use the new mechanisms would require only minor modifications to use a new key store.

Key Management Framework Utilities

KMF provides methods for managing the storage of keys and provides the overall policy for the use of those keys. KMF manages the policy, keys, and certificates for three public key technologies:

- Tokens from PKCS #11 providers, that is, from the Solaris Cryptographic Framework
- NSS, that is, Network Security Services
- OpenSSL, a file-based key store

The kmfcfg tool can create, modify, or delete KMF policy entries. The tool also manages plugins to the framework. KMF manages keystores through the pktool command. For more information, see the kmfcfg(1) and pktool(1) man pages, and the following sections.

KMF Policy Management

KMF policy is stored in a database. This policy database is accessed internally by all applications that use the KMF programming interfaces. The database can constrain the use of the keys and certificates that are managed by the KMF library. When an application attempts to verify a certificate, the application checks the policy database. The kmfcfg command modifies the policy database.

KMF Plugin Management

The kmfcfg command provides the following subcommands for plugins:

- list plugin – Lists plugins that are managed by KMF.
- install plugin – Installs the plugin by the module's path name and creates a key store for the plugin. To remove the plugin from KMF, you remove the key store.
- uninstall plugin – Removes the plugin from KMF by removing its key store.
- modify plugin – Enables the plugin to be run with an option that is defined in the code for the plugin, such as debug.
For more information, see the `kmf.cfg`(1) man page. For the procedure, see “How to Manage Third-Party Plugins in KMF” on page 287.

**KMF Keystore Management**

KMF manages the keystores for three public key technologies, PKCS #11 tokens, NSS, and OpenSSL. For all of these technologies, the `pktool` command enables you to do the following:

- Designate a keystore for each certificate and PIN.
- Generate a self-signed certificate.
- Generate a certificate request.
- Generate a symmetric key.
- Import objects into the keystore.
- List the objects in the keystore.
- Delete objects from the keystore.
- Download a CRL.

For the PKCS #11 and NSS technologies, the `pktool` command also enables you to set a PIN by generating a passphrase:

- Generate a passphrase for the keystore.
- Generate a passphrase for an object in the keystore.

For examples of using the `pktool` utility, see the `pktool`(1) man page and “Using the Key Management Framework (Task Map)” on page 281.

**Using the Key Management Framework (Task Map)**

The Key Management Framework (KMF) enables you to centrally manage public key technologies.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a certificate.</td>
<td>Creates a certificate for use by PKCS #11, NSS, or SSL.</td>
<td>&quot;How to Create a Certificate by Using the pktool gencert Command“ on page 282</td>
</tr>
<tr>
<td>Export a certificate.</td>
<td>Creates a file with the certificate and its supporting keys. The file can be protected with a password.</td>
<td>&quot;How to Export a Certificate and Private Key in PKCS #12 Format“ on page 285</td>
</tr>
<tr>
<td>Import a certificate.</td>
<td>Imports a certificate from another system.</td>
<td>&quot;How to Import a Certificate Into Your Keystore“ on page 283</td>
</tr>
<tr>
<td></td>
<td>Imports a certificate in PKCS #12 format from another system.</td>
<td>Example 15–2</td>
</tr>
</tbody>
</table>
Using the Key Management Framework (Tasks)

This section describes how to use the `pktool` command to manage your public key objects, such as passwords, passphrases, files, keystores, certificates, and CRLs.

### How to Create a Certificate by Using the `pktool gencert` Command

This procedure creates a self-signed certificate and stores the certificate in the PKCS #11 keystore. As a part of this operation, an RSA public/private key pair is also created. The private key is stored in the keystore with the certificate.

1. **Generate a self-signed certificate.**
   
   ```
   % pktool gencert [keystore=keystore] \label=label-name \subject=subject-DN serial=hex-serial-number
   ```

   - `keystore=keystore` Specifies the keystore by type of public key object. The value can be `nss`, `pkcs11`, or `ssl`. This keyword is optional.
   - `\label=label-name` Is a unique name that the issuer gives to the certificate.
   - `\subject=subject-DN` Is the distinguished name for the certificate.
   - `serial=hex-serial-number` Is the serial number in hexadecimal format. The issuer of the certificate chooses the number, such as `0x0102030405`.

2. **Verify the contents of the keystore.**
   
   ```
   % pktool list
   Found number certificates.
   1. (X.509 certificate)
      Label: label-name
      ID: Fingerprint that binds certificate to private key
   ```
This command lists all certificates in the keystore. In the following example, the keystore contains one certificate only.

Example 15–1 Creating a Self-Signed Certificate by Using pktool

In the following example, a user at My Company creates a self-signed certificate and stores the certificate in a keystore for PKCS #11 objects. The keystore is initially empty. If the keystore has not been initialized, the PIN for the softtoken is changeme.

% pktool gencert keystore=pkcs11 label="My Cert" \
subject="C=US, O=My Company, OU=Security Engineering Group, CN=MyCA" \
serial=0x000000001
Enter pin for Sun Software PKCS#11 softtoken: Type PIN for token

% pktool list
Found 1 certificates.
1. (X.509 certificate)
   Label: My Cert
   Subject: C=US, O=My Company, OU=Security Engineering Group, CN=MyCA
   Issuer: C=US, O=My Company, OU=Security Engineering Group, CN=MyCA
   Serial: 0x01

▼ How to Import a Certificate Into Your Keystore

This procedure describes how to import a file with PKI information that is encoded with PEM or with raw DER into your keystore. For an export procedure, see Example 15–4.

1 Import the certificate.
   % pktool import keystore=keystore infile=infile-name label=label-name

2 If you are importing private PKI objects, provide passwords when prompted.
   a. At the prompt, provide the password for the file.
      If you are importing PKI information that is private, such as an export file in PKCS #12 format, the file requires a password. The creator of the file that you are importing provides you with the PKCS #12 password.
      Enter password to use for accessing the PKCS12 file: Type PKCS #12 password
b. At the prompt, type the password for your keystore.
   Enter pin for Sun Software PKCS#11 softtoken: Type PIN for token

3 Verify the contents of the keystore.

% pktool list
Found number certificates.
1. (X.509 certificate)
   Label: label-name
   ID: Fingerprint that binds certificate to private key
   Subject: subject-DN
   Issuer: distinguished-name
   Serial: hex-serial-number
2. ...

Example 15–2 Importing a PKCS #12 File Into Your Keystore

In the following example, the user imports a PKCS #12 file from a third party. The pktool import command extracts the private key and the certificate from the gracedata.p12 file, and stores them in the user's preferred keystore.

% pktool import keystore=pkcs11 infile=gracedata.p12 label=GraceCert
Enter password to use for accessing the PKCS12 file: Type PKCS#12 password
Enter pin for Sun Software PKCS#11 softtoken: Type PIN for token
Found 1 certificate(s) and 1 key(s) in gracedata.p12
% pktool list
Found 1 certificates.
1. (X.509 certificate)
   Label: GraceCert
   Subject: C=US, O=My Company, OU=Security Engineering Group, CN=MyCA
   Issuer: C=US, O=My Company, OU=Security Engineering Group, CN=MyCA
   Serial: 0x01

Example 15–3 Importing an X.509 Certificate Into Your Keystore

In the following example, the user imports an X.509 certificate in PEM format into the user's preferred keystore. This public certificate is not protected with a password. The user's public keystore is also not protected by a password.

% pktool import keystore=pkcs11 infile=somecert.pem label="TheirCompany Root Cert"
% pktool list
Found 1 certificates.
1. (X.509 certificate)
   Label: TheirCompany Root Cert
How to Export a Certificate and Private Key in PKCS #12 Format

You can create a file in PKCS #12 format to export private keys and their associated X.509 certificate to other systems. Access to the file is protected by a password.

1 Find the certificate to export.
% pktool list
Found number certificates.
1. (X.509 certificate)
   Label: label-name
   ID: Fingerprint that binds certificate to private key
   Subject: subject-DN
   Issuer: distinguished-name
   Serial: hex-serial-number
2. ...

2 Export the keys and certificate.
Use the keystore and label from the pktool list command. Provide a file name for the export file. When the name contains a space, surround the name with double quotes.
% pktool export keystore=keystore outfile=outfile-name label=label-name

3 Protect the export file with a password.
At the prompt, type the current password for the keystore. At this point, you create a password for the export file. The receiver must provide this password when importing the file.
Enter pin for Sun Software PKCS#11 softtoken: Type PIN for token
Enter password to use for accessing the PKCS12 file: Create PKCS #12 password

Tip – Send the password separately from the export file. Best practice suggests that you provide the password out of band, such as during a telephone call.

Example 15–4 Exporting a Certificate and Private Key in PKCS #12 Format

In the following example, a user exports the private keys with their associated X.509 certificate into a standard PKCS #12 file. This file can be imported into other keystores. The PKCS #11 password protects the source keystore. The PKCS #12 password is used to protect private data in the PKCS #12 file. This password is required to import the file.
% pktool list
Found 1 certificates.
1. (X.509 certificate)
   Label: My Cert
   Subject: C=US, O=My Company, OU=Security Engineering Group, CN=MyCA
   Issuer: C=US, O=My Company, OU=Security Engineering Group, CN=MyCA
   Serial: 0x01

% pktool export keystore=pkcs11 outfile=mydata.p12 label="My Cert"
Enter pin for Sun Software PKCS#11 softtoken: Type PIN for token
Enter password to use for accessing the PKCS12 file: Create PKCS #12 password

The user then telephones the recipient and provides the PKCS #12 password.

▼ How to Generate a Passphrase by Using the pktool setpin Command

You can generate a passphrase for an object in a keystore, and for the keystore itself. The passphrase is required to access the object or keystore. For an example of generating a passphrase for an object in a keystore, see Example 15–4.

1  Generate a passphrase for access to a keystore.
   % pktool setpin keystore=nss|pkcs11 dir=directory

2  Answer the prompts.
   If the keystore does not have a password already set, press the Return key to create the password.
   Enter current token passphrase: Press the Return key
   Create new passphrase: Type the passphrase that you want to use
   Re-enter new passphrase: Retype the passphrase
   Passphrase changed.

   The keystore is now protected by passphrase. If you lose the passphrase, you lose access to the objects in the keystore.

Example 15–5  Protecting a Keystore With a Passphrase

The following example shows how to set the passphrase for an NSS database. Because no passphrase has been created, the user presses the Return key at the first prompt.

% pktool setpin keystore=nss dir=/var/nss
Enter current token passphrase: Press the Return key
Create new passphrase: has8n0Ndah
Re-enter new passphrase: has8n0Ndah
Passphrase changed.

▼ How to Manage Third-Party Plugins in KMF

You identify your plugin by giving it a keystore name. When you add the plugin to KMF, the software identifies it by its keystore name. The plugin can be defined to accept an option. This procedure includes how to remove the plugin from KMF.

1 Install the plugin.

   % /usr/bin/kmfcfg install keystore=keystore-name \
      modulepath=path-to-plugin [option="option-string"]

   where

   keystore-name – Is a unique name for the keystore that you provide.

   path-to-plugin – Is the full path to the shared library object for the KMF plugin.

   option-string – Is an optional argument to the shared library object.

2 List the plugins.

   % kmfcfg list plugin
   keystore-name:path-to-plugin [(built-in)] | [;option=option-string]

3 To remove the plugin, uninstall it and verify its removal.

   % kmfcfg uninstall keystore=keystore-name
   % kmfcfg plugin list

Example 15–6 Calling a KMF Plugin With an Option

In the following example, the administrator stores a KMF plugin in a site-specific directory. The plugin is defined to accept a debug option. The administrator adds the plugin and verifies that the plugin is installed.

   # /usr/bin/kmfcfg install keystore=mykmfplug \
      modulepath=/lib/security/site-modules/mykmfplug.so
   # kmfcfg list plugin
   KMF plugin information:
   ---------------------------
   pkcs11:kmf_pkcs11.so.1 (built-in)
   file:kmf_openssl.so.1 (built-in)
   nss:kmf_nss.so.1 (built-in)
   mykmfplug:/lib/security/site-modules/mykmfplug.so
Using the Key Management Framework (Tasks)

```
# kmfcfg modify plugin keystore=mykmfplug option="debug"
# kmfcfg list plugin
KMF plugin information:
-----------------------
...                      
mykmfplug:/lib/security/site-modules/mykmfplug.so;option=debug

The plugin now runs in debugging mode.
```
This section discusses authentication services that can be configured on a non-networked system, or between two systems. To configure a network of authenticated users and systems, see Part VI, "Kerberos Service."
This chapter provides information about how to use Secure RPC to authenticate a host and a user across an NFS mount. The following is a list of the topics in this chapter.

- “Overview of Secure RPC” on page 291
- “Administering Secure RPC (Task Map)” on page 296

### Overview of Secure RPC

Secure RPC (Remote Procedure Call) protects remote procedures with an authentication mechanism. The Diffie-Hellman authentication mechanism authenticates both the host and the user who is making a request for a service. The authentication mechanism uses Data Encryption Standard (DES) encryption. Applications that use Secure RPC include NFS and the name services, NIS and NIS+.

### NFS Services and Secure RPC

NFS enables several hosts to share files over the network. Under the NFS service, a server holds the data and resources for several clients. The clients have access to the file systems that the server shares with the clients. Users who are logged in to the client systems can access the file systems by mounting the file systems from the server. To the user on the client system, it appears as if the files are local to the client. One of the most common uses of NFS allows systems to be installed in offices, while storing all user files in a central location. Some features of the NFS service, such as the -nosuid option to the mount command, can be used to prohibit the opening of devices and file systems by unauthorized users.
The NFS service uses Secure RPC to authenticate users who make requests over the network. This process is known as Secure NFS. The Diffie-Hellman authentication mechanism, AUTH_DH, uses DES encryption to ensure authorized access. The AUTH_DH mechanism has also been called AUTH_DES. For more information, see the following:

- To set up and administer Secure NFS, see “Administering the Secure NFS System” in *System Administration Guide: Network Services*.
- To set up the NIS+ tables and enter names in the *cred* table, see *System Administration Guide: Naming and Directory Services (NIS+)*.
- For an outline of the transactions that are involved in RPC authentication, see “Implementation of Diffie-Hellman Authentication” on page 293.

**DES Encryption With Secure NFS**

The Data Encryption Standard (DES) encryption functions use a 56-bit key to encrypt data. If two credential users or principals know the same DES key, they can communicate in private by using the key to encipher and decipher text. DES is a relatively fast encryption mechanism. A DES chip makes the encryption even faster. However, if the chip is not present, a software implementation is substituted.

The risk of using just the DES key is that an intruder can collect enough cipher-text messages that were encrypted with the same key to be able to discover the key and decipher the messages. For this reason, security systems such as Secure NFS need to change the keys frequently.

**Kerberos Authentication**

Kerberos is an authentication system that was developed at MIT. Some encryption in Kerberos is based on DES. Kerberos V4 support is no longer supplied as part of Secure RPC. However, a client-side and server-side implementation of Kerberos V5, which uses RPCSEC_GSS, is included with this release. For more information, see Chapter 21, “Introduction to the Kerberos Service.”

**Diffie-Hellman Authentication and Secure RPC**

The Diffie-Hellman (DH) method of authenticating a user is nontrivial for an intruder to crack. The client and the server have their own private key, which they use with the public key to devise a common key. The private key is also known as the *secret key*. The client and the server use the common key to communicate with each other. The common key is encrypted with an agreed-upon encryption function, such as DES.
Authentication is based on the ability of the sending system to use the common key to encrypt the current time. Then, the receiving system can decrypt and check against its current time. The time on the client and the server must be synchronized. For more information, see “Managing Network Time Protocol (Tasks)” in System Administration Guide: Network Services.

The public keys and private keys are stored in an NIS or NIS+ database. NIS stores the keys in the publickey map. NIS+ stores the keys in the cred table. These files contain the public key and the private key for all potential users.

The system administrator is responsible for setting up NIS maps or NIS+ tables, and for generating a public key and a private key for each user. The private key is stored in encrypted form with the user’s password. This process makes the private key known only to the user.

Implementation of Diffie-Hellman Authentication

This section describes the series of transactions in a client-server session that use Diffie-Hellman authentication (AUTH_DH).

Generating the Public Keys and Secret Keys for Secure RPC

Sometime prior to a transaction, the administrator runs either the newkey or the nisaddcred command to generate a public key and a secret key. Each user has a unique public key and secret key. The public key is stored in a public database. The secret key is stored in encrypted form in the same database. The chkey command changes the key pair.

Running the keylogin Command for Secure RPC

Normally, the login password is identical to the Secure RPC password. In this case, the keylogin command is not required. However, if the passwords are different, the users have to log in and then run the keylogin command.

The keylogin command prompts the user for a Secure RPC password. The command then uses the password to decrypt the secret key. The keylogin command then passes the decrypted secret key to the keyserver program. The keyserver is an RPC service with a local instance on every computer. The keyserver saves the decrypted secret key and waits for the user to initiate a Secure RPC transaction with a server.

If both the login password and the RPC password are the same, the login process passes the secret key to the keyserver. If the passwords are required to be different, then the user must always run the keylogin command. When the keylogin command is included in the user’s environment configuration file, such as the ~/.login, ~/.cshrc, or ~/.profile file, the keylogin command runs automatically whenever the user logs in.
Generating the Conversation Key for Secure RPC

When the user initiates a transaction with a server, the following occurs:

1. The keyserver randomly generates a conversation key.
2. The kernel uses the conversation key, plus other material, to encrypt the client’s timestamp.
3. The keyserver looks up the server’s public key in the public key database. For more information, see the `publickey(4)` man page.
4. The keyserver uses the client’s secret key and the server’s public key to create a common key.
5. The keyserver encrypts the conversation key with the common key.

Initially Contacting the Server in Secure RPC

The transmission, which includes the encrypted timestamp and the encrypted conversation key, is then sent to the server. The transmission includes a credential and a verifier. The credential contains three components:

- The client’s network name
- The conversation key, which is encrypted with the common key
- A “window,” which is encrypted with the conversation key

The window is the difference in time that the client says should be allowed between the server’s clock and the client’s timestamp. If the difference between the server’s clock and the timestamp is greater than the window, the server rejects the client’s request. Under normal circumstances, this rejection does not happen, because the client first synchronizes with the server before starting the RPC session.

The client’s verifier contains the following:

- The encrypted timestamp
- An encrypted verifier of the specified window, which is decremented by 1

The window verifier is needed in case somebody wants to impersonate a user. The impersonator can write a program that, instead of filling in the encrypted fields of the credential and verifier, just inserts random bits. The server decrypts the conversation key into some random key. The server then uses the key to try to decrypt the window and the timestamp. The result is random numbers. After a few thousand trials, however, the random window/timestamp pair is likely to pass the authentication system. The window verifier lessens the chance that a fake credential could be authenticated.
Decrypting the Conversation Key in Secure RPC

When the server receives the transmission from the client, the following occurs:

1. The keyserver that is local to the server looks up the client’s public key in the public key database.
2. The keyserver uses the client’s public key and the server’s secret key to deduce the common key. The common key is the same common key that is computed by the client. Only the server and the client can calculate the common key because the calculation requires knowing one of the secret keys.
3. The kernel uses the common key to decrypt the conversation key.
4. The kernel calls the keyserver to decrypt the client’s timestamp with the decrypted conversation key.

Storing Information on the Server in Secure RPC

After the server decrypts the client’s timestamp, the server stores four items of information in a credential table:

- The client’s computer name
- The conversation key
- The window
- The client’s timestamp

The server stores the first three items for future use. The server stores the client’s timestamp to protect against replays. The server accepts only timestamps that are chronologically greater than the last timestamp seen. As a result, any replayed transactions are guaranteed to be rejected.

Note – Implicit in these transactions is the name of the caller, who must be authenticated in some manner. The keyserver cannot use DES authentication to authenticate the caller because the use of DES by the keyserver would create a deadlock. To avoid a deadlock, the keyserver stores the secret keys by user ID (UID) and grants requests only to local root processes.

Returning the Verifier to the Client in Secure RPC

The server returns a verifier to the client, which includes the following:

- The index ID, which the server records in its credential cache
- The client’s timestamp minus 1, which is encrypted by the conversation key

The reason for subtracting 1 from the client’s timestamp is to ensure that the timestamp is out of date. An out-of-date timestamp cannot be reused as a client verifier.
Authenticating the Server in Secure RPC

The client receives the verifier and authenticates the server. The client knows that only the server could have sent the verifier because only the server knows what timestamp the client sent.

Handling Transactions in Secure RPC

With every transaction after the first transaction, the client returns the index ID to the server in its next transaction. The client also sends another encrypted timestamp. The server sends back the client’s timestamp minus 1, which is encrypted by the conversation key.

Administering Secure RPC (Task Map)

The following task map points to procedures that configure Secure RPC for NIS, NIS+, and NFS.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Start the keyserver.</td>
<td>Ensures that keys can be created so that users can be authenticated.</td>
</tr>
<tr>
<td>2.</td>
<td>Set up credentials on an NIS+ host.</td>
<td>Ensures that the root user on a host can be authenticated in an NIS+ environment.</td>
</tr>
<tr>
<td>3.</td>
<td>Give an NIS+ user a key.</td>
<td>Enables a user to be authenticated in an NIS+ environment.</td>
</tr>
<tr>
<td>4.</td>
<td>Set up credentials on an NIS host.</td>
<td>Ensures that the root user on a host can be authenticated in an NIS environment.</td>
</tr>
<tr>
<td>5.</td>
<td>Give an NIS user a key.</td>
<td>Enables a user to be authenticated in an NIS environment.</td>
</tr>
</tbody>
</table>

Administering Authentication With Secure RPC

By requiring authentication for use of mounted NFS file systems, you increase the security of your network.
How to Restart the Secure RPC Keyserver

1. **Assume the Primary Administrator role, or become superuser.**
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2. **Verify that the keyserv daemon is running.**
   ```
   # svc \*keyserv\*
   STATE STIME FMRI
   disabled Dec_14 svc:/network/rpc/keyserv
   ```

3. **Enable the keyserver service if the service is not online.**
   ```
   # svcadm enable network/rpc/keyserv
   ```

How to Set Up a Diffie-Hellman Key for an NIS+ Host

This procedure should be done on every host in the NIS+ domain. After root has run the keylogin command, the server has GSS-API acceptor credentials for mech_dh and the client has GSS-API initiator credentials.

For a detailed description of NIS+ security, see System Administration Guide: Naming and Directory Services (NIS+).

1. **Assume the Primary Administrator role, or become superuser.**
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2. **Enable the publickey table in the name service.**
   Add the following line to the /etc/nsswitch.conf file:
   ```
   publickey: nisplus
   ```

3. **Initialize the NIS+ client.**
   ```
   # nisinit -cH hostname
   ```
   where `hostname` is the name of a trusted NIS+ server that contains an entry in its tables for the client system.
4  **Add the client to the cred table.**  
   Type the following commands:
   
   ```
   # nisaddcred local
   # nisaddcred des
   ```

5  **Verify the setup by using the keylogin command.**  
   If you are prompted for a password, the procedure has succeeded.
   
   ```
   # keylogin
   Password:
   ```

**Example 16–1  Setting Up a New Key for root on an NIS+ Client**

The following example uses the host pluto to set up earth as an NIS+ client. You can ignore the warnings. The keylogin command is accepted, verifying that earth is correctly set up as a secure NIS+ client.

```
# nisinit -cH pluto
NIS Server/Client setup utility.
This system is in the example.com. directory.
Setting up NIS+ client ...  
All done.
# nisaddcred local
# nisaddcred des
DES principal name : unix.earth@example.com
Adding new key for unix.earth@example.com (earth.example.com.)
Network password: <Type password>
Warning, password differs from login password.
Retype password: <Retype password>
# keylogin
Password: <Type password>
#
```

**How to Set Up a Diffie-Hellman Key for an NIS+ User**

This procedure should be done on every user in the NIS+ domain.

1  **Assume the Primary Administrator role, or become superuser.**  
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2  **Add the user to the cred table on the root master server.**  
   Type the following command:
   
   ```
   # nisaddcred -p unix.UID@domain-name -P username.domain-name.des
   ```
Note that, in this case, the `username.domain-name` must end with a dot (.).

3 Verify the setup by logging in as the client and typing the `keylogin` command.

### Example 16–2 Setting Up a New Key for an NIS+ User

In the following example, a key for Diffie-Hellman authentication is given to the user `jdoe`.

```
# nisaddcred -p unix.1234@example.com -P jdoe.example.com. des
DES principal name : unix.1234@example.com
Adding new key for unix.1234@example.com (jdoe.example.com.)
Password: <Type password>
Retype password: <Retype password>
# rlogin rootmaster -l jdoe
% keylogin
Password: <Type password>
%
```

### How to Set Up a Diffie-Hellman Key for an NIS Host

This procedure should be done on every host in the NIS domain.

1 **Assume the Primary Administrator role, or become superuser.**
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in System Administration Guide: Basic Administration.

2 **Enable the `publickey` map in the name service.**
   Add the following line to the `/etc/nsswitch.conf` file:
   ```
   publickey: nis
   ```

3 **Create a new key pair by using the `newkey` command.**
   ```
   # newkey -h hostname
   ```
   where `hostname` is the name of the client.

### Example 16–3 Setting Up a New Key for root on an NIS Client

In the following example, `earth` is set up as a secure NIS client.

```
# newkey -h earth
Adding new key for unix.earth@example.com
New Password: <Type password>
Retype password: <Retype password>
```
How to Set Up a Diffie-Hellman Key for an NIS User

This procedure should be done for every user in the NIS domain.

Before You Begin

- Only system administrators, when logged in to the NIS master server, can generate a new key for a user.

1. **Assume the Primary Administrator role, or become superuser.**
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, ”Working With the Solaris Management Console (Tasks),” in *System Administration Guide: Basic Administration.*

2. **Create a new key for a user.**
   
   ```
   # newkey -u username
   ```
   where *username* is the name of the user. The system prompts for a password. You can type a generic password. The private key is stored in an encrypted form by using the generic password.

3. **Tell the user to log in and type the chkey -p command.**
   This command allows users to re-encrypt their private keys with a password known only to the user.

   Note – The chkey command can be used to create a new key pair for a user.

---

Example 16–4: Setting Up and Encrypting a New User Key in NIS

In this example, superuser sets up the key.

```bash
# newkey -u jdoe
Adding new key for unix.12345@example.com
New Password: <Type password>
Retype password: <Retype password>
Please wait for the database to get updated...
Your new key has been successfully stored away.
#
```

Then the *jdoe* re-encrypts the key with a private password.
% chkey -p
Updating nis publickey database.
Reencrypting key for unix.12345@example.com
Please enter the Secure-RPC password for jdoe: <Type password>
Please enter the login password for jdoe: <Type password>
Sending key change request to central.example...

How to Share NFS Files With Diffie-Hellman Authentication

This procedure protects shared file systems on an NFS server by requiring authentication for access.

Before You Begin
Diffie-Hellman public key authentication must be enabled on the network. To enable authentication on the network, do one of the following:

- “How to Set Up a Diffie-Hellman Key for an NIS+ Host” on page 297
- “How to Set Up a Diffie-Hellman Key for an NIS Host” on page 299

1 Become superuser or assume a role that includes the System Management profile.
The System Administrator role includes the System Management profile. To create the role and assign the role to a user, see “Configuring RBAC (Task Map)” on page 174.

2 On the NFS server, share a file system with Diffie-Hellman authentication.

```
# share -F nfs -o sec=dh /filesystem
```
where `filesystem` is the file system that is being shared.
The `-o sec=dh` option means that AUTH_DH authentication is now required to access the file system.

3 On an NFS client, mount a file system with Diffie-Hellman authentication.

```
# mount -F nfs -o sec=dh server:filesystem mount-point
```
where `server` is the name of the system that is sharing `filesystem`
`filesystem` is the name of the file system that is being shared, such as `opt`
`mount-point` is the name of the mount point, such as `/opt`
The `-o sec=dh` option mounts the file system with AUTH_DH authentication.
This chapter covers the Pluggable Authentication Module (PAM) framework. PAM provides a method to “plug in” authentication services into the Solaris Operating System (Solaris OS). PAM provides support for multiple authentication services when accessing a system.

- “PAM (Overview)” on page 303
- “PAM (Tasks)” on page 306
- “PAM Configuration (Reference)” on page 309

PAM (Overview)

The Pluggable Authentication Module (PAM) framework lets you “plug in” new authentication services without changing system entry services, such as login, ftp, and telnet. You can also use PAM to integrate UNIX login with other security mechanisms such as Kerberos. Mechanisms for account, credential, session, and password management can also be “plugged in” by using this framework.

Benefits of Using PAM

The PAM framework enables you to configure the use of system entry services (such as, ftp, login, telnet, or rsh) for user authentication. Some benefits that PAM provides are as follows:

- Flexible configuration policy
  - Per-application authentication policy
  - The ability to choose a default authentication mechanism
  - The ability to require multiple authorizations on high-security systems
- Ease of use for the end user
  - No retying of passwords if the passwords are the same for different authentication services
The ability to prompt the user for passwords for multiple authentication services without requiring the user to type multiple commands

- The ability to pass optional options to the user authentication services

- The ability to implement a site-specific security policy without having to change the system entry services

**Introduction to the PAM Framework**

The PAM framework consists of four parts:

- PAM consumers
- PAM library
- The `pam.conf(4)` configuration file
- PAM service modules, also referred to as providers

The framework provides a uniform way for authentication-related activities to take place. This approach enables application developers to use PAM services without having to know the semantics of the policy. Algorithms are centrally supplied. The algorithms can be modified independently of the individual applications. With PAM, administrators can tailor the authentication process to the needs of a particular system without having to change any applications. Adjustments are made through `pam.conf`, the PAM configuration file.

The following figure illustrates the PAM architecture. Applications communicate with the PAM library through the PAM application programming interface (API). PAM modules communicate with the PAM library through the PAM service provider interface (SPI). Thus, the PAM library enables applications and modules to communicate with each other.
The Solaris 10 release includes the following changes to the Pluggable Authentication Module (PAM) framework:

- The `pam_authtok_check` module now allows for strict password checking using new tunable parameters in the `/etc/default/passwd` file. The new parameters define:
  - A list of comma separated dictionary files used for checking common dictionary words in a password
  - The minimum differences required between a new password and an old password
  - The minimum number of alphabetic or nonalphabetic characters that must be used in a new password
  - The minimum number of uppercase or lowercase letters that must be used in a new password
  - The number of allowable consecutive repeating characters

- The `pam_unix_auth` module implements account locking for local users. Account locking is enabled by the `LOCK_AFTER_RETRIES` parameter in `/etc/security/policy.conf` and the `lock_after_retries` key in `/etc/user_attr`. See the `policy.conf(4)` and the `user_attr(4)` man pages for more information.

- A new binding control flag has been defined. This control flag is documented in the `pam.conf(4)` man page and in “How PAM Stacking Works” on page 310.
The `pam_unix` module has been removed and replaced by a set of service modules of equivalent or greater functionality. Many of these modules were introduced in the Solaris 9 release. Here is a list of the replacement modules:

- `pam_authtok_check`
- `pam_authtok_get`
- `pam_authtok_store`
- `pam_dhkeys`
- `pam_passwd_auth`
- `pam_unix_account`
- `pam_unix_auth`
- `pam_unix_cred`
- `pam_unix_session`

The functionality of the `pam_unix_auth` module has been split into two modules. The `pam_unix_auth` module now verifies that the password is correct for the user. The new `pam_unix_cred` module provides functions that establish user credential information.

Additions to the `pam_krb5` module have been made to manage the Kerberos credentials cache using the PAM framework.

A new `pam_deny` module has been added. The module can be used to deny access to services. By default, the `pam_deny` module is not used. For more information, see the `pam_deny(5)` man page.

Changes to PAM for the Solaris Express 1/06 Release

The PAM framework for the Solaris Express 1/06 release includes a new `pam_allow` module. The module can be used to grant access to all users, without enforcing any security. The module should be used with caution. For more information, see the `pam_allow(5)` man page.

PAM (Tasks)

This section discusses some tasks that might be required to make the PAM framework use a particular security policy. You should be aware of some security issues that are associated with the PAM configuration file. For information about the security issues, see “Planning for Your PAM Implementation” on page 307.
PAM (Task Map)

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plan for your PAM installation.</td>
<td>Consider configuration issues and make decisions about them before you start the software configuration process.</td>
<td>“Planning for Your PAM Implementation” on page 307</td>
</tr>
<tr>
<td>Add new PAM modules.</td>
<td>Sometimes, site-specific modules must be written and installed to cover requirements that are not part of the generic software. This procedure explains how to install these new PAM modules.</td>
<td>“How to Add a PAM Module” on page 308</td>
</tr>
<tr>
<td>Block access through ~/.rhosts.</td>
<td>Further increase security by preventing access through ~/.rhosts.</td>
<td>“How to Prevent Rhost-Style Access From Remote Systems With PAM” on page 309</td>
</tr>
<tr>
<td>Initiate error logging.</td>
<td>Start the logging of PAM error messages through syslog.</td>
<td>“How to Log PAM Error Reports” on page 309</td>
</tr>
</tbody>
</table>

Planning for Your PAM Implementation

As delivered, the pam.conf configuration file implements the standard Solaris security policy. This policy should work in many situations. If you need to implement a different security policy, here are the issues that you should focus on:

- Determine what your needs are, especially which PAM service modules you should select.
- Identify the services that need special configuration options. Use other if appropriate.
- Decide the order in which the modules should be run.
- Select the control flag for each module. See "How PAM Stacking Works" on page 310 for more information about all of the control flags.
- Choose any options that are necessary for each module. The man page for each module should list any special options.

Here are some suggestions to consider before you change the PAM configuration file:

- Use other entries for each module type so that every application does not have to be included in /etc/pam.conf.
- Make sure to consider the security implications of the binding, sufficient, and optional control flags.
- Review the man pages that are associated with the modules. These man pages can help you understand how each module functions, what options are available, and the interactions between stacked modules.
Caution – If the PAM configuration file is misconfigured or the file becomes corrupted, no user might be able to log in. Because the su log in command does not use PAM, the root password would then be required to boot the machine into single-user mode and fix the problem.

After you change the /etc/pam.conf file, review the file as much as possible while you still have system access to correct problems. Test all the commands that might have been affected by your changes. An example is adding a new module to the telnet service. In this example, you would use the telnet command and verify that your changes make the service behave as expected.

▼ How to Add a PAM Module

This procedure shows how to add a new PAM module. New modules can be created to cover site-specific security policies or to support third party applications.

1 Become superuser or assume an equivalent role.
   Roles contain authorizations and privileged commands. For more information about roles, see “Configuring RBAC (Task Map)” on page 174.

2 Determine which control flags and which other options should be used.
   Refer to “How PAM Stacking Works” on page 310 for information on the control flags.

3 Ensure that the ownership and permissions are set so that the module file is owned by root and the permissions are 555.

4 Edit the PAM configuration file, /etc/pam.conf, and add this module to the appropriate services.

5 Verify that the module has been added properly.
   You must test before the system is rebooted in case the configuration file is misconfigured. Login using a direct service, such as ssh, and run the su command, before you reboot the system. The service might be a daemon that is spawned only once when the system is booted. Then, you must reboot the system before you can verify that the module has been added.
How to Prevent Rhost-Style Access From Remote Systems With PAM

1. **Become superuser or assume an equivalent role.**
   Roles contain authorizations and privileged commands. For more information about roles, see “Configuring RBAC (Task Map)” on page 174.

2. **Remove all of the lines that include rhosts_auth.so.1 from the PAM configuration file.**
   This step prevents the reading of the ~/.rhosts files during an rlogin session. Therefore, this step prevents unauthenticated access to the local system from remote systems. All rlogin access requires a password, regardless of the presence or contents of any ~/.rhosts or /etc/hosts.equiv files.

3. **Disable the rsh service.**
   To prevent other unauthenticated access to the ~/.rhosts files, remember to disable the rsh service.
   
   ```
   # svcadm disable network/shell
   ```

How to Log PAM Error Reports

1. **Become superuser or assume an equivalent role.**
   Roles contain authorizations and privileged commands. For more information about roles, see “Configuring RBAC (Task Map)” on page 174.

2. **Configure the /etc/syslog.conf file for the level of logging that you need.**
   See the syslog.conf(4) for more information about the logging levels.

3. **Refresh the configuration information for the syslog daemon.**
   
   ```
   # svcadm refresh system/system-log
   ```

PAM Configuration (Reference)

The PAM configuration file, **pam.conf(4)**, is used to configure PAM service modules for system services, such as login, rlogin, su, and cron. The system administrator manages this file. An incorrect order of entries in pam.conf can cause unforeseen side effects. For example, a badly configured pam.conf can lock out users so that single-user mode becomes necessary for repair. For a description of setting the order, see “How PAM Stacking Works” on page 310.
PAM Configuration File Syntax

The entries in the configuration file are in the format:

```
  service-name module-type control-flag module-path module-options
```

**service-name**
Name of the service, for example, `ftp`, `login`, or `passwd`. An application can use different service names for the services that the application provides. For example, the Solaris secure shell daemon uses these service names: `sshd-none`, `sshd-password`, `sshd-kbdint`, `sshd-pubkey`, and `sshd-hostbased`. The service-name `other` is a predefined name that is used as a wildcard service-name. If a particular service-name is not found in the configuration file, the configuration for `other` is used.

**module-type**
The type of service, that is, `auth`, `account`, `session`, or `password`.

**control-flag**
Indicates the role of the module in determining the integrated success or failure value for the service. Valid control flags are `binding`, `include`, `optional`, `required`, `requisite`, and `sufficient`. See “How PAM Stacking Works” on page 310 for information on the use of these flags.

**module-path**
The path to the library object that implements the service. If the pathname is not absolute, the pathname is assumed to be relative to `/usr/lib/security/$ISA/`. Use the architecture-dependent macro `$ISA` to cause `libpam` to look in the directory for the particular architecture of the application.

**module-options**
Options that are passed to the service modules. A module's man page describes the options that are accepted by that module. Typical module options include `nowarn` and `debug`.

How PAM Stacking Works

When an application calls on the following functions, `libpam` reads the configuration file `/etc/pam.conf` to determine which modules participate in the operation for this service:

- `pam_authenticate(3PAM)`
- `pam_acct_mgmt(3PAM)`
- `pam_setcred(3PAM)`
- `pam_open_session(3PAM)`
- `pam_close_session(3PAM)`
- `pam_chauthtok(3PAM)`
If /etc/pam.conf contains only one module for an operation for this service such as authentication or account management, the result of that module determines the outcome of the operation. For example, the default authentication operation for the passwd application contains one module, pam_passwd_auth.so.1:

```
passwd auth required pam_passwd_auth.so.1
```

If, on the other hand, there are multiple modules defined for the service’s operation, those modules are said to be stacked and that a PAM stack exists for that service. For example, consider the case where pam.conf contains the following entries:

```
login auth requisite pam_authtok_get.so.1
login auth required pam_dhkeys.so.1
login auth required pam_unix_cred.so.1
login auth required pam_unix_auth.so.1
login auth required pam_dial_auth.so.1
```

These entries represent a sample auth stack for the login service. To determine the outcome of this stack, the result codes of the individual modules require an integration process. In the integration process, the modules are executed in order as specified in /etc/pam.conf. Each success or failure code is integrated in the overall result depending on the module’s control flag. The control flag can cause early termination of the stack. For example, a requisite module might fail, or a sufficient or binding module might succeed. After the stack has been processed, the individual results are combined into a single, overall result that is delivered to the application.

The control flag indicates the role that a PAM module plays in determining access to the service. The control flags and their effects are:

- **Binding** – Success in meeting a binding module’s requirements returns success immediately to the application if no previous required modules have failed. If these conditions are met, then no further execution of modules occurs. Failure causes a required failure to be recorded and the processing of modules to be continued.

- **Include** – Adds lines from a separate PAM configuration file to be used at this point in the PAM stack. This flag does not control success or failure behaviors. When a new file is read, the PAM include stack is incremented. When the stack check in the new file finishes, the include stack value is decremented. When the end of a file is reached and the PAM include stack is 0, then the stack processing ends. The maximum number for the PAM include stack is 32.

- **Optional** – Success in meeting an optional module’s requirements is not necessary for using the service. Failure causes an optional failure to be recorded.
- **Required** – Success in meeting a required module’s requirements is necessary for using the service. Failure results in an error return after the remaining modules for this service have been executed. Final success for the service is returned only if no binding or required modules have reported failures.

- **Requisite** – Success in meeting a requisite module’s requirements is necessary for using the service. Failure results in an immediate error return with no further execution of modules. All requisite modules for a service must return success for the function to be able to return success to the application.

- **Sufficient** – If no previous required failures have occurred, success in a sufficient module returns success to the application immediately with no further execution of modules. Failure causes an optional failure to be recorded.

The following two diagrams shows how access is determined in the integration process. The first diagram indicates how success or failure is recorded for each type of control flag. The second diagram shows how the integrated value is determined.
Execute next module in PAM configuration file. Determine control flag.

---

**Figure 17-2**  PAM Stacking: Effect of Control Flags
Consider the following example of an `rlogin` service that requests authentication.

**EXAMPLE 17-1  Partial Contents of a Typical PAM Configuration File**

The `pam.conf` file in this example has the following contents for `rlogin` services:

```plaintext
# Authentication management
...
# rlogin service
rlogin  auth sufficient  pam_rhosts_auth.so.1
```

**FIGURE 17-3  PAM Stacking: How Integrated Value Is Determined**

**PAM Stacking Example**

Consider the following example of an `rlogin` service that requests authentication.

**EXAMPLE 17-1  Partial Contents of a Typical PAM Configuration File**

The `pam.conf` file in this example has the following contents for `rlogin` services:

```plaintext
# Authentication management
...
# rlogin service
rlogin  auth sufficient  pam_rhosts_auth.so.1
```
When the `rlogin` service requests authentication, `libpam` first executes the `pam_rhosts_auth(5)` module. The control flag is set to `sufficient` for the `pam_rhosts_auth` module. If the `pam_rhosts_auth` module is able to authenticate the user, then processing stops and success is returned to the application.

If the `pam_rhosts_auth` module fails to authenticate the user, then the next PAM module, `pam_authtok_get(5)` is executed. The control flag for this module is set to `requisite`. If `pam_authtok_get` fails, then the authentication process ends and the failure is returned to `rlogin`.

If `pam_authtok_get` succeeds, then the next two modules, `pam_dhkeys(5)` and `pam_unix_auth(5)`, are executed. Both modules have the associated control flags that are set to `required` so that the process continues regardless of whether an individual failure is returned. After `pam_unix_auth` is executed, no modules for `rlogin` authentication remain. At this point, if either `pam_dhkeys` or `pam_unix_auth` has returned a failure, the user is denied access through `rlogin`.

```
rlogin auth requisite       pam_authtok_get.so.1
rlogin auth required       pam_dhkeys.so.1
rlogin auth required       pam_unix_auth.so.1
...                       
```
This chapter includes information about the Simple Authentication and Security Layer (SASL).

- “SASL (Overview)” on page 317
- “SASL (Reference)” on page 318

**SASL (Overview)**

The Simple Authentication and Security Layer (SASL) is a framework that provides authentication and optional security services to network protocols. An application calls the SASL library, `/usr/lib/libsasl.so`, which provides a glue layer between the application and the various SASL mechanisms. The mechanisms are used in the authentication process and in providing optional security services. The version of SASL delivered with the Solaris 10 release is derived from the Cyrus SASL with a few changes.

SASL provides the following services:

- Loading of any plug-ins
- Determining the necessary security options from the application to aid in the choice of a security mechanism
- Listing of plug-ins that are available to the application
- Choosing the best mechanism from a list of available mechanisms for a particular authentication attempt
- Routing the authentication data between the application and the chosen mechanism
- Providing information about the SASL negotiation back to the application
The following section provides information about the implementation of SASL for the Solaris 10 release.

**SASL Plug-ins**

SASL plug-ins provide support for security mechanisms, user-canonicalization, and auxiliary property retrieval. By default, the dynamically loaded 32-bit plug-ins are installed in /usr/lib/sasl, and the 64-bit plug-ins are installed in /usr/lib/sasl/$ISA. The following security mechanism plug-ins are provided in the Solaris 10 release:

- **crammd5.so.1** CRAM-MD5, which supports authentication only, no authorization
- **digestmd5.so.1** DIGEST-MD5, which supports authentication, integrity, and privacy, as well as authorization
- **gssapi.so.1** GSSAPI, which supports authentication, integrity, and privacy, as well as authorization. The GSSAPI security mechanism requires a functioning Kerberos infrastructure.
- **plain.so.1** PLAIN, which supports authentication and authorization.

In addition, the EXTERNAL security mechanism plug-in and the INTERNAL user canonicalization plug-ins are built into libsasl.so.1. The EXTERNAL mechanism supports authentication and authorization. The mechanism supports integrity and privacy if the external security source provides it. The INTERNAL plug-in adds the realm name if necessary to the username.

The Solaris 10 release is not supplying any auxprop plug-ins at this time. For the CRAM-MD5 and DIGEST-MD5 mechanism plug-ins to be fully operational on the server side, the user must provide an auxprop plug-in to retrieve clear text passwords. The PLAIN plug-in requires additional support to verify the password. The support for password verification can be one of the following: a callback to the server application, an auxprop plug-in, saslauthd, or pwcheck. The saslauthd and pwcheck daemons are not provided in the Solaris releases. For better interoperability, restrict server applications to those mechanisms that are fully operational by using the mech_list SASL option.

**SASL Environment Variable**

By default, the client authentication name is set to getenv("LOGNAME"). This variable can be reset by the client or by the plug-in.
SASL Options

The behavior of libsasl and the plug-ins can be modified on the server side by using options that can be set in the /etc/sasl/app.conf file. The variable app is the server-defined name for the application. The documentation for the server app should specify the application name.

The following options are supported in the Solaris 10 release:

- **auto_transition**
  Automatically transitions the user to other mechanisms when the user does a successful plain text authentication.

- **auxprop_login**
  Lists the name of auxiliary property plug-ins to use.

- **canon_user_plugin**
  Selects the canon_user plug-in to use.

- **mech_list**
  Lists the mechanisms that are allowed to be used by the server application.

- **pwcheck_method**
  Lists the mechanisms used to verify passwords. Currently, auxprop is the only allowed value.

- **reauth_timeout**
  Sets the length of time, in minutes, that authentication information is cached for a fast reauthentication. This option is used by the DIGEST-MD5 plug-in. Setting this option to 0 disables reauthentication.

The following options are not supported in the Solaris 10 release:

- **plugin_list**
  Lists available mechanisms. Not used because the option changes the behavior of the dynamic loading of plugins.

- **saslauthd_path**
  Defines the location of the saslauthd door, which is used for communicating with the saslauthd daemon. The saslauthd daemon is not included in the Solaris 10 release. So, this option is also not included.

- **keytab**
  Defines the location of the keytab file used by the GSSAPI plug-in. Use the KRB5_KTNAME environment variable instead to set the default keytab location.

The following options are options not found in Cyrus SASL. However, they have been added for the Solaris 10 release:

- **use_authid**
  Acquire the client credentials rather than use the default credentials when creating the GSS client security context. By default, the default client Kerberos identity is used.

- **log_level**
  Sets the desired level of logging for a server.
Solaris Secure Shell enables a user to securely access a remote host over an unsecured network. The shell provides commands for remote login and remote file transfer. The following is a list of topics in this chapter.

- "Solaris Secure Shell (Overview)" on page 321
- "Solaris Secure Shell and the OpenSSH Project" on page 324
- "Configuring Solaris Secure Shell (Task Map)" on page 325
- "Using Solaris Secure Shell (Task Map)" on page 331

For reference information, see Chapter 20, "Solaris Secure Shell (Reference)." For information on the relationship of Solaris Secure Shell to the OpenSSH project, see "Solaris Secure Shell and the OpenSSH Project" on page 324.

**Solaris Secure Shell (Overview)**

In Solaris Secure Shell, authentication is provided by the use of passwords, public keys, or both. All network traffic is encrypted. Thus, Solaris Secure Shell prevents a would-be intruder from being able to read an intercepted communication. Solaris Secure Shell also prevents an adversary from spoofing the system.

Solaris Secure Shell can also be used as an on-demand virtual private network (VPN). A VPN can forward X Window system traffic or can connect individual port numbers between the local machines and remote machines over an encrypted network link.

With Solaris Secure Shell, you can perform these actions:

- Log in to another host securely over an unsecured network.
- Copy files securely between the two hosts.
- Run commands securely on the remote host.

Solaris Secure Shell supports two versions of the Secure Shell protocol. Version 1 is the original version of the protocol. Version 2 is more secure, and it amends some of the basic security
design flaws of version 1. Version 1 is provided only to assist users who are migrating to version 2. Users are strongly discouraged from using version 1.

Note – Hereafter in this text, v1 is used to represent version 1, and v2 is used to represent version 2.

Solaris Secure Shell Authentication

Solaris Secure Shell provides public key and password methods for authenticating the connection to the remote host. Public key authentication is a stronger authentication mechanism than password authentication, because the private key never travels over the network.

The authentication methods are tried in the following order. When the configuration does not satisfy an authentication method, the next method is tried.

- **GSS-API** – Uses credentials for GSS-API mechanisms such as mech_krb5 (Kerberos V) and mech_dh (AUTH_DH) to authenticate clients and servers. For more information on GSS-API, see “Introduction to GSS-API” in Solaris Security for Developers Guide.

- **Host-based authentication** – Uses host keys and rhosts files. Uses the client’s RSA and DSA public/private host keys to authenticate the client. Uses the rhosts files to authorize clients to users.

- **Public key authentication** – Authenticates users with their RSA and DSA public/private keys.

- **Password authentication** – Uses PAM to authenticate users. Keyboard authentication method in v2 allows for arbitrary prompting by PAM. For more information, see the SECURITY section in the sshd(1M) man page.

The following table shows the requirements for authenticating a user who is trying to log into a remote host. The user is on the local host, the client. The remote host, the server, is running the sshd daemon. The table shows the Solaris Secure Shell authentication methods, the compatible protocol versions, and the host requirements.

<table>
<thead>
<tr>
<th>Authentication Method (Protocol Version)</th>
<th>Local Host (Client) Requirements</th>
<th>Remote Host (Server) Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSS-API (v2)</td>
<td>Initiator credentials for the GSS mechanism.</td>
<td>Acceptor credentials for the GSS mechanism. For more information, see “Acquiring GSS Credentials in Solaris Secure Shell” on page 344.</td>
</tr>
<tr>
<td>Authentication Method (Protocol Version)</td>
<td>Local Host (Client) Requirements</td>
<td>Remote Host (Server) Requirements</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>---------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>Host-based (v2)</td>
<td>User account</td>
<td>User account</td>
</tr>
<tr>
<td>Local host private key in /etc/ssh/ssh_host_rsa_key or /etc/ssh/ssh_host_dsa_key</td>
<td>HostbasedAuthentication yes in /etc/ssh/ssh_config</td>
<td>Local host public key in /etc/ssh/known_hosts or ~/.ssh/known_hosts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HostbasedAuthentication yes in /etc/ssh/sshd_config</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Local host entry in /etc/ssh/shosts.equiv, /etc/hosts.equiv, ~/.rhosts, or ~/.shosts</td>
</tr>
<tr>
<td>RSA or DSA public key (v2)</td>
<td>User account</td>
<td>User account</td>
</tr>
<tr>
<td>Private key in ~/.ssh/id_rsa or ~/.ssh/id_dsa</td>
<td>User's public key in ~/.ssh/authorized_keys</td>
<td>User's public key in ~/.ssh/authorized_keys</td>
</tr>
<tr>
<td>User's public key in ~/.ssh/id_rsa.pub or ~/.ssh/id_dsa.pub</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RSA public key (v1)</td>
<td>User account</td>
<td>User account</td>
</tr>
<tr>
<td>User's public key in ~/.ssh/identity.pub</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Keyboard-interactive (v2)</td>
<td>User account</td>
<td>User account</td>
</tr>
<tr>
<td></td>
<td>Supports PAM, including arbitrary prompting and password changing when password aging is triggered.</td>
<td></td>
</tr>
<tr>
<td>Password-based (v1 or v2)</td>
<td>User account</td>
<td>User account</td>
</tr>
<tr>
<td></td>
<td>Supports PAM.</td>
<td></td>
</tr>
<tr>
<td>.rhosts only (v1)</td>
<td>User account</td>
<td>User account</td>
</tr>
<tr>
<td></td>
<td>IgnoreRhosts no in /etc/ssh/sshd_config</td>
<td>IgnoreRhosts no in /etc/ssh/sshd_config</td>
</tr>
<tr>
<td></td>
<td>Local host entry in /etc/ssh/shosts.equiv, /etc/hosts.equiv, ~/.rhosts, or ~/.shosts</td>
<td>Local host entry in /etc/ssh/shosts.equiv, /etc/hosts.equiv, ~/.rhosts, or ~/.shosts</td>
</tr>
<tr>
<td>.rhosts with RSA (v1) on server only</td>
<td>User account</td>
<td>User account</td>
</tr>
<tr>
<td>Local host public key in /etc/ssh/ssh_host_rsa1_key</td>
<td>Local host public key in /etc/ssh/sshd_config</td>
<td>IgnoreRhosts no in /etc/ssh/sshd_config</td>
</tr>
<tr>
<td></td>
<td>Local host entry in /etc/ssh/shosts.equiv, /etc/hosts.equiv, ~/.rhosts, or ~/.shosts</td>
<td>Local host entry in /etc/ssh/shosts.equiv, /etc/hosts.equiv, ~/.rhosts, or ~/.shosts</td>
</tr>
</tbody>
</table>
Solaris Secure Shell in the Enterprise

For a comprehensive discussion of Secure Shell on a Solaris system, see *Secure Shell in the Enterprise*, by Jason Reid, ISBN 0-13-142900-0, June 2003. The book is part of the Sun BluePrints Series, which is published by Sun Microsystems Press.

For online information, navigate to Sun’s BigAdmin System Administration Portal web site, http://www.sun.com/bigadmin/home/index.jsp. Click Docs, then Sun BluePrints under Misc./Comprehensive. Click Sun BluePrints OnLine, then Archives by Subject, then Security. The archives include the following articles:

- Role Based Access Control and Secure Shell – A Closer Look At Two Solaris Operating Environment Security Features
- Integrating the Secure Shell Software
- Configuring the Secure Shell Software

Solaris Secure Shell and the OpenSSH Project

The Solaris Secure Shell is a fork of the OpenSSH (http://www.openssh.com) project. Security fixes for vulnerabilities that are discovered in later versions of OpenSSH are integrated into Solaris Secure Shell, as are individual bug fixes and features. Internal development continues on the Solaris Secure Shell fork.

While Sun engineers provide bug fixes to the project, they have also integrated the following Solaris features into the Solaris fork of Secure Shell:

- PAM - Solaris Secure Shell uses PAM. The OpenSSH UsePAM configuration option is not supported.
- Privilege separation - Solaris Secure Shell does not use the privilege separation code from the OpenSSH project. Solaris Secure Shell separates the processing of auditing, record keeping and re-keying from the processing of the session protocols.
  
  Solaris Secure Shell privilege separation code is always on and cannot be switched off. The OpenSSH UsePrivilegeSeparation option is not supported.
- Locale - Solaris Secure Shell fully supports language negotiation as defined in RFC 4253, Secure Shell Transfer Protocol. After the user logs in, the user’s login shell profile can override the Solaris Secure Shell negotiated locale settings.
- Auditing - Solaris Secure Shell is fully integrated into the Solaris auditing subsystem. For information on auditing, see Part VII, “Solaris Auditing.”
- GSS-API support - GSS-API can be used for user authentication and for initial key exchange. The GSS-API is defined in RFC4462, Generic Security Service Application Program Interface.
Proxy commands - Solaris Secure Shell provides proxy commands for SOCKS5 and HTTP protocols. For an example, see "How to Set Up Default Connections to Hosts Outside a Firewall" on page 340.

In the Solaris Express Community Edition and the OpenSolaris releases, Solaris Secure Shell resyncs the SSH_OLD_FORWARD_ADDR compatibility flag from the OpenSSH project. As of March 2009, the Solaris Secure Shell version is 1.3.

Solaris Secure Shell (Task Map)

The following task map points to task maps for configuring Solaris Secure Shell and for using Solaris Secure Shell.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure Solaris Secure Shell</td>
<td>Guides administrators in configuring Solaris Secure Shell for users.</td>
<td>&quot;Configuring Solaris Secure Shell (Task Map)&quot; on page 325</td>
</tr>
<tr>
<td>Use Solaris Secure Shell</td>
<td>Guides users in using Solaris Secure Shell.</td>
<td>&quot;Using Solaris Secure Shell (Task Map)&quot; on page 331</td>
</tr>
</tbody>
</table>

Configuring Solaris Secure Shell (Task Map)

The following task map points to procedures for configuring Solaris Secure Shell.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure host-based authentication</td>
<td>Configures host-based authentication on the client and server.</td>
<td>&quot;How to Set Up Host-Based Authentication for Solaris Secure Shell&quot; on page 326</td>
</tr>
<tr>
<td>Configure a host to use v1 and v2</td>
<td>Creates public key files for hosts that use v1 and v2 protocols.</td>
<td>&quot;How to Enable Solaris Secure Shell v1&quot; on page 328</td>
</tr>
<tr>
<td>Configure port forwarding</td>
<td>Enables users to use port forwarding.</td>
<td>&quot;How to Configure Port Forwarding in Solaris Secure Shell&quot; on page 329</td>
</tr>
<tr>
<td>Configure exceptions to SSH system defaults</td>
<td>For users, hosts, groups, and addresses, specifies SSH settings that are different from the system defaults.</td>
<td>&quot;How to Create User and Host Exceptions to SSH System Defaults&quot; on page 330</td>
</tr>
</tbody>
</table>
Configuring Solaris Secure Shell

By default, host-based authentication and the use of both protocols are not enabled in Solaris Secure Shell. Changing these defaults requires administrative intervention. Also, for port forwarding to work requires administrative intervention.

▼ How to Set Up Host-Based Authentication for Solaris Secure Shell

The following procedureset up a public key system where the client’s public key is used for authentication on the server. The user must also create a public/private key pair.

In the procedure, the terms client and local host refer to the machine where a user types the ssh command. The terms server and remote host refer to the machine that the client is trying to reach.

1 Assume the Primary Administrator role, or become superuser.
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in System Administration Guide: Basic Administration.

2 On the client, enable host-based authentication.
   In the client configuration file, /etc/ssh/ssh_config, type the following entry:
   
   ```
   HostbasedAuthentication yes
   ```
   
   For the syntax of the file, see the ssh_config(4) man page

3 On the server, enable host-based authentication.
   In the server configuration file, /etc/ssh/sshd_config, type the same entry:
   
   ```
   HostbasedAuthentication yes
   ```
   
   For the syntax of the file, see the sshd_config(4) man page

4 On the server, configure a file that enables the client to be recognized as a trusted host.
   For more information, see the FILES section of the sshd(1M) man page.
   
   - Add the client as an entry to the server’s /etc/ssh/hosts.equiv file.
     ```
     client-host
     ```
   - Or, you can instruct users to add an entry for the client to their ~/.ssh/hosts file on the server.
     ```
     client-host
     ```
5. On the server, ensure that the `sshd` daemon can access the list of trusted hosts.
   Set `IgnoreRhosts` to `no` in the `/etc/ssh/sshd_config` file.
   ```
   ## sshd_config
   IgnoreRhosts   no
   ```

6. Ensure that users of Solaris Secure Shell at your site have accounts on both hosts.

7. Do one of the following to put the client's public key on the server.
   - **Modify the `sshd_config` file on the server, then instruct your users to add the client's public host keys to their `~/.ssh/known_hosts` file.**
     ```
     ## sshd_config
     IgnoreUserKnownHosts no
     ```
     For user instructions, see "How to Generate a Public/Private Key Pair for Use With Solaris Secure Shell" on page 332.
   - **Copy the client's public key to the server.**
     The host keys are stored in the `/etc/ssh` directory. The keys are typically generated by the `sshd` daemon on first boot.
     - **Add the key to the `/etc/ssh/ssh_known_hosts` file on the server.**
       On the client, type the command on one line with no backslash.
       ```
       # cat /etc/ssh/ssh_host_dsa_key.pub | ssh RemoteHost \ 'cat >> /etc/ssh/ssh_known_hosts && echo "Host key copied"'
       ```
     - **When you are prompted, supply your login password.**
       When the file is copied, the message “Host key copied” is displayed.
       Each line in the `/etc/ssh/ssh_known_hosts` file consists of fields that are separated by spaces:
       ```
       hostnames algorithm-name publickey comment
       ```
     - **Edit the `/etc/ssh/ssh_known_hosts` file and add `RemoteHost` as the first field in the copied entry.**
       ```
       ## /etc/ssh/ssh_known_hosts File
       RemoteHost <copied entry>
       ```
Example 19–1 Setting Up Host-based Authentication

In the following example, each host is configured as a server and as a client. A user on either host can initiate an ssh connection to the other host. The following configuration makes each host a server and a client:

- On each host, the Solaris Secure Shell configuration files contain the following entries:
  ```
  ## /etc/ssh/ssh_config
  HostBasedAuthentication yes
  #
  ## /etc/ssh/sshd_config
  HostBasedAuthentication yes
  IgnoreRhosts no
  ```

- On each host, the `shosts.equiv` file contains an entry for the other host:
  ```
  ## /etc/ssh/shosts.equiv on machine2
  machine1
  ## /etc/ssh/shosts.equiv on machine1
  machine2
  ```

- The public key for each host is in the `/etc/ssh/ssh_known_hosts` file on the other host:
  ```
  ## /etc/ssh/ssh_known_hosts on machine2
  ... machine1
  ## /etc/ssh/ssh_known_hosts on machine1
  ... machine2
  ```

- Users have an account on both hosts:
  ```
  ## /etc/passwd on machine1
  jdoe:x:3111:10:J Doe:/home/jdoe:/bin/sh
  ## /etc/passwd on machine2
  jdoe:x:3111:10:J Doe:/home/jdoe:/bin/sh
  ```

How to Enable Solaris Secure Shell v1

This procedure is useful when a host interoperates with hosts that run v1 and v2.

1 Assume the Primary Administrator role, or become superuser.

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in System Administration Guide: Basic Administration.
2 Configure the host to use both Solaris Secure Shell protocols.
   Edit the /etc/ssh/sshd_config file.
   # Protocol 2
   Protocol 2,1

3 Provide a separate file for the host key for v1.
   Add a HostKey entry to the /etc/ssh/sshd_config file.
   HostKey /etc/ssh/ssh_host_rsa_key
   HostKey /etc/ssh/ssh_host_dsa_key
   HostKey /etc/ssh/ssh_host_rsa1_key

4 Generate a host key for v1.
   # ssh-keygen -t rsa1 -f /etc/ssh/ssh_host_rsa1_key -N ''
   -t rsa1 Indicates the RSA algorithm for v1.
   -f Indicates the file that holds the host key.
   -N '' Indicates that no passphrase is required.

5 Restart the sshd daemon.
   # svcadm restart network/ssh:default
   You can also reboot the system.

▼ How to Configure Port Forwarding in Solaris Secure Shell

Port forwarding enables a local port be forwarded to a remote host. Effectively, a socket is allocated to listen to the port on the local side. Similarly, a port can be specified on the remote side.

Note – Solaris Secure Shell port forwarding must use TCP connections. Solaris Secure Shell does not support UDP connections for port forwarding.

1 Assume the Primary Administrator role, or become superuser.
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.
2  **Configure a Solaris Secure Shell setting on the remote server to allow port forwarding.**

Change the value of `AllowTcpForwarding` to `yes` in the `/etc/ssh/sshd_config` file.

```bash
# Port forwarding
AllowTcpForwarding yes
```

3  **Restart the Solaris Secure Shell service.**

```bash
remoteHost# svcadm restart network/ssh:default
```

For information on managing persistent services, see Chapter 16, "Managing Services (Overview)," in *System Administration Guide: Basic Administration* and the `svcadm(1M)` man page.

4  **Verify that port forwarding can be used.**

```bash
remoteHost# /usr/bin/pgrep -lf sshd
```

```
1296 ssh -L 2001:remoteHost:23 remoteHost
```

▼  **How to Create User and Host Exceptions to SSH System Defaults**

This procedure adds a conditional `Match` block after the global section of the `/etc/ssh/sshd_config` file. Keyword-value pairs that follow the `Match` block specify exceptions for the user, group, host, or address that is specified as the match.

1  **Assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2  **Edit the `sshd_config` file.**

3  **Configure a user, group, host, or address to use different SSH keyword settings from the default settings.**

Place the `Match` blocks after the global settings.

---

**Note** – The global section of the file might or might not list the default settings. For the defaults, see the `sshd_config(4)` man page.
You might have users who should not be allowed to use TCP forwarding. In the following example, any user in the group public, and any user name that begins with test cannot use TCP forwarding:

```bash
# Example (reflects default settings):
# Host *
# ForwardAgent no
# ForwardX11 no
# PubkeyAuthentication yes
# PasswordAuthentication yes
# FallBackToRsh no
# UseRsh no
# BatchMode no
# CheckHostIP yes
# StrictHostKeyChecking ask
# EscapeChar ~
Match Group public
    AllowTcpForwarding no
Match User test*
    AllowTcpForwarding no
```

For information about the syntax of the `Match` block, see the `sshd_config(4)` man page.

**Using Solaris Secure Shell (Task Map)**

The following task map points to user procedures for using Solaris Secure Shell.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a public/private key pair</td>
<td>Enables access to Solaris Secure Shell for sites that require public-key authentication.</td>
<td>&quot;How to Generate a Public/Private Key Pair for Use With Solaris Secure Shell&quot; on page 332</td>
</tr>
<tr>
<td>Change your passphrase</td>
<td>Changes the phrase that authenticates your private key.</td>
<td>&quot;How to Change the Passphrase for a Solaris Secure Shell Private Key&quot; on page 334</td>
</tr>
<tr>
<td>Log in with Solaris Secure Shell</td>
<td>Provides encrypted Solaris Secure Shell communication when logging in remotely. The process is similar to using the <code>rsh</code> command.</td>
<td>&quot;How to Log In to a Remote Host With Solaris Secure Shell&quot; on page 335</td>
</tr>
</tbody>
</table>
Using Solaris Secure Shell

Solaris Secure Shell provides secure access between a local shell and a remote shell. For more information, see the `ssh_config(4)` and `ssh(1)` man pages.

▼ How to Generate a Public/Private Key Pair for Use With Solaris Secure Shell

Users must generate a public/private key pair when their site implements host-based authentication or user public-key authentication. For additional options, see the `ssh-keygen(1)` man page.

**Before You Begin**

Determine from your system administrator if host-based authentication is configured.

1 **Start the key generation program.**

   ```bash
   myLocalHost% ssh-keygen -t rsa
   Generating public/private rsa key pair.
   ...
   ```

   where `-t` is the type of algorithm, one of `rsa`, `dsa`, or `rsa1`.

2 **Specify the path to the file that will hold the key.**

   By default, the file name `id_rsa`, which represents an RSA v2 key, appears in parentheses. You can select this file by pressing the Return key. Or, you can type an alternative file name.

   ```bash
   Enter file in which to save the key (/home/jdoe/.ssh/id_rsa): <Press Return>
   ```

   The file name of the public key is created automatically by appending the string `.pub` to the name of the private key file.
3  **Type a passphrase for using your key.**

   This passphrase is used for encrypting your private key. A null entry is *strongly discouraged.*
   Note that the passphrase is not displayed when you type it in.

   Enter passphrase (empty for no passphrase):  <Type passphrase>

4  **Retype the passphrase to confirm it.**

   Enter same passphrase again:  <Type passphrase>

   Your identification has been saved in /home/jdoe/.ssh/id_rsa.
   Your public key has been saved in /home/jdoe/.ssh/id_rsa.pub.
   The key fingerprint is:

5  **Check the results.**

   Check that the path to the key file is correct.

   % ls ~/.ssh
   id_rsa
   id_rsa.pub

   At this point, you have created a public/private key pair.

6  **Choose the appropriate option:**

   - If your administrator has configured host-based authentication, you might need to copy the local host's public key to the remote host.
     You can now log in to the remote host. For details, see "How to Log In to a Remote Host With Solaris Secure Shell" on page 335.

     a. Type the command on one line with no backslash.

     ```
     % cat /etc/ssh/ssh_host_dsa_key.pub | ssh RemoteHost
     'cat >> ~/.ssh/known_hosts & echo "Host key copied"
     ```

     b. When you are prompted, supply your login password.

     Enter password:  <Type password>

     Host key copied

     %

   - If your site uses user authentication with public keys, populate your `authorized_keys` file on the remote host.

     a. Copy your public key to the remote host.

     Type the command on one line with no backslash.

     ```
     myLocalHost% cat $HOME/.ssh/id_rsa.pub | ssh myRemoteHost
     'cat >> .ssh/authorized_keys & echo "Key copied"
     ```
b. When you are prompted, supply your login password.

When the file is copied, the message “Key copied” is displayed.

```
Enter password: Type login password
Key copied
myLocalHost%
```

7 (Optional) Reduce the prompting for passphrases.

For a procedure, see “How to Reduce Password Prompts in Solaris Secure Shell” on page 336. For more information, see the `ssh-agent(1)` and `ssh-add(1)` man pages.

Example 19–2 Establishing a v1 RSA Key for a User

In the following example, the user can contact hosts that run v1 of the Solaris Secure Shell protocol. To be authenticated by v1 hosts, the user creates a v1 key, then copies the public key portion to the remote host.

```
myLocalHost% ssh-keygen -t rsa1 -f /home/jdoe/.ssh/identity
Generating public/private rsa key pair.
...
Enter passphrase (empty for no passphrase): <Type passphrase>
Enter same passphrase again: <Type passphrase>
Your identification has been saved in /home/jdoe/.ssh/identity.
Your public key has been saved in /home/jdoe/.ssh/identity.pub.
The key fingerprint is:
...
myLocalHost% ls ~/.ssh
id_rsa
id_rsa.pub
identity
identity.pub
myLocalHost% cat $HOME/.ssh/identity.pub | ssh myRemoteHost \
'cat >> .ssh/authorized_keys && echo "Key copied"'
```

▼ How to Change the Passphrase for a Solaris Secure Shell Private Key

The following procedure does not change the private key. The procedure changes the authentication mechanism for the private key, the passphrase. For more information, see the `ssh-keygen(1)` man page.
● Change your passphrase.

Type the ssh-keygen command with the -p option, and answer the prompts.

```
myLocalHost% ssh-keygen -p
```

Enter file which contains the private key (/home/jdoe/.ssh/id_rsa): <Press Return>
Enter passphrase (empty for no passphrase):   <Type passphrase>
Enter same passphrase again:     <Type passphrase>

where -p requests changing the passphrase of a private key file.

▼ How to Log In to a Remote Host With Solaris Secure Shell

1 Start a Solaris Secure Shell session.

Type the ssh command, and specify the name of the remote host.

```
myLocalHost% ssh myRemoteHost
```

A prompt questions the authenticity of the remote host:

```
The authenticity of host 'myRemoteHost' can't be established.
Are you sure you want to continue connecting(yes/no)?
```

This prompt is normal for initial connections to remote hosts.

2 If prompted, verify the authenticity of the remote host key.

- If you cannot confirm the authenticity of the remote host, type no and contact your system administrator.

```
Are you sure you want to continue connecting(yes/no)? no
```

The administrator is responsible for updating the global /etc/ssh/ssh_known_hosts file.
An updated ssh_known_hosts file prevents this prompt from appearing.

- If you confirm the authenticity of the remote host, answer the prompt and continue to the next step.

```
Are you sure you want to continue connecting(yes/no)? yes
```

3 Authenticate yourself to Solaris Secure Shell.

a. When prompted, type your passphrase.

```
Enter passphrase for key '/home/jdoe/.ssh/id_rsa':   <Type passphrase>
```
b. When prompted, type your account password.

```
jdoe@myRemoteHost's password: <Type password>
Last login: Fri Jul 20 14:24:10 2001 from myLocalHost
```

myRemoteHost%

4 Conduct transactions on the remote host.

The commands that you send are encrypted. Any responses that you receive are encrypted.

5 Close the Solaris Secure Shell connection.

When you are finished, type `exit` or use your usual method for exiting your shell.

```
myRemoteHost% exit
myRemoteHost% logout
```

Connection to myRemoteHost closed

myLocalHost%

▼ How to Reduce Password Prompts in Solaris Secure Shell

If you do not want to type your passphrase and your password to use Solaris Secure Shell, you can use the agent daemon. Start the daemon at the beginning of the session. Then, store your private keys with the agent daemon by using the `ssh-add` command. If you have different accounts on different hosts, add the keys that you need for the session.

You can start the agent daemon manually when needed, as described in the following procedure.

1 Start the agent daemon.

```
myLocalHost% eval `ssh-agent`
Agent pid 9892
```

2 Verify that the agent daemon has been started.

```
myLocalHost% pgrep ssh-agent
9892
```

3 Add your private key to the agent daemon.

Type the `ssh-add` command.

```
myLocalHost% ssh-add
Enter passphrase for /home/jdoe/.ssh/id_rsa: <Type passphrase>
Identity added: /home/jdoe/.ssh/id_rsa (/home/jdoe/.ssh/id_rsa)
```

myLocalHost%
4 Start a Solaris Secure Shell session.

```
myLocalHost% ssh myRemoteHost
```

You are not prompted for a passphrase.

**Example 19–3 Using ssh-add Options**

In this example, jdoe adds two keys to the agent daemon. The `-l` option is used to list all keys that are stored in the daemon. At the end of the session, the `-D` option is used to remove all the keys from the agent daemon.

```
myLocalHost% ssh-agent
myLocalHost% ssh-add
Enter passphrase for /home/jdoe/.ssh/id_rsa: <Type passphrase>
Identity added: /home/jdoe/.ssh/id_rsa(/home/jdoe/.ssh/id_rsa)
myLocalHost% ssh-add /home/jdoe/.ssh/id_dsa
Enter passphrase for /home/jdoe/.ssh/id_dsa: <Type passphrase>
Identity added: /home/jdoe/.ssh/id_dsa(/home/jdoe/.ssh/id_dsa)
```

User conducts Solaris Secure Shell transactions

```
myLocalHost% ssh-add -l
/home/jdoe/.ssh/id_rsa(RSA)
/home/jdoe/.ssh/id_dsa(DSA)
```

How to Use Port Forwarding in Solaris Secure Shell

You can specify that a local port be forwarded to a remote host. Effectively, a socket is allocated to listen to the port on the local side. The connection from this port is made over a secure channel to the remote host. For example, you might specify port 143 to obtain email remotely with IMAP4. Similarly, a port can be specified on the remote side.

Before You Begin

To use port forwarding, the administrator must have enabled port forwarding on the remote Solaris Secure Shell server. For details, see “How to Configure Port Forwarding in Solaris Secure Shell” on page 329.
To use secure port forwarding, choose one of the following options:

- To set a local port to receive secure communication from a remote port, specify both ports. Specify the local port that listens for remote communication. Also, specify the remote host and the remote port that forward the communication.
  
  ```
  myLocalHost% ssh -L localPort:remoteHost:remotePort
  ```

- To set a remote port to receive a secure connection from a local port, specify both ports. Specify the remote port that listens for remote communication. Also, specify the local host and the local port that forward the communication.
  
  ```
  myLocalHost% ssh -R remotePort:localhost:localPort
  ```

**Example 19–4 Using Local Port Forwarding to Receive Mail**

The following example demonstrates how you can use local port forwarding to receive mail securely from a remote server.

```
myLocalHost% ssh -L 9143:myRemoteHost:143 myRemoteHost
```

This command forwards connections from port 9143 on `myLocalHost` to port 143. Port 143 is the IMAP v2 server port on `myRemoteHost`. When the user launches a mail application, the user needs to specify the local port number, as shown in the following dialog box.

```
<table>
<thead>
<tr>
<th>Mailer — Login</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMAP Server: localhost:9143</td>
</tr>
<tr>
<td>Login: jdoe</td>
</tr>
<tr>
<td>Password: *********</td>
</tr>
</tbody>
</table>

Retrieve attachments: Always | When Open
```

Do not confuse `localhost` in the dialog box with `myLocalHost`. `myLocalHost` is a hypothetical host name. `localhost` is a keyword that identifies your local system.
Example 19–5 Using Remote Port Forwarding to Communicate Outside of a Firewall

This example demonstrates how a user in an enterprise environment can forward connections from a host on an external network to a host inside a corporate firewall.

```
myLocalHost% ssh -R 9022:myLocalHost:22 myOutsideHost
```

This command forwards connections from port 9022 on myOutsideHost to port 22, the sshd server, on the local host.

```
myOutsideHost% ssh -p 9022 localhost
```


▼ ▼

How to Copy Files With Solaris Secure Shell

The following procedures show how to use the scp command to copy encrypted files between hosts. You can copy encrypted files either between a local host and a remote host, or between two remote hosts. The command operates similarly to the rcp command, except that the scp command prompts for authentication. For more information, see the scp(1) man page.

You can also use the sftp, a more secure form of the ftp command. For more information, see the sftp(1) man page. For an example, see Example 19–6.

1 Start the secure copy program.

Specify the source file, the username at the remote destination, and the destination directory.

```
myLocalHost% scp myfile.1 jdoe@myRemoteHost:
```

2 Supply your passphrase when prompted.

```
Enter passphrase for key '/home/jdoe/.ssh/id_rsa': <Type passphrase>
myfile.1 25% |********| 640 KB 0:20 ETA
```

After you type the passphrase, a progress meter is displayed. See the second line in the preceding output. The progress meter displays:

- The file name
- The percentage of the file that has been transferred
- A series of asterisks that indicate the percentage of the file that has been transferred
- The quantity of data transferred
- The estimated time of arrival, or ETA, of the complete file (that is, the remaining amount of time)
Specifying a Port When Using the `sftp` Command

In this example, the user wants the `sftp` command to use a specific port. The user uses the `-o` option to specify the port.

```
% sftp -o port=2222 guest@RemoteFileServer
```

How to Set Up Default Connections to Hosts Outside a Firewall

You can use Solaris Secure Shell to make a connection from a host inside a firewall to a host outside the firewall. This task is done by specifying a proxy command for `ssh` either in a configuration file or as an option on the command line. For the command-line option, see Example 19–7.

In general, you can customize your `ssh` interactions through a configuration file.

- You can customize either your own personal file in `~/.ssh/config`.
- Or, you can use the settings in the administrative configuration file, `/etc/ssh/ssh_config`.

The files can be customized with two types of proxy commands. One proxy command is for HTTP connections. The other proxy command is for SOCKS5 connections. For more information, see the `ssh_config(4)` man page.

Specify the proxy commands and hosts in a configuration file.

Use the following syntax to add as many lines as you need:

```
[Host outside-host]
ProxyCommand proxy-command [-h proxy-server] \ [-p proxy-port] outside-host|%h outside-port|%p

Host outside-host
  Limits the proxy command specification to instances when a remote host name is specified on the command line. If you use a wildcard for `outside-host`, you apply the proxy command specification to a set of hosts.

proxy-command
  Specifies the proxy command.

  The command can be either of the following:
  ■ `/usr/lib/ssh/ssh-http-proxy-connect` for HTTP connections
  ■ `/usr/lib/ssh/ssh-socks5-proxy-connect` for SOCKS5 connections

-h proxy-server and -p proxy-port
  These options specify a proxy server and a proxy port, respectively. If present, the proxies override any environment variables that specify proxy servers and proxy ports, such as `HTTPPROXY`, `HTTP_PROXYPORT`, `SOCKS5_PORT`, `SOCKS5_SERVER`, and `http_proxy`. The
http_proxy variable specifies a URL. If the options are not used, then the relevant environment variables must be set. For more information, see the ssh-socks5-proxy-connect(1) and ssh-http-proxy-connect(1) man pages.

outside-host
Designates a specific host to connect to. Use the %h substitution argument to specify the host on the command line.

outside-port
Designates a specific port to connect to. Use the %p substitution argument to specify the port on the command line. By specifying %h and %p without using the Host outside-host option, the proxy command is applied to the host argument whenever the ssh command is invoked.

2 Run Solaris Secure Shell, specifying the outside host.
For example, type the following:

myLocalHost% ssh myOutsideHost

This command looks for a proxy command specification for myOutsideHost in your personal configuration file. If the specification is not found, then the command looks in the system-wide configuration file, /etc/ssh/ssh_config. The proxy command is substituted for the ssh command.

Example 19-7 Connecting to Hosts Outside a Firewall From the Command Line

“How to Set Up Default Connections to Hosts Outside a Firewall” on page 340 explains how to specify a proxy command in a configuration file. In this example, a proxy command is specified on the ssh command line.

% ssh -o 'Proxycommand=/usr/lib/ssh/ssh-http-proxy-connect \ -h myProxyServer -p 8080 myOutsideHost 22' myOutsideHost

The -o option to the ssh command provides a command-line method of specifying a proxy command. This example command does the following:

+ Substitutes the HTTP proxy command for ssh
+ Uses port 8080 and myProxyServer as the proxy server
+ Connects to port 22 on myOutsideHost
This chapter describes the configuration options in Solaris Secure Shell. The following is a list of the reference information in this chapter.

- "A Typical Solaris Secure Shell Session" on page 343
- "Client and Server Configuration in Solaris Secure Shell" on page 346
- "Keywords in Solaris Secure Shell" on page 346
- "Maintaining Known Hosts in Solaris Secure Shell" on page 351
- "Solaris Secure Shell Packages and Initialization" on page 352
- "Solaris Secure Shell Files" on page 352
- "Solaris Secure Shell Commands" on page 354

For procedures to configure Solaris Secure Shell, see Chapter 19, "Using Solaris Secure Shell (Tasks)."

**A Typical Solaris Secure Shell Session**

The Solaris Secure Shell daemon (sshd) is normally started at boot time when network services are started. The daemon listens for connections from clients. A Solaris Secure Shell session begins when the user runs an ssh, scp, or sftp command. A new sshd daemon is forked for each incoming connection. The forked daemons handle key exchange, encryption, authentication, command execution, and data exchange with the client. These session characteristics are determined by client-side configuration files and server-side configuration files. Command-line arguments can override the settings in the configuration files.

The client and server must authenticate themselves to each other. After successful authentication, the user can execute commands remotely and copy data between hosts.
Session Characteristics in Solaris Secure Shell

The server-side behavior of the sshd daemon is controlled by keyword settings in the /etc/ssh/sshd_config file. For example, the sshd_config file controls which types of authentication are permitted for accessing the server. The server-side behavior can also be controlled by the command-line options when the sshd daemon is started.

The behavior on the client side is controlled by Solaris Secure Shell keywords in this order of precedence:

- Command-line options
- User's configuration file, ~/.ssh/config
- System-wide configuration file, /etc/ssh/ssh_config

For example, a user can override a system-wide configuration Ciphers setting that prefers aes128–ctr by specifying -c aes256–ctr, aes128–ctr, arcfour on the command line. The first cipher, aes256–ctr, is now preferred.

Authentication and Key Exchange in Solaris Secure Shell

The Solaris Secure Shell protocols, v1 and v2, both support client user/host authentication and server host authentication. Both protocols involve the exchange of session cryptographic keys for the protection of Solaris Secure Shell sessions. Each protocol provides various methods for authentication and key exchange. Some methods are optional. Solaris Secure Shell supports a number of client authentication mechanisms, as shown in Table 19–1. Servers are authenticated by using known host public keys.

For the v1 protocol, Solaris Secure Shell supports user authentication with passwords. The protocol also supports user public keys and authentication with trusted host public keys. Server authentication is done with a host public key. For the v1 protocol, all public keys are RSA keys. Session key exchanges involve the use of an ephemeral server key that is periodically regenerated.

For the v2 protocol, Solaris Secure Shell supports user authentication and generic interactive authentication, which usually involves passwords. The protocol also supports authentication with user public keys and with trusted host public keys. The keys can be RSA or DSA. Session key exchanges consist of Diffie-Hellman ephemeral key exchanges that are signed in the server authentication step. Additionally, Solaris Secure Shell can use GSS credentials for authentication.

Acquiring GSS Credentials in Solaris Secure Shell

To use GSS-API for authentication in Solaris Secure Shell, the server must have GSS-API acceptor credentials and the client must have GSS-API initiator credentials. Support is available for mech_dh and for mech_krb5.
For mech_dh, the server has GSS-API acceptor credentials if root has run the keylogin command.

For mech_krb5, the server has GSS-API acceptor credentials when the host principal that corresponds to the server has a valid entry in /etc/krb5/krb5.keytab.

The client has initiator credentials for mech_dh if one of the following has been done:
- The keylogin command has been run.
- The pam_dhkeys module is used in the pam.conf file.

The client has initiator credentials for mech_krb5 if one of the following has been done:
- The kinit command has been run.
- The pam_krb5 module is used in the pam.conf file.

For the use of mech_dh in secure RPC, see Chapter 16, “Using Authentication Services (Tasks).” For the use of mech_krb5, see Chapter 21, “Introduction to the Kerberos Service.” For more information on mechanisms, see the mech(4) and mech_spnego(5) man pages.

**Command Execution and Data Forwarding in Solaris Secure Shell**

After authentication is complete, the user can use Solaris Secure Shell, generally by requesting a shell or executing a command. Through the ssh command options, the user can make requests. Requests can include allocating a pseudo-tty, forwarding X11 connections or TCP/IP connections, or enabling an ssh-agent authentication program over a secure connection.

The basic components of a user session are as follows:

1. The user requests a shell or the execution of a command, which begins the session mode.
   - In this mode, data is sent or received through the terminal on the client side. On the server side, data is sent through the shell or a command.

2. When data transfer is complete, the user program terminates.

3. All X11 forwarding and TCP/IP forwarding is stopped, except for those connections that already exist. Existing X11 connections and TCP/IP connections remain open.

4. The server sends an exit status message to the client. When all connections are closed, such as forwarded ports that had remained open, the client closes the connection to the server. Then, the client exits.
Client and Server Configuration in Solaris Secure Shell

The characteristics of a Solaris Secure Shell session are controlled by configuration files. The configuration files can be overridden to a certain degree by options on the command line.

Client Configuration in Solaris Secure Shell

In most cases, the client-side characteristics of a Solaris Secure Shell session are governed by the system-wide configuration file, /etc/ssh/ssh_config. The settings in the ssh_config file can be overridden by the user's configuration file, ~/.ssh/config. In addition, the user can override both configuration files on the command line.

The settings in the server's /etc/ssh/sshd_config file determine which client requests are permitted by the server. For a list of server configuration settings, see “Keywords in Solaris Secure Shell” on page 346. For detailed information, see the sshd_config(4) man page.

The keywords in the client configuration file are listed in “Keywords in Solaris Secure Shell” on page 346. If the keyword has a default value, the value is given. These keywords are described in detail in the ssh(1), scp(1), sftp(1), and ssh_config(4) man pages. For a list of keywords in alphabetical order and their equivalent command-line overrides, see Table 20–8.

Server Configuration in Solaris Secure Shell

The server-side characteristics of a Solaris Secure Shell session are governed by the /etc/ssh/sshd_config file. The keywords in the server configuration file are listed in “Keywords in Solaris Secure Shell” on page 346. If the keyword has a default value, the value is given. For a full description of the keywords, see the sshd_config(4) man page.

Keywords in Solaris Secure Shell

The following tables list the keywords and their default values, if any. The keywords are in alphabetical order. The location of keywords on the client is the ssh_config file. Keywords that apply to the server are in the sshd_config file. Some keywords are set in both files. If the keyword applies to only one protocol version, the version is listed.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Default Value</th>
<th>Location</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllowGroups</td>
<td>No default.</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>AllowTcpForwarding</td>
<td>yes</td>
<td>Server</td>
<td></td>
</tr>
</tbody>
</table>
### TABLE 20–1  Keywords in Solaris Secure Shell Configuration Files (A to Escape)  

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Default Value</th>
<th>Location</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllowUsers</td>
<td>No default.</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>AuthorizedKeysFile</td>
<td>~/.ssh/authorized_keys</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>Banner</td>
<td>/etc/issue</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>Batchmode</td>
<td>no</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>BindAddress</td>
<td>No default.</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>CheckHostIP</td>
<td>yes</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>Cipher</td>
<td>blowfish, 3des</td>
<td>Client</td>
<td>v1</td>
</tr>
<tr>
<td>Ciphers</td>
<td>aes128-ctr, aes128-cbc, 3des-cbc, blowfish-cbc, arcfour</td>
<td>Both</td>
<td>v2</td>
</tr>
<tr>
<td>ClearAllForwardings</td>
<td>No default.</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>ClientAliveInterval</td>
<td>0</td>
<td>Server</td>
<td>v2</td>
</tr>
<tr>
<td>ClientAliveCountMax</td>
<td>3</td>
<td>Server</td>
<td>v2</td>
</tr>
<tr>
<td>Compression</td>
<td>yes</td>
<td>Both</td>
<td></td>
</tr>
<tr>
<td>CompressionLevel</td>
<td>No default.</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>ConnectionAttempts</td>
<td>1</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>DenyGroups</td>
<td>No default.</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>DenyUsers</td>
<td>No default.</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>DynamicForward</td>
<td>No default.</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>EscapeChar</td>
<td>~</td>
<td>Client</td>
<td></td>
</tr>
</tbody>
</table>

### TABLE 20–2  Keywords in Solaris Secure Shell Configuration Files (Fall to Local)  

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Default Value</th>
<th>Location</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>FallBackToRsh</td>
<td>no</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>ForwardAgent</td>
<td>no</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>ForwardX11</td>
<td>no</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>GatewayPorts</td>
<td>no</td>
<td>Both</td>
<td></td>
</tr>
<tr>
<td>GlobalKnownHostsFile</td>
<td>/etc/ssh/ssh_known_hosts</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>GSSAPIAuthentication</td>
<td>yes</td>
<td>Both</td>
<td>v2</td>
</tr>
</tbody>
</table>
### TABLE 20–2  Keywords in Solaris Secure Shell Configuration Files (Fall to Local)  
(Continued)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Default Value</th>
<th>Location</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSSAPIDelegateCredentials</td>
<td>no</td>
<td>Client</td>
<td>v2</td>
</tr>
<tr>
<td>GSSAPIKeyExchange</td>
<td>yes</td>
<td>Both</td>
<td>v2</td>
</tr>
<tr>
<td>GSSAPIStoreDelegateCredentials</td>
<td>no</td>
<td>Client</td>
<td>v2</td>
</tr>
<tr>
<td>Host</td>
<td>* For more information, see &quot;Host-Specific Parameters in Solaris Secure Shell&quot; on page 350.</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>HostbasedAuthentication</td>
<td>no</td>
<td>Both</td>
<td>v2</td>
</tr>
<tr>
<td>HostbasedUsesNamesFromPacketOnly</td>
<td>no</td>
<td>Server</td>
<td>v2</td>
</tr>
<tr>
<td>HostKey</td>
<td>/etc/ssh/ssh_host_key</td>
<td>Server</td>
<td>v1</td>
</tr>
<tr>
<td>HostKey</td>
<td>/etc/ssh/host_rsa_key, /etc/ssh/host_dsa_key</td>
<td>Server</td>
<td>v2</td>
</tr>
<tr>
<td>HostKeyAlgorithms</td>
<td>ssh-rsa, ssh-dss</td>
<td>Client</td>
<td>v2</td>
</tr>
<tr>
<td>HostKeyAlias</td>
<td>No default.</td>
<td>Client</td>
<td>v2</td>
</tr>
<tr>
<td>IdentityFile</td>
<td>~/.ssh/identity</td>
<td>Client</td>
<td>v1</td>
</tr>
<tr>
<td>IdentityFile</td>
<td>~/.ssh/id_dsa, ~/.ssh/id_rsa</td>
<td>Client</td>
<td>v2</td>
</tr>
<tr>
<td>IgnoreRhosts</td>
<td>yes</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>IgnoreUserKnownHosts</td>
<td>yes</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>KbdInteractiveAuthentication</td>
<td>yes</td>
<td>Both</td>
<td></td>
</tr>
<tr>
<td>KeepAlive</td>
<td>yes</td>
<td>Both</td>
<td></td>
</tr>
<tr>
<td>KeyRegenerationInterval</td>
<td>3600 (seconds)</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>ListenAddress</td>
<td>No default.</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>LocalForward</td>
<td>No default.</td>
<td>Client</td>
<td></td>
</tr>
</tbody>
</table>

### TABLE 20–3  Keywords in Solaris Secure Shell Configuration Files (Login to R)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Default Value</th>
<th>Location</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoginGraceTime</td>
<td>600 (seconds)</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>LogLevel</td>
<td>info</td>
<td>Both</td>
<td></td>
</tr>
<tr>
<td>LookupClientHostname</td>
<td>yes</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>MACs</td>
<td>hmac-sha1,hmac-md5</td>
<td>Both</td>
<td>v2</td>
</tr>
</tbody>
</table>
### TABLE 20–3 Keywords in Solaris Secure Shell Configuration Files (Login to R) (Continued)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>DefaultValue</th>
<th>Location</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxAuthTries</td>
<td>6</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>MaxAuthTriesLog</td>
<td>3</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>MaxStartups</td>
<td>10:30:60</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>NoHostAuthenticationForLocalHost</td>
<td>no</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>NumberOfPasswordPrompts</td>
<td>3</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>PAMAuthenticationViaKBDInt</td>
<td>yes</td>
<td>Server</td>
<td>v2</td>
</tr>
<tr>
<td>PasswordAuthentication</td>
<td>yes</td>
<td>Both</td>
<td></td>
</tr>
<tr>
<td>PermitEmptyPasswords</td>
<td>no</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>PermitRootLogin</td>
<td>no</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>PermitUserEnvironment</td>
<td>no</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>PreferredAuthentications</td>
<td>gssapi-keyex, gssapi-with-mic, hostbased, publickey, keyboard-interactive, password</td>
<td>Client</td>
<td>v2</td>
</tr>
<tr>
<td>Port</td>
<td>22</td>
<td>Both</td>
<td></td>
</tr>
<tr>
<td>PrintMotd</td>
<td>no</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>Protocol</td>
<td>2</td>
<td>Both</td>
<td></td>
</tr>
<tr>
<td>ProxyCommand</td>
<td>No default.</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>PubkeyAuthentication</td>
<td>yes</td>
<td>Both</td>
<td>v2</td>
</tr>
<tr>
<td>RemoteForward</td>
<td>No default.</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>RhostsAuthentication</td>
<td>no</td>
<td>Both</td>
<td>v1</td>
</tr>
<tr>
<td>RhostsRSAAuthentication</td>
<td>no</td>
<td>Both</td>
<td>v1</td>
</tr>
<tr>
<td>RSAAuthentication</td>
<td>no</td>
<td>Both</td>
<td>v1</td>
</tr>
</tbody>
</table>

### TABLE 20–4 Keywords in Solaris Secure Shell Configuration Files (S to X)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>DefaultValue</th>
<th>Location</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServerKeyBits</td>
<td>768</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>StrictHostKeyChecking</td>
<td>ask</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>StrictModes</td>
<td>yes</td>
<td>Server</td>
<td></td>
</tr>
</tbody>
</table>
### Keywords in Solaris Secure Shell Configuration Files (Continued)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>DefaultValue</th>
<th>Location</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subsystem</td>
<td>sftp</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td></td>
<td>/usr/lib/ssh/sftp-server</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SyslogFacility</td>
<td>auth</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>UseLogin</td>
<td>no Deprecated and ignored.</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>UseOpenSSLEngine</td>
<td>yes</td>
<td>Both</td>
<td>v2</td>
</tr>
<tr>
<td>User</td>
<td>No default.</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>UserKnownHostsFile</td>
<td>~/.ssh/known_hosts</td>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>VerifyReverseMapping</td>
<td>no</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>X11Forwarding</td>
<td>yes</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>X11DisplayOffset</td>
<td>10</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>X11UseLocalHost</td>
<td>yes</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>XAuthLocation</td>
<td>/usr/openwin/bin/xauth</td>
<td>Both</td>
<td></td>
</tr>
</tbody>
</table>

### Host-Specific Parameters in Solaris Secure Shell

If it is useful to have different Solaris Secure Shell characteristics for different local hosts, the administrator can define separate sets of parameters in the `/etc/ssh/ssh_config` file to be applied according to host or regular expression. This task is done by grouping entries in the file by `Host` keyword. If the `Host` keyword is not used, the entries in the client configuration file apply to whichever local host a user is working on.

### Solaris Secure Shell and Login Environment Variables

When the following Solaris Secure Shell keywords are not set in the `sshd_config` file, they get their value from equivalent entries in the `/etc/default/login` file:

<table>
<thead>
<tr>
<th>Entry in <code>/etc/default/login</code></th>
<th>Keyword and Value in <code>sshd_config</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>CONSOLE=*</td>
<td>PermitRootLogin=without-password</td>
</tr>
<tr>
<td>#CONSOLE=*</td>
<td>PermitRootLogin=yes</td>
</tr>
<tr>
<td>PASSREQ=YES</td>
<td>PermitEmptyPasswords=no</td>
</tr>
<tr>
<td>PASSREQ=NO</td>
<td>PermitEmptyPasswords=yes</td>
</tr>
<tr>
<td>#PASSREQ</td>
<td>PermitEmptyPasswords=no</td>
</tr>
</tbody>
</table>
When the following variables are set by the initialization scripts from the user’s login shell, the sshd daemon uses those values. When the variables are not set, the daemon uses the default value.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIMEZONE</td>
<td>Controls the setting of the TZ environment variable.</td>
</tr>
<tr>
<td>ALTSHELL</td>
<td>Controls the setting of the SHELL environment variable.</td>
</tr>
<tr>
<td>PATH</td>
<td>Controls the setting of the PATH environment variable.</td>
</tr>
<tr>
<td>SUPATH</td>
<td>Controls the setting of the PATH environment variable for root.</td>
</tr>
</tbody>
</table>

For more information, see the `login(1)` and `sshd(1M)` man pages.

Maintaining Known Hosts in Solaris Secure Shell

Each host that needs to communicate securely with another host must have the server’s public key stored in the local host’s `/etc/ssh/ssh_known_hosts` file. Although a script could be used to update the `/etc/ssh/ssh_known_hosts` files, such a practice is heavily discouraged because a script opens a major security vulnerability.

The `/etc/ssh/ssh_known_hosts` file should only be distributed by a secure mechanism as follows:

- Over a secure connection, such as Solaris Secure Shell, IPsec, or Kerberized ftp from a known and trusted machine
- At system install time

To avoid the possibility of an intruder gaining access by inserting bogus public keys into a known_hosts file, you should use a JumpStart™ server as the known and trusted source of the ssh_known_hosts file. The ssh_known_hosts file can be distributed during installation. Later,
scripts that use the scp command can be used to pull in the latest version. This approach is secure because each host already has the public key from the JumpStart server.

Solaris Secure Shell Packages and Initialization

Solaris Secure Shell depends on core Solaris packages and the following packages:

- **SUNWgss** – Contains Generic Security Service (GSS) software
- **SUNWtcpd** – Contains TCP wrappers
- **SUNWopenssl-libraries** – Contains OpenSSL libraries
- **SUNWzlib** – Contains the zip compression library

The following packages install Solaris Secure Shell:

- **SUNWsshr** – Contains client files and utilities for the root (/) directory
- **SUNWsshdr** – Contains server files and utilities for the root (/) directory
- **SUNWsshua** – Contains common source files for the /usr directory
- **SUNWsshd** – Contains server files for the /usr directory
- **SUNWsshu** – Contains client files and utilities for the /usr directory

Upon reboot after installation, the sshd daemon is running. The daemon creates host keys on the system. A Solaris system that runs the sshd daemon is a Solaris Secure Shell server.

Solaris Secure Shell Files

The following table shows the important Solaris Secure Shell files and the suggested file permissions.

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
<th>Suggested Permissions and Owner</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/ssh/sshd_config</td>
<td>Contains configuration data for sshd, the Solaris Secure Shell daemon.</td>
<td>-rw-r--r-- root</td>
</tr>
<tr>
<td>/etc/ssh/ssh_host_key</td>
<td>Contains the host private key (v1).</td>
<td>-rw------- root</td>
</tr>
<tr>
<td>/etc/ssh/ssh_host_dsa_key or</td>
<td>Contains the host private key (v2).</td>
<td>-rw------- root</td>
</tr>
<tr>
<td>/etc/ssh/ssh_host_rsa_key</td>
<td></td>
<td></td>
</tr>
<tr>
<td>host-private-key.pub</td>
<td>Contains the host public key, for example, /etc/ssh/ssh_host_rsa_key.pub. Is used to copy the host key to the local known_hosts file.</td>
<td>-rw-r--r-- root</td>
</tr>
</tbody>
</table>
### Solaris Secure Shell Files

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
<th>Suggested Permissions and Owner</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/var/run/sshd.pid</code></td>
<td>Contains the process ID of the Solaris Secure Shell daemon, <code>sshd</code>. If multiple daemons are running, the file contains the last daemon that was started.</td>
<td><code>-rw-r--r-- root</code></td>
</tr>
<tr>
<td><code>~/.ssh/authorized_keys</code></td>
<td>Holds the public keys of the user who is allowed to log in to the user account.</td>
<td><code>-rw-r--r-- username</code></td>
</tr>
<tr>
<td><code>/etc/ssh/ssh_known_hosts</code></td>
<td>Contains the host public keys for all hosts with which the client can communicate securely. The file is populated by the administrator.</td>
<td><code>-rw-r--r-- root</code></td>
</tr>
<tr>
<td><code>~/.ssh/known_hosts</code></td>
<td>Contains the host public keys for all hosts with which the client can communicate securely. The file is maintained automatically. Whenever the user connects with an unknown host, the remote host key is added to the file.</td>
<td><code>-rw-r--r-- username</code></td>
</tr>
<tr>
<td><code>/etc/default/login</code></td>
<td>Provides defaults for the <code>sshd</code> daemon when corresponding <code>sshd_config</code> parameters are not set.</td>
<td><code>-r--r--r-- root</code></td>
</tr>
<tr>
<td><code>/etc/nologin</code></td>
<td>If this file exists, the <code>sshd</code> daemon only permits root to log in. The contents of this file are displayed to users who are attempting to log in.</td>
<td><code>-rw-r--r-- root</code></td>
</tr>
<tr>
<td><code>~/.rhosts</code></td>
<td>Contains the host-user name pairs that specify the hosts to which the user can log in without a password. This file is also used by the <code>rlogind</code> and <code>rshd</code> daemons.</td>
<td><code>-rw-r--r-- username</code></td>
</tr>
<tr>
<td><code>~/.shosts</code></td>
<td>Contains the host-user name pairs that specify the hosts to which the user can log in without a password. For more information, see the <code>sshd(1M)</code> man page in the FILES section.</td>
<td><code>-rw-r--r-- username</code></td>
</tr>
<tr>
<td><code>/etc/hosts.equiv</code></td>
<td>Contains the hosts that are used in <code>.rhosts</code> authentication. This file is also used by the <code>rlogind</code> and <code>rshd</code> daemons.</td>
<td><code>-rw-r--r-- root</code></td>
</tr>
<tr>
<td><code>/etc/ssh/hosts.equiv</code></td>
<td>Contains the hosts that are used in host-based authentication. This file is not used by other utilities.</td>
<td><code>-rw-r--r-- root</code></td>
</tr>
<tr>
<td><code>~/.ssh/environment</code></td>
<td>Contains initial assignments at login. By default, this file is not read. The <code>PermitUserEnvironment</code> keyword in the <code>sshd_config</code> file must be set to <code>yes</code> for this file to be read.</td>
<td><code>-rw-r--r-- username</code></td>
</tr>
<tr>
<td><code>~/.ssh/rc</code></td>
<td>Contains initialization routines that are run before the user shell starts. For a sample initialization routine, see the <code>sshd</code> man page.</td>
<td><code>-rw-r--r-- username</code></td>
</tr>
</tbody>
</table>
**Solaris Secure Shell Commands**

The following table lists the Solaris Secure Shell files that can be overridden by keywords or command options.

**TABLE 20–6**  Overrides for the Location of Solaris Secure Shell Files

<table>
<thead>
<tr>
<th>File Name</th>
<th>Keyword Override</th>
<th>Command-Line Override</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/etc/ssh/ssh_config</code></td>
<td>HostKey</td>
<td><code>ssh -F config-file</code></td>
</tr>
<tr>
<td><code>/etc/ssh/host_rsa_key</code></td>
<td></td>
<td><code>scp -F config-file</code></td>
</tr>
<tr>
<td><code>~/.ssh/identity</code></td>
<td>IdentityFile</td>
<td><code>ssh -i id-file</code></td>
</tr>
<tr>
<td><code>~/.ssh/id_dsa,~/.ssh/id_rsa</code></td>
<td></td>
<td><code>scp -i id-file</code></td>
</tr>
<tr>
<td><code>~/.ssh/authorized_keys</code></td>
<td>AuthorizedKeysFile</td>
<td></td>
</tr>
<tr>
<td><code>/etc/ssh/ssh_known_hosts</code></td>
<td>GlobalKnownHostsFile</td>
<td></td>
</tr>
<tr>
<td><code>~/.ssh/known_hosts</code></td>
<td>UserKnownHostsFile</td>
<td></td>
</tr>
</tbody>
</table>

**Solaris Secure Shell Commands**

The following table summarizes the major Solaris Secure Shell commands.

**TABLE 20–7**  Commands in Solaris Secure Shell

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Man Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssh</td>
<td>Logs a user in to a remote machine and securely executes commands on a remote machine. This command is the Solaris Secure Shell replacement for the <code>rlogin</code> and <code>rsh</code> commands. The <code>ssh</code> command enables secure encrypted communications between two untrusted hosts over an insecure network. X11 connections and arbitrary TCP/IP ports can also be forwarded over the secure channel.</td>
<td><code>ssh(1)</code></td>
</tr>
</tbody>
</table>
**TABLE 20–7  Commands in Solaris Secure Shell  (Continued)**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Man Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sshd</code></td>
<td>Is the daemon for Solaris Secure Shell. The daemon listens for connections from clients and enables secure encrypted communications between two untrusted hosts over an insecure network.</td>
<td><code>sshd(1M)</code></td>
</tr>
<tr>
<td><code>ssh-add</code></td>
<td>Adds RSA or DSA identities to the authentication agent, <code>ssh-agent</code>. Identities are also called <em>keys</em>.</td>
<td><code>ssh-add(1)</code></td>
</tr>
<tr>
<td><code>ssh-agent</code></td>
<td>Holds private keys that are used for public key authentication. The <code>ssh-agent</code> program is started at the beginning of an X-session or a login session. All other windows and other programs are started as clients of the <code>ssh-agent</code> program. Through the use of environment variables, the agent can be located and used for authentication when users use the <code>ssh</code> command to log in to other systems.</td>
<td><code>ssh-agent(1)</code></td>
</tr>
<tr>
<td><code>ssh-keygen</code></td>
<td>Generates and manages authentication keys for Solaris Secure Shell.</td>
<td><code>ssh-keygen(1)</code></td>
</tr>
<tr>
<td><code>ssh-keyscan</code></td>
<td>Gathers the public keys of a number of Solaris Secure Shell hosts. Aids in building and verifying <code>ssh_known_hosts</code> files.</td>
<td><code>ssh-keyscan(1)</code></td>
</tr>
<tr>
<td><code>ssh-keysign</code></td>
<td>Is used by the <code>ssh</code> command to access the host keys on the local host. Generates the digital signature that is required during host-based authentication with Solaris Secure Shell v2. The command is invoked by the <code>ssh</code> command, not by the user.</td>
<td><code>ssh-keysign(1M)</code></td>
</tr>
<tr>
<td><code>scp</code></td>
<td>Securely copies files between hosts on a network over an encrypted <code>ssh</code> transport. Unlike the <code>rcp</code> command, the <code>scp</code> command prompts for passwords or passphrases, if password information is needed for authentication.</td>
<td><code>scp(1)</code></td>
</tr>
<tr>
<td><code>sftp</code></td>
<td>Is an interactive file transfer program that is similar to the <code>ftp</code> command. Unlike the <code>ftp</code> command, the <code>sftp</code> command performs all operations over an encrypted <code>ssh</code> transport. The command connects, logs in to the specified host name, and then enters interactive command mode.</td>
<td><code>sftp(1)</code></td>
</tr>
</tbody>
</table>

The following table lists the command options that override Solaris Secure Shell keywords. The keywords are specified in the `ssh_config` and `sshd_config` files.

**TABLE 20–8  Command-Line Equivalents for Solaris Secure Shell Keywords**

<table>
<thead>
<tr>
<th>Keyword</th>
<th><code>ssh</code> Command-Line Override</th>
<th><code>scp</code> Command-Line Override</th>
</tr>
</thead>
<tbody>
<tr>
<td>BatchMode</td>
<td><code>ssh -B</code></td>
<td><code>scp -B</code></td>
</tr>
<tr>
<td>BindAddress</td>
<td><code>ssh -b bind-addr</code></td>
<td><code>scp -a bind-addr</code></td>
</tr>
<tr>
<td>Cipher</td>
<td><code>ssh -c cipher</code></td>
<td><code>scp -c cipher</code></td>
</tr>
<tr>
<td>Ciphers</td>
<td><code>ssh -c cipher-spec</code></td>
<td><code>scp -c cipher-spec</code></td>
</tr>
<tr>
<td>Compression</td>
<td><code>ssh -C</code></td>
<td><code>scp -C</code></td>
</tr>
<tr>
<td>DynamicForward</td>
<td><code>ssh -D SOCKS4-port</code></td>
<td></td>
</tr>
</tbody>
</table>
## Solaris Secure Shell Commands

**TABLE 20–8 Command-Line Equivalents for Solaris Secure Shell Keywords**  
(Continued)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>ssh Command-Line Override</th>
<th>scp Command-Line Override</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EscapeChar</strong></td>
<td>ssh -e escape-char</td>
<td></td>
</tr>
<tr>
<td><strong>ForwardAgent</strong></td>
<td>ssh -A to enable</td>
<td>ssh -a to disable</td>
</tr>
<tr>
<td><strong>ForwardX11</strong></td>
<td>ssh -X to enable</td>
<td>ssh -x to disable</td>
</tr>
<tr>
<td><strong>GatewayPorts</strong></td>
<td>ssh -g</td>
<td></td>
</tr>
<tr>
<td><strong>IPv4</strong></td>
<td>ssh -4</td>
<td>scp -4</td>
</tr>
<tr>
<td><strong>IPv6</strong></td>
<td>ssh -6</td>
<td>scp -6</td>
</tr>
<tr>
<td><strong>LocalForward</strong></td>
<td>ssh -L localport:remotehost:remoteport</td>
<td></td>
</tr>
<tr>
<td><strong>MACS</strong></td>
<td>ssh -m mac-spec</td>
<td></td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>ssh -p port</td>
<td>scp -P port</td>
</tr>
<tr>
<td><strong>Protocol</strong></td>
<td>ssh -1 for v1 only</td>
<td>ssh -2 for v2 only</td>
</tr>
<tr>
<td><strong>RemoteForward</strong></td>
<td>ssh -R remoteport:localhost:localport</td>
<td></td>
</tr>
</tbody>
</table>
This section provides information on the configuration, management and use of the Kerberos service.
Introduction to the Kerberos Service

This chapter introduces the Kerberos Service. The following is a list of the overview information in this chapter.

- “What Is the Kerberos Service?” on page 359
- “How the Kerberos Service Works” on page 360
- “Kerberos Security Services” on page 367
- “The Components of Various Kerberos Releases” on page 367

What Is the Kerberos Service?

The Kerberos service is a client-server architecture that provides secure transactions over networks. The service offers strong user authentication, as well as integrity and privacy. Authentication guarantees that the identities of both the sender and the recipient of a network transaction are true. The service can also verify the validity of data being passed back and forth (integrity) and encrypt the data during transmission (privacy). Using the Kerberos service, you can log in to other machines, execute commands, exchange data, and transfer files securely. Additionally, the service provides authorization services, which allows administrators to restrict access to services and machines. Moreover, as a Kerberos user, you can regulate other people’s access to your account.

The Kerberos service is a single-sign-on system, which means that you only need to authenticate yourself to the service once per session, and all subsequent transactions during the session are automatically secured. After the service has authenticated you, you do not need to authenticate yourself every time you use a Kerberos-based command such as ftp or rsh, or to access data on an NFS file system. Thus, you do not have to send your password over the network, where it can be intercepted, each time you use these services.

The Solaris Kerberos service is based on the Kerberos V5 network authentication protocol that was developed at the Massachusetts Institute of Technology (MIT). People who have used Kerberos V5 product should therefore find the Solaris version very familiar. Because the Kerberos V5 protocol is a de facto industry standard for network security, the Solaris version...
promotes interoperability with other systems. In other words, because the Solaris Kerberos service works with systems that use the Kerberos V5 protocol, the service allows for secure transactions even over heterogeneous networks. Moreover, the service provides authentication and security both between domains and within a single domain.

The Kerberos service allows for flexibility in running Solaris applications. You can configure the service to allow both Kerberos-based and non-Kerberos-based requests for network services such as the NFS service, telnet, and ftp. As a result, current Solaris applications still work even if they are running on systems on which the Kerberos service is not enabled. Of course, you can also configure the Kerberos service to allow only Kerberos-based network requests.

The Kerberos service provides a security mechanism which allows the use of Kerberos for authentication, integrity, and privacy when using applications that use the Generic Security Service Application Programming Interface (GSS-API). However, applications do not have to remain committed to the Kerberos service if other security mechanisms are developed. Because the service is designed to integrate modularly into the GSS-API, applications that use the GSS-API can utilize whichever security mechanism best suits their needs.

How the Kerberos Service Works

The following is an overview of the Kerberos authentication system. For a more detailed description, see “How the Kerberos Authentication System Works” on page 542.

From the user’s standpoint, the Kerberos service is mostly invisible after the Kerberos session has been started. Commands such as rsh or ftp work about the same. Initializing a Kerberos session often involves no more than logging in and providing a Kerberos password.

The Kerberos system revolves around the concept of a ticket. A ticket is a set of electronic information that identifies a user or a service such as the NFS service. Just as your driver’s license identifies you and indicates what driving privileges you have, so a ticket identifies you and your network access privileges. When you perform a Kerberos-based transaction (for example, if you remote log in to another machine), you transparently send a request for a ticket to a Key Distribution Center, or KDC. The KDC accesses a database to authenticate your identity and returns a ticket that grants you permission to access the other machine. “Transparency” means that you do not need to explicitly request a ticket. The request happens as part of the rlogin command. Because only an authenticated client can get a ticket for a specific service, another client cannot use rlogin under an assumed identity.

Tickets have certain attributes associated with them. For example, a ticket can be forwardable, which means that it can be used on another machine without a new authentication process. A ticket can also be postdated, which means that it is not valid until a specified time. How tickets can be used, for example, to specify which users are allowed to obtain which types of ticket, is set by policies. Policies are determined when the Kerberos service is installed or administered.
Note – You will frequently see the terms credential and ticket. In the greater Kerberos world, they are often used interchangeably. Technically, however, a credential is a ticket plus the session key for that session. This difference is explained in more detail in “Gaining Access to a Service Using Kerberos” on page 542.

The following sections further explain the Kerberos authentication process.

**Initial Authentication: the Ticket-Granting Ticket**

Kerberos authentication has two phases: an initial authentication that allows for all subsequent authentications, and the subsequent authentications themselves.

The following figure shows how the initial authentication takes place.

1. At login (or with `kinit`), Client requests a TGT that allows it to obtain tickets for services.
   2. KDC checks database, sends TGT.
   3. Client uses password to decrypt TGT, thus proving identity; can now use the TGT to obtain other tickets.

TGT = Ticket-granting ticket  
KDC = Key Distribution Center

**FIGURE 21–1** Initial Authentication for a Kerberos Session

1. A client (a user, or a service such as NFS) begins a Kerberos session by requesting a *ticket-granting ticket* (TGT) from the Key Distribution Center (KDC). This request is often done automatically at login.
A ticket-granting ticket is needed to obtain other tickets for specific services. Think of the ticket-granting ticket as similar to a passport. Like a passport, the ticket-granting ticket identifies you and allows you to obtain numerous “visas,” where the “visas” (tickets) are not for foreign countries but for remote machines or network services. Like passports and visas, the ticket-granting ticket and the other various tickets have limited lifetimes. The difference is that “Kerberized” commands notice that you have a passport and obtain the visas for you. You don’t have to perform the transactions yourself.

Another analogy for the ticket-granting ticket is that of a three-day ski pass that is good at four different ski resorts. You show the pass at whichever resort you decide to go to and you receive a lift ticket for that resort, as long as the pass has not expired. Once you have the lift ticket, you can ski all you want at that resort. If you go to another resort the next day, you once again show your pass, and you get an additional lift ticket for the new resort. The difference is that the Kerberos-based commands notice that you have the weekend ski pass, and they get the lift ticket for you. So you don’t have to perform the transactions yourself.

2. The KDC creates a ticket–granting ticket and sends it back, in encrypted form, to the client. The client decrypts the ticket-granting ticket by using the client’s password.

3. Now in possession of a valid ticket-granting ticket, the client can request tickets for all sorts of network operations, such as `rlogin` or `telnet`, for as long as the ticket-granting ticket lasts. This ticket usually lasts for a few hours. Each time the client performs a unique network operation, it requests a ticket for that operation from the KDC.

**Subsequent Kerberos Authentications**

After the client has received the initial authentication, each subsequent authentication follows the pattern that is shown in the following figure.
1. The client requests a ticket for a particular service, for example, remote login to another machine, from the KDC by sending the KDC its ticket-granting ticket as proof of identity.

2. The KDC sends the ticket for the specific service to the client.

   For example, suppose user joe wants to access an NFS file system that has been shared with krb5 authentication required. Because he is already authenticated (that is, he already has a ticket-granting ticket), as he attempts to access the files, the NFS client system automatically and transparently obtains a ticket from the KDC for the NFS service.

   For example, suppose the user joe uses rlogin on the server boston. Because he is already authenticated, that is, he already has a ticket-granting ticket, he automatically and transparently obtains a ticket as part of the rlogin command. This ticket allows him to remote log in to boston as often as he wants until the ticket expires. If joe wants to remote log in to the machine denver, he obtains another ticket, as in Step 1.

3. The client sends the ticket to the server.

   When using the NFS service, the NFS client automatically and transparently sends the ticket for the NFS service to the NFS server.

TGT = Ticket-granting ticket
KDC = Key Distribution Center
FIGURE 21-2 Obtaining Access to a Service Using Kerberos Authentication
4. The server allows the client access.

These steps make it appear that the server doesn’t ever communicate with the KDC. The server does, though; it registers itself with the KDC, just as the first client does. For simplicity’s sake, that part has been left out.

The Kerberos Remote Applications

The Kerberos-based (or “Kerberized”) commands that a user such as joe can use are the following:

- ftp
- rcp
- rdist
- rlogin
- rsh
- ssh
- telnet

These applications are the same as the Solaris applications of the same name. However, they have been extended to use Kerberos principals to authenticate transactions, thereby providing Kerberos-based security. See "Kerberos Principals" on page 364 for information on principals.

These commands are discussed further in "Kerberos User Commands" on page 526.

Kerberos Principals

A client in the Kerberos service is identified by its principal. A principal is a unique identity to which the KDC can assign tickets. A principal can be a user, such as joe, or a service, such as nfs or telnet.
By convention, a principal name is divided into three components: the primary, the instance, and the realm. A typical Kerberos principal would be, for example, joe/admin@ENG.EXAMPLE.COM. In this example:

- joe is the primary. The primary can be a user name, as shown here, or a service, such as nfs. The primary can also be the word host, which signifies that this principal is a service principal that is set up to provide various network services, ftp, rcp, rlogin, and so on.

- admin is the instance. An instance is optional in the case of user principals, but it is required for service principals. For example, if the user joe sometimes acts as a system administrator, he can use joe/admin to distinguish himself from his usual user identity. Likewise, if joe has accounts on two different hosts, he can use two principal names with different instances, for example, joe/denver.example.com and joe/boston.example.com. Notice that the Kerberos service treats joe and joe/admin as two completely different principals.

In the case of a service principal, the instance is the fully qualified host name. bigmachine.eng.example.com is an example of such an instance. The primary/instance for this example might be ftp/bigmachine.eng.example.com or host/bigmachine.eng.example.com.

- ENG.EXAMPLE.COM is the Kerberos realm. Realms are discussed in “Kerberos Realms” on page 365.

The following are all valid principal names:

- joe
- joe/admin
- joe/admin@ENG.EXAMPLE.COM
- nfs/host.eng.example.com@ENG.EXAMPLE.COM
- host/eng.example.com@ENG.EXAMPLE.COM

**Kerberos Realms**

A realm is a logical network, similar to a domain, that defines a group of systems under the same master KDC. Figure 21–3 shows how realms can relate to one another. Some realms are hierarchical, where one realm is a superset of the other realm. Otherwise, the realms are nonhierarchical (or "direct") and the mapping between the two realms must be defined. A feature of the Kerberos service is that it permits authentication across realms. Each realm only needs to have a principal entry for the other realm in its KDC. This Kerberos feature is called cross-realm authentication.
Kerberos Servers

Each realm must include a server that maintains the master copy of the principal database. This server is called the master KDC server. Additionally, each realm should contain at least one slave KDC server, which contains duplicate copies of the principal database. Both the master KDC server and the slave KDC server create tickets that are used to establish authentication.

The realm can also include a Kerberos application server. This server provides access to Kerberized services (such as ftp, telnet, rsh and NFS). If you have installed SEAM 1.0 or 1.0.1, the realm might include a Kerberos network application server, but this software was not included with these releases.

The following figure shows what a hypothetical realm might contain.
Kerberos Security Services

In addition to providing secure authentication of users, the Kerberos service provides two security services:

- **Integrity** – Just as authentication ensures that clients on a network are who they claim to be, integrity ensures that the data they send is valid and has not been tampered with during transit. Integrity is done through cryptographic checksumming of the data. Integrity also includes user authentication.

- **Privacy** – Privacy takes security a step further. Privacy not only includes verifying the integrity of transmitted data, but it encrypts the data before transmission, protecting it from eavesdroppers. Privacy authenticates users, as well.

Developers can design their RPC-based applications to choose a security service by using the RPCSEC_GSS programming interface.

The Components of Various Kerberos Releases

Components of the Kerberos service have been included in many releases. Originally, the Kerberos service and changes to the base operating system to support the Kerberos service were released using the product name “Sun Enterprise Authentication Mechanism” which was shortened to SEAM. As more parts of the SEAM product were included in the Solaris software, the contents of the SEAM release decreased. For the Solaris 10 release, all parts of the SEAM product are included, so there is no longer a need for the SEAM product. The SEAM product name exists in the documentation for historical reasons.

The following table describes which components are included in each release. Each product release is listed in chronological order. All components are described in the following sections.
TABLE 21–1  Kerberos Release Contents

<table>
<thead>
<tr>
<th>Release Name</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEAM 1.0 in Solaris Easy Access Server 3.0</td>
<td>Full release of the Kerberos service for the Solaris 2.6 and 7 releases</td>
</tr>
<tr>
<td>The Kerberos service in the Solaris 8 release</td>
<td>Kerberos client software only</td>
</tr>
<tr>
<td>SEAM 1.0.1 in the Solaris 8 Admin Pack</td>
<td>Kerberos KDC and remote applications for the Solaris 8 release</td>
</tr>
<tr>
<td>The Kerberos service in the Solaris 9 release</td>
<td>Kerberos KDC and client software only</td>
</tr>
<tr>
<td>SEAM 1.0.2</td>
<td>Kerberos remote applications for the Solaris 9 release</td>
</tr>
<tr>
<td>The Kerberos service in the Solaris 10 release</td>
<td>Full release of the Kerberos service with enhancements</td>
</tr>
</tbody>
</table>

**Kerberos Components**

Similar to the MIT distribution of the Kerberos V5 product, the Solaris Kerberos service includes the following:

- **Key Distribution Center (KDC):**
  - Kerberos database administration daemon – kadmin.
  - Kerberos ticket processing daemon – krb5kdc.
  - Database administration programs – kadmin (master only), kadmin.local and kdb5_util.
  - Database propagation software – kprop (slave only) and kpropd.
- User programs for managing credentials – kinit, klist, and kdestroy.
- User program for changing your Kerberos password – kpasswd.
- Remote applications – ftp, rcp, rdist, rlogin, rsh, ssh, and telnet.
- Remote application daemons – ftpd, rlogind, rshd, sshd, and telnetd.
- Keytab administration utility – ktutil.
- The Generic Security Service Application Programming Interface (GSS-API) – Enables applications to use multiple security mechanisms without requiring you to recompile the application every time a new mechanism is added. The GSS-API uses standard interfaces that allow applications to be portable to many operating systems. GSS-API provides applications with the ability to include the integrity and privacy security services, as well as authentication. Both ftp and ssh use the GSS-API.
The RPCSEC_GSS Application Programming Interface (API) – Enables NFS services to use Kerberos authentication. RPCSEC_GSS is a security flavor that provides security services that are independent of the mechanisms being used. RPCSEC_GSS sits on top of the GSS-API layer. Any pluggable GSS_API-based security mechanism can be used by applications that use RPCSEC_GSS.

In addition, the Solaris Kerberos service includes the following:

- Graphical Kerberos Administration Tool (gkadmn) – Enables you to administer the principals and principal policies. This Java™ technology-based GUI is an alternative to the kadm5 command.
- A Kerberos V5 service module for PAM – Provides authentication, account management, session management and password management for the Kerberos service. The module can be used to make Kerberos authentication transparent to the user.
- Kernel modules – Provides kernel-based implementations of the kerberos service for use by the NFS service, which greatly improves performance.

### Kerberos Additions for the Solaris Express Community Edition Release

In build 90, the kclient script was enhanced. The script includes the feature of joining Microsoft Active Directory servers. See “How to Interactively Configure a Kerberos Client” on page 422 and “How to Configure a Kerberos Client for an Active Directory Server” on page 426 for instructions. In addition, the script includes a -T option that may be used to identify the KDC server type for the client. All of the options for this script are covered in the kclient(1M) man page.

In build 102, the /etc/krb5/kadm5.keytab file is no longer needed. The keys that were stored in this file are now directly read from the Kerberos database.
Kerberos Additions for the Solaris Express Developer Edition 1/08 Release

These enhancements are available starting in the Solaris Express Developer Edition 1/08 release:

- The Solaris Kerberos software has been synchronized with the MIT 1.4 version. In particular, the software for the KDC, the `kinit` command and the Kerberos mechanism have been updated.

- Support for accessing Kerberos principal and policy records using LDAP from a directory server has been added. This change simplifies administration and can provide greater availability, depending on the deployment of the KDCs and the DSs. See "Managing a KDC on an LDAP Directory Server" on page 457 for a list of LDAP-related procedures.

- The new `kdcmgr` command which can be used to automatically or interactively setup any KDC. This command works to create both master and slave KDC servers. Also, used with the `status` option, the `kdcmgr` command shows information about any KDC installed on the localhost. Look for the pointers to the automatic or interactive procedures in Table 23–1.

- Support for Solaris clients that require no additional setup has been added to this release. Changes were made to the Kerberos service and to some default settings. Solaris Kerberos clients work with no client-side configuration in environments that are appropriately configured. See "Client Configuration Options" on page 382 for more information.

Kerberos Additions for the Solaris 10 8/07 Release

The MIT Kerberos V5 application programming interface (krb5-api) is supported in the Solaris 10 8/07 release. See the `libkrb5(3LIB)` and `krb5-config(1)` man pages for more information. Also, see the MIT Kerberos V5 project web pages at mit.edu for more detailed documentation as it becomes available.

Although the krb5-api is now available, Sun strongly encourages the use of the GSS-API for network authentication and integrity and privacy as the GSS-API is security-mechanism independent and an IETF standard. See the `libgss(3LIB)` man page for more information.

Kerberos Additions for the Solaris 10 6/06 Release

In the Solaris 10 6/06 release, the `ktkt_warnd` daemon can automatically renew credentials, rather than just warn the user when the credential is about to expire. The user must be logged in for the credential to be renewed automatically.
Kerberos Enhancements in the Solaris 10 3/05 Release

These Kerberos enhancements are included in the Solaris 10 Release. Several of the enhancements were introduced in prior Software Express releases and updated in the Solaris 10 Beta releases.

- Kerberos protocol support is provided in remote applications, such as `ftp`, `rcp`, `rdist`, `rlogin`, `rsh`, `ssh`, and `telnet`. See the man pages for each command or daemon and the `krb5_auth_rules(5)` man page for more information.

- The Kerberos principal database can now be transferred by incremental update instead of by transferring the entire database each time. Incremental propagation provides these advantages:
  - Increased database consistencies across servers
  - The need for fewer resources (network, CPU, and so forth)
  - Much more timely propagation of updates
  - An automated method of propagation

- A new script to help automatically configure a Kerberos client is now available. The script helps an administrator quickly and easily set up a Kerberos client. For procedures using the new script, see "Configuring Kerberos Clients" on page 419. Also, see the `kclient(1M)` man page for more information.

- Several new encryption types have been added to the Kerberos service. These new encryption types increase security and enhance compatibility with other Kerberos implementations that support these encryption types. See "Using Kerberos Encryption Types" on page 545 for more information. The encryption types include:
  - The AES encryption type can be used for high speed, high security encryption of Kerberos sessions.
  - ARCFOUR-HMAC provides better compatibility with other Kerberos implementations.
  - Triple DES (3DES) with SHA1 increases security. This encryption type also enhances interoperability with other Kerberos implementations that support this encryption type.

- The encryption types are enabled through the Cryptographic Framework. The framework can provide for hardware accelerated cryptography for the Kerberos service.

- The KDC software, the user commands, and user applications now support the use of the TCP network protocol. This enhancement provides more robust operation and better interoperability with other Kerberos implementations, including Microsoft’s Active Directory. The KDC now listens on both the traditional UDP ports as well as TCP ports so it can respond to requests using either protocol. The user commands and applications first try UDP when sending a request to the KDC, and if that fails, then try TCP.

- Support for IPv6 was added to the KDC software, which includes the `kinit`, `klist` and `kprop` commands. Support for IPv6 addresses is provided by default. There are no configuration parameters to change to enable IPv6 support. No IPv6 support is available for the `kadmin` and `kadmind` commands.
- A new `-e` option has been included to several subcommands of the `kadmin` command. This new option allows for the selection of the encryption type during the creation of principals. See the `kadmin(1M)` man page for more information.

- Additions to the `pam_krb5` module to manage the Kerberos credentials cache by using the PAM framework. See the `pam_krb5(5)` man page for more information.

- Support is provided for auto-discovery of the Kerberos KDC, admin server, `kpasswd` server, and host or domain name-to-realm mappings by using DNS lookups. This enhancement reduces some of the steps needed to install a Kerberos client. The client is able to locate a KDC server by using DNS instead of by reading a configuration file. See the `krb5.conf(4)` man page for more information.

- A new PAM module called `pam_krb5_migrate` has been introduced. The new module helps in the automatic migration of users to the local Kerberos realm, if they do not already have Kerberos accounts. See the `pam_krb5_migrate(5)` man page for more information.

- The `~/.k5login` file can now be used with the GSS applications `ftp` and `ssh`. For more information, see the `gss_auth_rules(5)` man page.

- The `kproplog` utility has been updated to output all attribute names per log entry. For more information, see the `kproplog(1M)` man page.

- Strict TGT verification can now be disabled using a configuration option in the `krb5.conf` file. See the `krb5.conf(4)` man page for more information.

- Extensions to the password-changing utilities enable the Solaris Kerberos V5 administration server to accept password change requests from clients that do not run Solaris software. See the `kadmin(1M)` man page for more information.

- The default location of the replay cache has been moved from RAM-based file systems to persistent storage in `/var/krb5/rcache`. The new location protects against replays if a system is rebooted. Performance enhancements were made to the rcache code. However, overall replay cache performance might be slower due to the use of persistent storage.

- The replay cache can now be configured to use file or memory only storage. Refer to the `krb5envvar(5)` man page for more information about environment variables that can be configured for key table and credential cache types or locations.

- The GSS credential table is no longer necessary for the Kerberos GSS mechanism. For more information, see “Mapping GSS Credentials to UNIX Credentials” on page 381 or the `gsscred(1M)`, `gsdif(1M)`, and `gsscred.conf(4)` man pages.

- The Kerberos utilities, `kinit` and `ktutil`, are now based on MIT Kerberos version 1.2.1. This change added new options to the `kinit` command and new subcommands to the `ktutil` command. For more information, see the `kinit(1)` and `ktutil(1)` man pages.

- The Solaris Kerberos Key Distribution Center (KDC) and `kadmin` is now based on MIT Kerberos version 1.2.1. The KDC now defaults to a btree-based database, which is more reliable than the current hash-based database. See the `kdb5_util(1M)` man page for more information.
The kpropd, kadmnd, krb5kdc and ktkt_warnd daemons are managed by the Service Management Facility. Administrative actions on this service, such as enabling, disabling, or restarting, can be performed using the svcadm command. The service's status for all daemons can be queried using the svcs command. For an overview of the Service Management Facility refer to Chapter 16, "Managing Services (Overview)," in System Administration Guide: Basic Administration.

Kerberos Components in the Solaris 9 Release

The Solaris 9 release includes all components included in “Kerberos Components” on page 368, except for the remote applications.

SEAM 1.0.2 Components

The SEAM 1.0.2 release includes the remote applications. These applications are the only part of SEAM 1.0 that have not been incorporated into the Solaris 9 release. The components for the remote applications are as follows:

- Client applications – ftp, rcp, rlogin, rsh, and telnet
- Server daemons – ftpd, rlogind, rshd, and telnetd

Kerberos Components in the Solaris 8 Release

The Solaris 8 release includes only the client-side portions of the Kerberos service, so many components are not included. This product enables systems that run the Solaris 8 release to become Kerberos clients without requiring you to install SEAM 1.0.1 separately. To use these capabilities, you must install a KDC that uses either Solaris Easy Access Server 3.0 or the Solaris 8 Admin Pack, the MIT distribution, or Windows 2000. The client-side components are not useful without a configured KDC to distribute tickets. The following components are included in this release:

- User programs for obtaining, viewing, and destroying tickets – kinit, klist, and kdestroy.
- User program for changing your Kerberos password – kpasswd.
- Key table administration utility – ktutil.
- Additions to the Pluggable Authentication Module (PAM) – Enables applications to use various authentication mechanisms. PAM can be used to make logins and logouts transparent to the user.
- GSS_API plug-ins – Provides Kerberos protocol and cryptographic support.
- NFS client and server support.
The SEAM 1.0.1 release includes all components of the SEAM 1.0 release that are not already included in the Solaris 8 release. The components are as follows:

- Key Distribution Center (KDC) (master):
  - Kerberos database administration daemon – kadmin
  - Kerberos ticket processing daemon – krb5kdc
- Slave KDCs.
- Database administration programs – kadmin and kadmin.local.
- Database propagation software – kprop.
- Remote applications – ftp, rcp, rlogin, rsh, and telnet.
- Remote application daemons – ftpd, rlogind, rshd, and telnetd.
- Administration utility – kdb5_util.
- Graphical Kerberos Administration Tool (gkadmin) – Enables you to administer principals and principal policies. This Java technology-based GUI is an alternative to the kadmin command.
- A preconfiguration procedure – Enables you to set the parameters for installing and configuring SEAM 1.0.1, which makes SEAM installation automatic. This procedure is especially useful for multiple installations.
- Several libraries.

The SEAM 1.0 release includes all of the items included in “Kerberos Components” on page 368 as well as the following:

- A utility (gsscred) and a daemon (gssd) – These programs help map UNIX user IDs (UIDs) to principal names. These programs are needed because NFS servers use UNIX UIDs to identify users and not principal names, which are stored in a different format.
- The Generic Security Service Application Programming Interface (GSS-API) – Enables applications to use multiple security mechanisms without requiring you to recompile the application every time a new mechanism is added. Because GSS-API is machine-independent, it is appropriate for applications on the Internet. GSS-API provides applications with the ability to include the integrity and privacy security services, as well as authentication.
- The RPCSEC_GSS Application Programming Interface (API) – Enables NFS services to use Kerberos authentication. RPCSEC_GSS is a security flavor that provides security services that are independent of the mechanisms being used. RPCSEC_GSS sits on top of the GSS-API layer. Any pluggable GSS_API-based security mechanism can be used by applications that use RPCSEC_GSS.
- A preconfiguration procedure – Enables you to set the parameters for installing and configuring SEAM 1.0, which makes installation automatic. This procedure is especially useful for multiple installations.
Planning for the Kerberos Service

This chapter should be studied by administrators who are involved in the installation and maintenance of the Kerberos service. The chapter discusses several installation and configuration options that administrators must resolve before they install or configure the service.

This is a list of the topics that a system administrator or other knowledgeable support staff should study:

- “Why Plan for Kerberos Deployments?” on page 377
- “Planning Kerberos Realms” on page 378
- “Mapping Host Names Onto Realms” on page 379
- “Client and Service Principal Names” on page 379
- “Ports for the KDC and Admin Services” on page 380
- “The Number of Slave KDCs” on page 380
- “Which Database Propagation System to Use” on page 382
- “Clock Synchronization Within a Realm” on page 382
- “Client Configuration Options” on page 382
- “Improving Client Login Security” on page 383
- “KDC Configuration Options” on page 383
- “Kerberos Encryption Types” on page 384
- “Online Help URL in the Graphical Kerberos Administration Tool” on page 385

Why Plan for Kerberos Deployments?

Before you install the Kerberos service, you must resolve several configuration issues. Although changing the configuration after the initial install is not impossible, some changes can be difficult to implement. In addition, some changes require that the KDC be rebuilt, so it is better to consider long-term goals when you plan your Kerberos configuration.
Deploying a Kerberos infrastructure involves such tasks as installing KDCs, creating keys for your hosts, and migrating users. Reconfiguring a Kerberos deployment can be as hard as performing an initial deployment, so plan a deployment carefully to avoid having to re-configure.

Planning Kerberos Realms

A realm is logical network, similar to a domain, that defines a group of systems that are under the same master KDC. As with establishing a DNS domain name, issues such as the realm name, the number and size of each realm, and the relationship of a realm to other realms for cross-realm authentication should be resolved before you configure the Kerberos service.

Realm Names

Realm names can consist of any ASCII string. Usually, the realm name is the same as your DNS domain name, except that the realm name is in uppercase. This convention helps differentiate problems with the Kerberos service from problems with the DNS namespace, while using a name that is familiar. If you do not use DNS or you choose to use a different string, then you can use any string. However, the configuration process requires more work. The use of realm names that follow the standard Internet naming structure is wise.

Number of Realms

The number of realms that your installation requires depends on several factors:

- The number of clients to be supported. Too many clients in one realm makes administration more difficult and eventually requires that you split the realm. The primary factors that determine the number of clients that can be supported are as follows:
  - The amount of Kerberos traffic that each client generates
  - The bandwidth of the physical network
  - The speed of the hosts

Because each installation will have different limitations, no rule exists for determining the maximum number of clients.

- How far apart the clients are. Setting up several small realms might make sense if the clients are in different geographic regions.

- The number of hosts that are available to be installed as KDCs. Each realm should have at least two KDC servers, one master server and one slave server.

Alignment of Kerberos realms with administrative domains is recommended. Note that a Kerberos V realm can span multiple sub-domains of the DNS domain to which the realm corresponds.
Realm Hierarchy

When you are configuring multiple realms for cross-realm authentication, you need to decide how to tie the realms together. You can establish a hierarchical relationship among the realms, which provides automatic paths to the related domains. Of course, all realms in the hierarchical chain must be configured properly. The automatic paths can ease the administration burden. However, if there are many levels of domains, you might not want to use the default path because it requires too many transactions.

You can also choose to establish the trust relationship directly. A direct trust relationship is most useful when too many levels exist between two hierarchical realms or when no hierarchical relationship exists. The connection must be defined in the /etc/krb5/krb5.conf file on all hosts that use the connection. So, some additional work is required. The direct trust relationship is also referred to as a transitive relationship. For an introduction, see “Kerberos Realms” on page 365. For the configuration procedures for multiple realms, see “Configuring Cross-Realm Authentication” on page 409.

Mapping Host Names Onto Realms

The mapping of host names onto realm names is defined in the domain_realm section of the krb5.conf file. These mappings can be defined for a whole domain and for individual hosts, depending on the requirements.

DNS can also be used to look up information about the KDCs. Using DNS makes it easier to change the information because you will not need to edit the krb5.conf file on all of the clients each time you make a change. See the krb5.conf(4) man page for more information.

As of the Solaris Express Developer Edition 1/08 and the Solaris 10 5/08 releases, Solaris Kerberos clients can interoperate better with Active Directory servers. The Active Directory servers can be configured to provide the realm to host mapping.

Client and Service Principal Names

When you are using the Kerberos service, DNS must be enabled on all hosts. With DNS, the principal should contain the Fully Qualified Domain Name (FQDN) of each host. For example, if the host name is boston, the DNS domain name is example.com, and the realm name is EXAMPLE.COM, then the principal name for the host should be host/boston.example.com@EXAMPLE.COM. The examples in this book require that DNS is configured and use the FQDN for each host.

The Kerberos service canonicalizes host alias names through DNS, and uses the canonicalized form (cname) when constructing the service principal for the associated service. Therefore when creating a service principal, the host name component of service principal names should be the canonical form of the host name of the system hosting the service.
The following is an example of how the Kerberos service canonicalizes host name. If a user runs the command "ssh alpha.example.com" where alpha.example.com is a DNS host alias for the cname beta.example.com. When ssh calls Kerberos and requests a host service ticket for alpha.example.com, the Kerberos service canonicalizes alpha.example.com to beta.example.com and requests a ticket for the service principal "host/beta.example.com" from the KDC.

For the principal names that include the FQDN of a host, it is important to match the string that describes the DNS domain name in the /etc/resolv.conf file. The Kerberos service requires that the DNS domain name be in lowercase letters when you are specifying the FQDN for a principal. The DNS domain name can include uppercase and lowercase letters, but only use lowercase letters when you are creating a host principal. For example, it doesn't matter if the DNS domain name is example.com, Example.COM, or any other variation. The principal name for the host would still be host/boston.example.com@EXAMPLE.COM.

In addition, the Service Management Facility has been configured so that many of the daemons or commands do not start if the DNS client service is not running. The kdb5_util, kadm5, and kpropd daemons, as well as the kprop command all are configured to depend on the DNS service. To fully utilize the features available using the Kerberos service and SMF, you must enable the DNS client service on all hosts.

### Ports for the KDC and Admin Services

By default, port 88 and port 750 are used for the KDC, and port 749 is used for the KDC administration daemon. Different port numbers can be used. However, if you change the port numbers, then the /etc/services and /etc/krb5/krb5.conf files must be changed on every client. In addition to these files, the /etc/krb5/kdc.conf file on each KDC must be updated.

### The Number of Slave KDCs

Slave KDCs generate credentials for clients just as the master KDC does. Slave KDCs provide backup if the master becomes unavailable. Each realm should have at least one slave KDC. Additional slave KDCs might be required, depending on these factors:

- The number of physical segments in the realm. Normally, the network should be set up so that each segment can function, at least minimally, without the rest of the realm. To do so, a KDC must be accessible from each segment. The KDC in this instance could be either a master or a slave.
- The number of clients in the realm. By adding more slave KDC servers, you can reduce the load on the current servers.

It is possible to add too many slave KDCs. Remember that the KDC database must be propagated to each server, so the more KDC servers that are installed, the longer it can take to
get the data updated throughout the realm. Also, because each slave retains a copy of the KDC
database, more slaves increase the risk of a security breach.

In addition, one or more slave KDCs can easily be configured to be swapped with the master
KDC. The advantage of configuring at least one slave KDC in this way is that if the master KDC
fails for any reason, you will have a system preconfigured that will be easy to swap as the master
KDC. For instructions on how to configure a swappable slave KDC, see “Swapping a Master
KDC and a Slave KDC” on page 438.

Mapping GSS Credentials to UNIX Credentials

The Kerberos service provides a default mapping of GSS credential names to UNIX user IDs
(UIDs) for GSS applications that require this mapping, such as NFS. GSS credential names are
equivalent to Kerberos principal names when using the Kerberos service. The default mapping
algorithm is to take a one component Kerberos principal name and use that component, which
is the primary name of the principal, to look up the UID. The look up occurs in the default
realm or any realm that is allowed by using the auth_to_local_realm parameter in
/etc/krb5/krb5.conf. For example, the user principal name bob@example.com is mapped to
the UID of the UNIX user named bob using the password table. The user principal name
bob/admin@example.com would not be mapped, because the principal name includes an
instance component of admin. If the default mappings for the user credentials are sufficient, the
GSS credential table does not need to be populated. In past releases, populating the GSS
credential table was required to get the NFS service to work. If the default mapping is not
sufficient, for example if you want to map a principal name which contains an instance
component, then other methods should be used. For more information see:

- “How to Create a Credential Table” on page 415
- “How to Add a Single Entry to the Credential Table” on page 416
- “How to Provide Credential Mapping Between Realms” on page 417
- “Observing Mapping from GSS Credentials to UNIX Credentials” on page 477

Automatic User Migration to a Kerberos Realm

UNIX users who do not have valid user accounts in the default Kerberos realm can be
automatically migrated using the PAM framework. Specifically, the pam_krb5_migrate module
would be used in the authentication stack of the PAM service. Services would be setup up so that
whenever a user, who does not have a Kerberos principal, performs a successful log in to a
system using their password, a Kerberos principal would be automatically created for that user.
The new principal password would be the same as the UNIX password. See “How to Configure
Automatic Migration of Users in a Kerberos Realm” on page 434 for instructions on how to use
the pam_krb5_migrate module.
Which Database Propagation System to Use

The database that is stored on the master KDC must be regularly propagated to the slave KDCs. You can configure the propagation of the database to be incremental. The incremental process propagates only updated information to the slave KDCs, rather than the entire database. For more information about database propagation, see “Administering the Kerberos Database” on page 443.

If you do not use incremental propagation, one of the first issues to resolve is how often to update the slave KDCs. The need to have up-to-date information that is available to all clients must be weighed against the amount of time it takes to complete the update.

In large installations with many KDCs in one realm, one or more slaves can propagate the data so that the process is done in parallel. This strategy reduces the amount of time that the update takes, but it also increases the level of complexity in administering the realm. For a complete description of this strategy, see “Setting Up Parallel Propagation” on page 455.

Clock Synchronization Within a Realm

All hosts that participate in the Kerberos authentication system must have their internal clocks synchronized within a specified maximum amount of time. Known as clock skew, this feature provides another Kerberos security check. If the clock skew is exceeded between any of the participating hosts, requests are rejected.

One way to synchronize all the clocks is to use the Network Time Protocol (NTP) software. See “Synchronizing Clocks Between KDCs and Kerberos Clients” on page 437 for more information. Other ways of synchronizing the clocks are available, so the use of NTP is not required. However, some form of synchronization should be used to prevent access failures because of clock skew.

Client Configuration Options

A new feature in the Solaris 10 release is the kclient configuration utility. The utility can be run in interactive mode or noninteractive mode. In interactive mode, the user is prompted for Kerberos-specific parameter values, which allows the user to make changes to the existing installation when configuring the client. In noninteractive mode, a file with previously set parameter values is used. Also, command-line options can be used in the noninteractive mode. Both interactive and noninteractive modes require less steps than the manual process, which should make the process quicker and less prone to error.
In the Solaris Express Developer Edition 1/08 release, changes were made to allow for a zero-configuration Kerberos client. If these rules are followed in your environment then no explicit configuration procedure is necessary for a Solaris Kerberos client:

- DNS is configured to return SRV records for KDCs.
- The realm name matches the DNS domain name or the KDC supports referrals.
- The Kerberos client does not require a keytab.

In some cases it may be better to explicitly configure the Kerberos client:

- If referrals are not used, the zero-configuration logic depends on the DNS domain name of the host to determine the realm. This introduces a small security risk, but the risk is much smaller than enabling `dns_lookup_realm`.
- The `pam_krb5` module relies on a host key entry in the keytab. This requirement may be disabled in the `krb5.conf` file however it is not recommend for security reasons. See the `krb5.conf(4)` man page.
- The zero-configuration process is less efficient than direct configuration, and has a greater reliance on DNS. The process performs more DNS lookups than a directly configured client.

See "Configuring Kerberos Clients" on page 419 for a description of all the client configuration processes.

### Improving Client Login Security

In the Solaris 10 11/06 release, on login a client, using the `pam_krb5` module, verifies that the KDC that issued the latest TGT, is the same KDC that issued the client host principal that is stored in `/etc/krb5/krb5.keytab`. The `pam_krb5` module verifies the KDC when the module is configured in the authentication stack. For some configurations, like DHCP clients that do not store a client host principal, this check needs to be disabled. To turn off this check, you must set the `verify_ap_req_nofail` option in the `krb5.conf` file to be false. See "How to Disable Verification of the Ticket Granting Ticket (TGT)" on page 432 for more information.

### KDC Configuration Options

Starting in the the Solaris Express Developer Edition 1/08 release, there are several ways to configure a KDC. The simplest versions use the `kdcmgr` utility to configure the KDC automatically or interactively. The automatic version requires that you use command line options to define the configuration parameters. This method is especially useful for scripts. The interactive version prompts you for all information that is needed. See Table 23–1 for pointers to the instructions for using this command.

In addition, starting in the Solaris Express Developer Edition 1/08 release, support for using LDAP to manage the database files for Kerberos has been added. See “How to Configure a KDC"
to Use an LDAP Data Server” on page 396 for instructions. Using LDAP simplifies administration for sites that require better coordination between the Solaris Kerberos databases and their existing DS setup.

Kerberos Encryption Types

An encryption type is an identifier that specifies the encryption algorithm, encryption mode, and hash algorithms used in the Kerberos service. The keys in the Kerberos service have an associated encryption type to identify the cryptographic algorithm and mode to be used when the service performs cryptographic operations with the key. Here are the supported encryption types:

- des-cbc-md5
- des-cbc-crc
- des3-cbc-sha1-kd
- arcfour-hmac-md5
- arcfour-hmac-md5-exp
- aes128-cts-hmac-sha1-96
- aes256-cts-hmac-sha1-96

Note – In releases prior to Solaris 10 8/07 release, the aes256-cts-hmac-sha1-96 encryption type can be used with the Kerberos service if the unbundled Strong Cryptographic packages are installed.

If you want to change the encryption type, you should do so when creating a new principal database. Because of the interaction between the KDC, the server, and the client, changing the encryption type on an existing database is difficult. Leave these parameters unset unless you are re-creating the database. Refer to “Using Kerberos Encryption Types” on page 545 for more information.

Note – If you have a master KDC installed that is not running the Solaris 10 release, the slave KDCs must be upgraded to the Solaris 10 release before you upgrade the master KDC. A Solaris 10 master KDC will use the new encryption types, which an older slave will not be able to handle.
Online Help URL in the Graphical Kerberos Administration Tool

The online help URL is used by the Graphical Kerberos Administration Tool, gkadmin, so the URL should be defined properly to enable the “Help Contents” menu to work. The HTML version of this manual can be installed on any appropriate server. Alternately, you can decide to use the collections at http://docs.sun.com.

The URL is specified in the krb5.conf file when configuring a host to use the Kerberos service. The URL should point to the section titled “Graphical Kerberos Administration Tool” in the “Administering Principals and Policies (Tasks)” chapter in this book. You can choose another HTML page, if another location is more appropriate.
This chapter provides configuration procedures for KDC servers, network application servers, NFS servers, and Kerberos clients. Many of these procedures require superuser access, so they should be used by system administrators or advanced users. Cross-realm configuration procedures and other topics related to KDC servers are also covered.

The following topics are covered.

- “Configuring the Kerberos Service (Task Map)” on page 387
- “Configuring KDC Servers” on page 388
- “Configuring Kerberos Clients” on page 419
- “Configuring Cross-Realm Authentication” on page 409
- “Configuring Kerberos Network Application Servers” on page 411
- “Configuring Kerberos NFS Servers” on page 413
- “Synchronizing Clocks Between KDCs and Kerberos Clients” on page 437
- “Swapping a Master KDC and a Slave KDC” on page 438
- “Administering the Kerberos Database” on page 443
- “Increasing Security on Kerberos Servers” on page 459

**Configuring the Kerberos Service (Task Map)**

Parts of the configuration process depend on other parts and must be done in a specific order. These procedures often establish services that are required to use the Kerberos service. Other procedures are not dependent on any order, and can be done when appropriate. The following task map shows a suggested order for a Kerberos installation.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Plan for your Kerberos installation.</td>
<td>Lets you resolve configuration issues before you start the software configuration process. Planning ahead saves you time and other resources in the long run.</td>
<td>Chapter 22, “Planning for the Kerberos Service”</td>
</tr>
</tbody>
</table>
Configure cross-realm authentication. Enables communications from one realm to another realm. "Configuring Cross-Realm Authentication" on page 409

Configure Kerberos application servers. Enables a server to support services such as ftp, telnet, and rsh using Kerberos authentication. "Configuring Kerberos Network Application Servers" on page 411

Configure Kerberos clients. Enables a client to use Kerberos services. "Configuring Kerberos Clients" on page 419

Configure Kerberos NFS server. Enables a server to share a file system that requires Kerberos authentication. "Configuring Kerberos NFS Servers" on page 413

Increase security on an application server. Increases security on an application server by restricting access to authenticated transactions only. "How to Enable Only Kerberized Applications" on page 459

Configuring KDC Servers

After you install the Kerberos software, you must configure the KDC servers. Configuring a master KDC and at least one slave KDC provides the service that issues credentials. These credentials are the basis for the Kerberos service, so the KDCs must be installed before you attempt other tasks.

The most significant difference between a master KDC and a slave KDC is that only the master KDC can handle database administration requests. For instance, changing a password or
adding a new principal must be done on the master KDC. These changes can then be propagated to the slave KDCs. Both the slave KDC and master KDC generate credentials. This feature provides redundancy in case the master KDC cannot respond.

### TABLE 23–1 Configuring KDC Servers (Task Map)

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuring a Master KDC</td>
<td>Configures and builds the master KDC server and database for a realm using an automatic process, which is good for scripts.</td>
<td>“How to Automatically Configure a Master KDC” on page 389</td>
</tr>
<tr>
<td></td>
<td>Configures and builds the master KDC server and database for a realm using an interactive process, which is sufficient for most installations</td>
<td>“How to Interactively Configure a Master KDC” on page 390</td>
</tr>
<tr>
<td></td>
<td>Configures and builds the master KDC server and database for a realm using a manual process, which is needed for more complex installations</td>
<td>“How to Manually Configure a Master KDC” on page 391</td>
</tr>
<tr>
<td></td>
<td>Configures and builds the master KDC server and database for a realm using a manual process and using LDAP for the KDC</td>
<td>“How to Configure a KDC to Use an LDAP Data Server” on page 396</td>
</tr>
<tr>
<td>Configuring a slave KDC server.</td>
<td>Configures and builds a slave KDC server for a realm using an automatic process, which is good for scripts</td>
<td>“How to Automatically Configure a Slave KDC” on page 402</td>
</tr>
<tr>
<td></td>
<td>Configures and builds a slave KDC server for a realm using an interactive process, which is sufficient for most installations</td>
<td>“How to Interactively Configure a Slave KDC” on page 403</td>
</tr>
<tr>
<td></td>
<td>Configures and builds a slave KDC server for a realm using a manual process, which is needed for more complex installations</td>
<td>“How to Manually Configure a Slave KDC” on page 404</td>
</tr>
<tr>
<td>Refreshing principal keys on a KDC server.</td>
<td>Updates the session key on a KDC server to use new encryption types.</td>
<td>“How to Refresh the Ticket Granting Service Keys on a Master Server” on page 408</td>
</tr>
</tbody>
</table>

### How to Automatically Configure a Master KDC

Starting with the Solaris Express Developer Edition 1/08 release, a master KDC can be automatically configured by using the following procedure.

1. **Become superuser.**
2 Create the KDC.
Run the kdcmgr utility to create the KDC. You need to provide both the master key password and the password for the administrative principal.

```
kdc1# kdcmgr -a kws/admin -r EXAMPLE.COM create master
```

Starting server setup
---------------------------------------

```
Setting up /etc/krb5/kdc.conf

Setting up /etc/krb5/krb5.conf

Initializing database '/var/krb5/principal' for realm 'EXAMPLE.COM',
master key name 'K/M@EXAMPLE.COM'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key: <Type the password>
Re-enter KDC database master key to verify: <Type it again>

Authenticating as principal root/admin@EXAMPLE.COM with password.
WARNING: no policy specified for kws/admin@EXAMPLE.COM; defaulting to no policy
Enter password for principal "kws/admin@EXAMPLE.COM": <Type the password>
Re-enter password for principal "kws/admin@EXAMPLE.COM": <Type it again>
Principal "kws/admin@EXAMPLE.COM" created.
```

Setting up /etc/krb5/kadm5.acl.
-----------------------------------

Setup COMPLETE.
```
kdc1#
```

▼ How to Interactively Configure a Master KDC
Starting with the Solaris Express Developer Edition 1/08 release, a master KDC can be interactively configured by using the following procedure.

1 Become superuser.

2 Create the KDC.
Run the kdcmgr utility to create the KDC. You need to provide both the master key password and the password for the administrative principal.

```
kdc1# kdcmgr create master
```

Starting server setup
Enter the Kerberos realm: EXAMPLE.COM

Setting up /etc/krb5/kdc.conf

Setting up /etc/krb5/krb5.conf

Initializing database '/var/krb5/principal' for realm 'EXAMPLE.COM',
master key name 'K/M@EXAMPLE.COM'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key: <Type the password>
Re-enter KDC database master key to verify: <Type it again>

Enter the krb5 administrative principal to be created: kws/admin

Authenticating as principal root/admin@EXAMPLE.COM with password.
WARNING: no policy specified for kws/admin@EXAMPLE.COM; defaulting to no policy
Enter password for principal "kws/admin@EXAMPLE.COM": <Type the password>
Re-enter password for principal "kws/admin@EXAMPLE.COM": <Type it again>
Principal "kws/admin@EXAMPLE.COM" created.

Setting up /etc/krb5/kadm5.acl.

---------------------------------------------------
Setup COMPLETE.

kdc1#

Example 23–1  Displaying the Status of a KDC Server

The kdcmgr status command can be used to display information about either a master or a slave KDC server.

▼ How to Manually Configure a Master KDC

In this procedure, incremental propagation is configured. In addition, the following configuration parameters are used:

- Realm name = EXAMPLE.COM
- DNS domain name = example.com
- Master KDC = kdc1.example.com
- admin principal = kws/admin
Configuring KDC Servers

**Note** – Adjust the URL to point to the “Graphical Kerberos Administration Tool” section, as described in “Online Help URL in the Graphical Kerberos Administration Tool” on page 385.

**Before You Begin**
This procedure requires that the host is configured to use DNS. For specific naming instructions if this master is to be swappable, see “Swapping a Master KDC and a Slave KDC” on page 438.

1. **Become superuser on the master KDC.**

2. **Edit the Kerberos configuration file** (*krb5.conf*).

   You need to change the realm names and the names of the servers. See the *krb5.conf*(4) man page for a full description of this file.

   ```bash
   kdc1 # cat /etc/krb5/krb5.conf
   [libdefaults]
     default_realm = EXAMPLE.COM
   
   [realms]
   EXAMPLE.COM = {
     kdc = kdc1.example.com
     admin_server = kdc1.example.com
   }
   
   [domain_realm]
   .example.com = EXAMPLE.COM
   #
   # if the domain name and realm name are equivalent,
   # this entry is not needed
   #
   [logging]
   default = FILE:/var/krb5/kdc.log
   kdc = FILE:/var/krb5/kdc.log
   
   [appdefaults]
   gkadmin = {
   }
   ```

   In this example, the lines for `default_realm`, `kdc`, `admin_server`, and all `domain_realm` entries were changed. In addition, the line that defines the `help_url` was edited.
3 Edit the KDC configuration file (`kdc.conf`).
You need to change the realm name. See the `kdc.conf(4)` man page for a full description of this file.

```bash
ekdc1 # cat /etc/krb5/kdc.conf
[kdcdefaults]
kdc_ports = 88,750

[realms]
EXAMPLE.COM = {
    profile = /etc/krb5/krb5.conf
    database_name = /var/krb5/principal
    acl_file = /etc/krb5/kadm5.acl
    kadmind_port = 749
    max_life = 8h 0m 0s
    max_renewable_life = 7d 0h 0m 0s
    sunw_dbprop_enable = true
    sunw_dbprop_master_ulogfile = 1000
}
```

In this example, the realm name definition in the `realms` section was changed. Also, in the `realms` section, lines to enable incremental propagation and to select the number of updates the KDC master keeps in the log were added.

Note – If you want to restrict the encryption types, you can set the `permitted_enctypes`, `supported_enctypes`, or `master_key_type` lines. Refer to “Using Kerberos Encryption Types” on page 545 for a description of the issues involved with restricting the encryption types.

4 Create the KDC database by using the `kdb5_util` command.
The `kdb5_util` command creates the KDC database. Also, when used with the `-s` option, this command creates a stash file that is used to authenticate the KDC to itself before the `kadmind` and `krb5kdc` daemons are started.

```bash
kdc1 # /usr/sbin/kdb5_util create -s
Initializing database '/var/krb5/principal' for realm 'EXAMPLE.COM'
master key name 'K/M@EXAMPLE.COM'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key: <Type the key>
Re-enter KDC database master key to verify: <Type it again>
```

Note – If you want to restrict the encryption types, you can set the `default_tkt_enctypes` or `default_tgs_enctypes` lines. Refer to “Using Kerberos Encryption Types” on page 545 for a description of the issues involved with restricting the encryption types.
5 **Edit the Kerberos access control list file** ([kad5.acl](https://example.com/kad5.acl)).

Once populated, the `/etc/krb5/kad5.acl` file should contain all principal names that are allowed to administer the KDC.

```
    kws/admin@EXAMPLE.COM   *
```

The entry gives the `kws/admin` principal in the `EXAMPLE.COM` realm the ability to modify principals or policies in the KDC. The default installation includes an asterisk (*) to match all `admin` principals. This default could be a security risk, so it is more secure to include a list of all of the `admin` principals. See the `kad5.acl(4)` man page for more information.

6 **Start the `kadmin.local` command and add principals.**

The next substeps create principals that are used by the Kerberos service.

```
kdc1 # /usr/sbin/kadmin.local
kadmin.local:
```

**a. Add administration principals to the database.**

You can add as many `admin` principals as you need. You must add at least one `admin` principal to complete the KDC configuration process. For this example, a `kws/admin` principal is added. You can substitute an appropriate principal name instead of "kws.”

```
kadmin.local: addprinc kws/admin
Enter password for principal kws/admin@EXAMPLE.COM: <Type the password>
Re-enter password for principal kws/admin@EXAMPLE.COM: <Type it again>
Principal "kws/admin@EXAMPLE.COM" created.
kadmin.local:
```

**b. Create the kprop principals.**

The `kprop` principal is used to authorize updates from the master KDC.

```
kadmin.local: addprinc -randkey kprop/kdc1.example.com
Principal "kprop/kdc1.example.com@EXAMPLE.COM" created.
kadmin.local:
```

**c. Quit `kadmin.local`.**

You have added all of the required principals for the next steps.

```
kadmin.local: quit
```

7 **Start the Kerberos daemons.**

```
kdc1 # svcadm enable -r network/security/krb5kdc
kdc1 # svcadm enable -r network/security/kadmin
```
8 Start \texttt{kadmin} and add more principals.

At this point, you can add principals by using the Graphical Kerberos Administration Tool. To do so, you must log in with one of the \texttt{admin} principal names that you created earlier in this procedure. However, the following command-line example is shown for simplicity.

\texttt{kdc1# /usr/sbin/kadmin -p kws/admin}

Enter password: \texttt{<Type kws/admin password>}

\texttt{kadmin:}

\textbf{a. Create the master KDC host principal.}

The host principal is used by Kerberized applications, such as \texttt{kprop} to propagate changes to the slave KDCs. This principal is also used to provide secure remote access to the KDC server using applications, like \texttt{ssh}. Note that when the principal instance is a host name, the FQDN must be specified in lowercase letters, regardless of the case of the domain name in the /etc/resolv.conf file.

\texttt{kadmin: addprinc -randkey host/kdc1.example.com}

Principal "host/kdc1.example.com@EXAMPLE.COM" created.

\texttt{kadmin:}

\textbf{b. (Optional) Create the kclient principal.}

This principal is used by the \texttt{kclient} utility during the installation of a Kerberos client. If you do not plan on using this utility, then you do not need to add the principal. The users of the \texttt{kclient} utility need to use this password.

\texttt{kadmin: addprinc clntconfig/admin}

Enter password for principal clntconfig/admin@EXAMPLE.COM: \texttt{<Type the password>}

Re-enter password for principal clntconfig/admin@EXAMPLE.COM: \texttt{<Type it again>}

Principal "clntconfig/admin@EXAMPLE.COM" created.

\texttt{kadmin:}

\textbf{c. Add the master KDC's host principal to the master KDC's keytab file.}

Adding the host principal to the keytab file allows this principal to be used by application servers, like \texttt{sshd}, automatically.

\texttt{kadmin: ktadd host/kdc1.example.com}

Entry for principal host/kdc1.example.com with kvno 3, encryption type AES-256 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.

Entry for principal host/kdc1.example.com with kvno 3, encryption type AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.

Entry for principal host/kdc1.example.com with kvno 3, encryption type Triple DES cbc mode with HMAC/shal added to keytab WRFILE:/etc/krb5/krb5.keytab.

Entry for principal host/kdc1.example.com with kvno 3, encryption type ArcFour with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.

Entry for principal host/kdc1.example.com with kvno 3, encryption type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.

\texttt{kadmin:}
d. Quit kadmin.

kadmin: quit

9 (Optional) Synchronize the master KDCs clock by using NTP or another clock synchronization mechanism.

Installing and using the Network Time Protocol (NTP) is not required. However, every clock must be within the default time that is defined in the libdefaults section of the krb5.conf file for authentication to succeed. See “Synchronizing Clocks Between KDCs and Kerberos Clients” on page 437 for information about NTP.

10 Configure Slave KDCs.

To provide redundancy, make sure to install at least one slave KDC. See “How to Manually Configure a Slave KDC” on page 404 for specific instructions.

▼ How to Configure a KDC to Use an LDAP Data Server

Starting with the Solaris 10 5/08 and the Solaris Express Developer Edition 1/08 releases, a KDC can be configured to use an LDAP data server by using the following procedure.

In this procedure, the following configuration parameters are used:

- Realm name = EXAMPLE.COM
- DNS domain name = example.com
- Master KDC = kdc1.example.com
- Directory Server = dsserver.example.com
- admin principal = kws/admin

Note – Adjust the URL to point to the “Graphical Kerberos Administration Tool” section, as described in “Online Help URL in the Graphical Kerberos Administration Tool” on page 385.

Before You Begin

This procedure also requires that the host is configured to use DNS. For better performance, install the KDC and the LDAP Directory Service on the same server. In addition, a directory server should be running. The procedure below works with servers using the Sun Java™ Directory Server Enterprise Edition release.

1 Become superuser on the KDC.
2 Configure the master KDC to use SSL to reach the directory server.
   The following steps configure a Solaris Express Developer Release KDC to use the Directory Server 6.1 self-signed certificate. If the certificate has expired, follow the instructions for renewing a certificate in “To Manage Self-Signed Certificates” in *Sun Java System Directory Server Enterprise Edition 6.3 Administration Guide.*
   a. On the directory server, export the self-signed directory server certificate.
   
   ```bash
   # /export/sun-ds6.1/ds6/bin/dsadm show-cert -F der /export/sun-ds6.1/directory2 \
   defaultCert > /tmp/defaultCert.cert.der
   ```
   b. On the master KDC, import the directory server certificate.
   
   ```bash
   # pktool setpin keystore=nss dir=/var/ldap
   # chmod a+r /var/ldap/*.db
   # pktool import keystore=nss objtype=cert trust="CT" infile=/tmp/defaultCert.certutil.der \
   label=defaultCert dir=/var/ldap
   ```
   c. On the master KDC, test that SSL is working.
   
   This example assumes that the cn=directory manager entry has administration privileges.
   
   ```bash
   /usr/bin/ldapsearch -Z -P /var/ldap -D "cn=directory manager" \
   -h dsserver.example.com -b "" -s base objectclass="*
   ```
   Subject:
   "CN=dsserver.example.com,CN=636,CN=Directory Server,O=Example Corporation"
   
   Note that the CN=dsserver.example.com entry should include the fully qualified host name, not a short version.

3 Populate the LDAP directory, if necessary.

4 Add the Solaris Kerberos schema to the existing schema.

   ```bash
   # ldapmodify -h dsserver.example.com -D "cn=directory manager" -f /usr/share/lib/ldif/kerberos.ldif
   ```

5 Create the Kerberos container in the LDAP directory.
   Add the following entries to the krb5.conf file.
   a. Define the database type.
      Add an entry to define the database module to the realms section.
      ```bash
      database_module = LDAP
      ```
   b. Define the database module.
      ```bash
      [dbmodules]
      LDAP = {
      ldap_kerberos_container_dn = "cn=krbcontainer,dc=example,dc=com"
      db_library = kldap
      ldap_kdc_dn = "cn=kdc service,ou=profile,dc=example,dc=com"
      ```
c. **Create the KDC in the LDAP directory.**

This command creates krbcontainer and several other objects. It also creates a /var/krb5/.k5.EXAMPLE.COM master key stash file.

```
# kdb5_ldap_util -D "cn=directory manager" create -P abcd1234 -r EXAMPLE.COM -s
```

6 **Stash the KDC bind Distinguished Name (DN) passwords.**

These passwords are used by the KDC when it binds to the DS. The KDC uses different roles depending on the type of access the KDC is using.

```
# kdb5_ldap_util stashsrvpw "cn=kdc service,ou=profile,dc=example,dc=com"
# kdb5_ldap_util stashsrvpw "cn=kadmin service,ou=profile,dc=example,dc=com"
```

7 **Add KDC service roles.**

a. **Create a kdc_roles.ldif file with contents like this:**

```
dn: cn=kdc service,ou=profile,dc=example,dc=com
  cn: kdc service
  sn: kdc service
  objectclass: top
  objectclass: person
  userpassword: test123

dn: cn=kadmin service,ou=profile,dc=example,dc=com
  cn: kadmin service
  sn: kadmin service
  objectclass: top
  objectclass: person
  userpassword: test123
```

b. **Create the role entries in the LDAP directory**

```
# ldapmodify -a -h dsserver.example.com -D "cn=directory manager" -f kdc_roles.ldif
```

8 **Set the ACLs for the KDC-related roles.**

```
# cat << EOF | ldapmodify -h dsserver.example.com -D "cn=directory manager"
# Set kadmin ACL for everything under krbcontainer.
 dn: cn=krbcontainer,dc=example,dc=com
 changetype: modify
 add: acl
 acl: (target="ldap:///cn=krbcontainer,dc=example,dc=com") (targetattr="krb") (version 3.0; acl kadmin_ACL; allow (all); userdn = "ldap:///cn=kadmin service,ou=profile,dc=example,dc=com")
EOF
```
# Set kadmin ACL for everything under the people subtree if there are
# mix-in entries for krb princs:
dn: ou=people,dc=example,dc=com
changetype: modify
add: aci
aci: (target="ldap:///ou=people,dc=example,dc=com"
    acl kadmin_ACL; allow (all)\n    userdn = "ldap:///cn=kadmin\n    service,ou=profile,dc=example,dc=com";)
EOF

9 Edit the Kerberos configuration file (`krb5.conf`).

You need to change the realm names and the names of the servers. See the `krb5.conf(4)` man page for a full description of this file.

```
# kdc1 # cat /etc/krb5/krb5.conf
[libdefaults]
    default_realm = EXAMPLE.COM

[realms]
    EXAMPLE.COM = {
        kdc = kdc1.example.com
        admin_server = kdc1.example.com
    }

[domain_realm]
    .example.com = EXAMPLE.COM

# if the domain name and realm name are equivalent,
# this entry is not needed
#[logging]
    default = FILE:/var/krb5/kdc.log
    kdc = FILE:/var/krb5/kdc.log

[appdefaults]
gkadmin = {
}
```

In this example, the lines for `default_realm`, `kdc`, `admin_server`, and all `domain_realm` entries were changed. In addition, the line that defines the `help_url` was edited.

**Note** – If you want to restrict the encryption types, you can set the `default_tkt_enctypes` or `default_tgs_enctypes` lines. Refer to “Using Kerberos Encryption Types” on page 545 for a description of the issues involved with restricting the encryption types.
10 Edit the KDC configuration file (/etc/krb5/kdc.conf).

You need to change the realm name. See the `/etc/krb5/kdc.conf` man page for a full description of this file.

```bash
kdc1 # cat /etc/krb5/kdc.conf
[kdcdefaults]
    kdc_ports = 88,750

[realms]
    EXAMPLE.COM = {
        profile = /etc/krb5/krb5.conf
        database_name = /var/krb5/principal
        acl_file = /etc/krb5/kadm5.acl
        kadmind_port = 749
        max_life = 8h 0m 0s
        max_renewable_life = 7d 0h 0m 0s
        sunw_dbprop_enable = true
        sunw_dbprop_master_ulogsize = 1000
    }
```

In this example, the realm name definition in the `realms` section was changed. Also, in the `realms` section, lines to enable incremental propagation and to select the number of updates the KDC master keeps in the log were added.

Note – If you want to restrict the encryption types, you can set the `permitted_enctypes`, `supported_enctypes`, or `master_key_type` lines. Refer to “Using Kerberos Encryption Types” on page 545 for a description of the issues involved with restricting the encryption types.

11 Edit the Kerberos access control list file (/etc/krb5/kadm5.acl).

Once populated, the `/etc/krb5/kadm5.acl` file should contain all principal names that are allowed to administer the KDC.

```bash
kws/admin@EXAMPLE.COM *
```

The entry gives the `kws/admin` principal in the `EXAMPLE.COM` realm the ability to modify principals or policies in the KDC. The default installation includes an asterisk (*) to match all `admin` principals. This default could be a security risk, so it is more secure to include a list of all of the `admin` principals. See the `/etc/krb5/kadm5.acl` man page for more information.
12 Start the `kadmin.local` command and add principals.

The next substeps create principals that are used by the Kerberos service.

```
kdc1 # /usr/sbin/kadmin.local
kadmin.local:
```

a. Add administration principals to the database.

You can add as many admin principals as you need. You must add at least one admin principal to complete the KDC configuration process. For this example, a `kws/admin` principal is added. You can substitute an appropriate principal name instead of "kws."

```
kadmin.local: addprinc kws/admin
Enter password for principal kws/admin@EXAMPLE.COM:  <Type the password>
Re-enter password for principal kws/admin@EXAMPLE.COM:  <Type it again>
Principal "kws/admin@EXAMPLE.COM" created.
kadmin.local:
```

b. Quit `kadmin.local`.

You have added all of the required principals for the next steps.

```
kadmin.local: quit
```

13 Start the Kerberos daemons.

```
kdc1 # ssvcadm enable -r network/security/krb5kdc
kdc1 # ssvcadm enable -r network/security/kadmin
```

14 Start `kadmin` and add more principals.

At this point, you can add principals by using the Graphical Kerberos Administration Tool. To do so, you must log in with one of the admin principal names that you created earlier in this procedure. However, the following command-line example is shown for simplicity.

```
kdc1 # /usr/sbin/kadmin -p kws/admin
Enter password:  <Type kws/admin password>
kadmin:
```

a. Create the master KDC host principal.

The host principal is used by Kerberized applications, such as `klist` and `kprop`. Solaris 10 clients use this principal when mounting an authenticated NFS file system. Note that when the principal instance is a host name, the FQDN must be specified in lowercase letters, regardless of the case of the domain name in the `/etc/resolv.conf` file.

```
kadmin: addprinc -randkey host/kdc1.example.com
Principal "host/kdc1.example.com@EXAMPLE.COM" created.
kadmin:
```
b. **Optional** Create the `kclient` principal.

This principal is used by the `kclient` utility during the installation of a Kerberos client. If you do not plan on using this utility, then you do not need to add the principal. The users of the `kclient` utility need to use this password.

```
ktadd principal clntconfig/admin
Enter password for principal clntconfig/admin@EXAMPLE.COM: <Type the password>
Re-enter password for principal clntconfig/admin@EXAMPLE.COM: <Type it again>
Principal "clntconfig/admin@EXAMPLE.COM" created.
```

```
kadmin:
```

c. Add the master KDC's host principal to the master KDC's keytab file.

Adding the host principal to the keytab file allows this principal to be used automatically.

```
kadmin: ktadd host/kdc1.example.com
Entry for principal host/kdc1.example.com with kvno 3, encryption type AES-256 CTS mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc1.example.com with kvno 3, encryption type AES-128 CTS mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc1.example.com with kvno 3, encryption type Triple DES cbc
mode with HMAC/shal added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc1.example.com with kvno 3, encryption type ArcFour
with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc1.example.com with kvno 3, encryption type DES cbc mode
with RSA-MDS added to keytab WRFILE:/etc/krb5/krb5.keytab.
```

```
kadmin:
```

d. Quit `kadmin`.

```
kadmin: quit
```

⚠️ **How to Automatically Configure a Slave KDC**

Starting with the Solaris Express Developer Edition 1/08 release, a slave KDC can be automatically configured by using the following procedure.

---

**15** (Optional) Synchronize the master KDCs clock by using NTP or another clock synchronization mechanism.

Installing and using the Network Time Protocol (NTP) is not required. However, every clock must be within the default time that is defined in the `libdefaults` section of the `krb5.conf` file for authentication to succeed. See “Synchronizing Clocks Between KDCs and Kerberos Clients” on page 437 for information about NTP.

**16** Configure Slave KDCs.

To provide redundancy, make sure to install at least one slave KDC. See “How to Manually Configure a Slave KDC” on page 404 for specific instructions.
1. Become superuser.

2. Create the KDC.
   
   Run the `kdcsmgr` utility to create the KDC. You need to provide both the master key password and the password for the administrative principal.
   
   ```
   kdc2# kdcsmgr -a kws/admin -r EXAMPLE.COM create -m kdc1 slave
   ```

   Starting server setup
   
   Setting up `/etc/krb5/kdc.conf`
   
   Setting up `/etc/krb5/krb5.conf`
   
   Obtaining TGT for kws/admin ...
   
   Password for kws/admin@EXAMPLE.COM:  
   
   Setting up `/etc/krb5/kadm5.acl`.
   
   Setting up `/etc/krb5/kpropd.acl`.
   
   Waiting for database from master...
   
   Waiting for database from master...
   
   Waiting for database from master...
   
   kdb5 util: Cannot find/read stored master key while reading master key
   
   kdb5 util: Warning: proceeding without master key
   
   Enter KDC database master key:  
   
   Setup COMPLETE.
   
   kdc2#

   ▼ How to Interactively Configure a Slave KDC

   Starting with the Solaris Express Developer Edition 1/08 release, a slave KDC can be interactively configured by using the following procedure.

   1. Become superuser.

   2. Create the KDC.
      
      Run the `kdcsmgr` utility to create the KDC. You need to provide both the master key password and the password for the administrative principal.
      
      ```
      kdc1# kdcsmgr create slave
      ```

      Starting server setup
Enter the Kerberos realm: EXAMPLE.COM
What is the master KDC’s host name?: kdc1

Setting up /etc/krb5/kdc.conf

Setting up /etc/krb5/krb5.conf
Obtaining TGT for kws/admin ...
Password for kws/admin@EXAMPLE.COM: <Type the password>

Setting up /etc/krb5/kadm5.acl.

Setting up /etc/krb5/kpropd.acl.

Waiting for database from master... Waiting for database from master... Waiting for database from master...

kdb5_util: Cannot find/read stored master key while reading master key
kdb5_util: Warning: proceeding without master key

Enter KDC database master key: <Type the password>

---------------------------------------
Setup COMPLETE.

kdc2#

▼ How to Manually Configure a Slave KDC

In this procedure, a new slave KDC named kdc2 is configured. Also, incremental propagation is configured. This procedure uses the following configuration parameters:

- Realm name = EXAMPLE.COM
- DNS domain name = example.com
- Master KDC = kdc1.example.com
- Slave KDC = kdc2.example.com
- admin principal = kws/admin

Before You Begin

The master KDC must be configured. For specific instructions if this slave is to be swappable, see “Swapping a Master KDC and a Slave KDC” on page 438.

1 On the master KDC, become superuser.
2 **On the master KDC, start** `kadmin`.

You must log in with one of the `admin` principal names that you created when you configured the master KDC.

```
kdcl # /usr/sbin/kadmin -p kws/admin
```

Enter password:  *Type kws/admin password*

```
kadmin:
```

a. **On the master KDC, add slave host principals to the database, if not already done.**

For the slave to function, it must have a host principal. Note that when the principal instance is a host name, the FQDN must be specified in lowercase letters, regardless of the case of the domain name in the `/etc/resolv.conf` file.

```
kadmin: addprinc -randkey host/kdc2.example.com
Principal "host/kdc2.example.com@EXAMPLE.COM" created.
kadmin:
```

b. **On the master KDC, create the kiprop principal.**

The kiprop principal is used to authorize incremental propagation from the master KDC.

```
kadmin: addprinc -randkey kiprop/kdc2.example.com
Principal "kiprop/kdc2.example.com@EXAMPLE.COM" created.
kadmin:
```

c. **Quit kadmin.**

```
kadmin: quit
```

3 **On the master KDC, edit the Kerberos configuration file** (`krb5.conf`).

You need to add an entry for each slave. See the `krb5.conf(4)` man page for a full description of this file.

```
kdc1 # cat /etc/krb5/krb5.conf
```

```
[realms]
EXAMPLE.COM = {
    kdc = kdc1.example.com
    kdc = kdc2.example.com
    admin_server = kdc1.example.com
}
```

4 **On the master KDC, add an kiprop entry to** `kadm5.acl`.

This entry allows the master KDC to receive requests for incremental propagation for the kdc2 server.

```
kdc1 # cat /etc/krb5/kadm5.acl
```

```
*/admin@EXAMPLE.COM *
kiprop/kdc2.example.com@EXAMPLE.COM p
```

---

**Chapter 23 • Configuring the Kerberos Service (Tasks)**
On the master KDC, restart `kadmind` to use the new entries in the `kadm5.acl` file.

```
kdc1 # svcadm restart network/security/kadmin
```

On all slave KDCs, copy the KDC administration files from the master KDC server. This step needs to be followed on all slave KDCs, because the master KDC server has updated information that each KDC server needs. You can use `ftp` or a similar transfer mechanism to grab copies of the following files from the master KDC:

- `/etc/krb5/krb5.conf`
- `/etc/krb5/kdc.conf`

On all slave KDCs, add an entry for the master KDC and each slave KDC into the database propagation configuration file, `kpropd.acl`. This information needs to be updated on all slave KDC servers.

```
kdc2 # cat /etc/krb5/kpropd.acl
host/kdc1.example.com@EXAMPLE.COM
host/kdc2.example.com@EXAMPLE.COM
```

On all slave KDCs, make sure that the Kerberos access control list file, `kadm5.acl`, is not populated. An unmodified `kadm5.acl` file would look like:

```
kdc2 # cat /etc/krb5/kadm5.acl
*/admin@___default_realm___ *
```

If the file has `kprop` entries, remove them.

On the new slave, change an entry in `kdc.conf`. Replace the `sunw_dbprop_master_uologsize` entry with an entry defining `sunw_dbprop_slave_poll`. The entry sets the poll time to 2 minutes.

```
kdc1 # cat /etc/krb5/kdc.conf
[kdcdefaults]
kdc_ports = 88,750

[realms]
EXAMPLE.COM=
    profile = /etc/krb5/krb5.conf
database_name = /var/krb5/principal
acl_file = /etc/krb5/kadm5.acl
kadmind_port = 749
max_life = 8h 0m 0s
max_renewable_life = 7d 0h 0m 0s
sunw_dbprop_enable = true
sunw_dbprop_slave_poll = 2m
```
10 On the new slave, start the `kadmin` command.

You must log in with one of the admin principal names that you created when you configured the master KDC.

```
kdc2 # /usr/sbin/kadmin -p kws/admin
Enter password: <Type kws/admin password>
```

a. Add the slave's host principal to the slave's keytab file by using `kadmin`.

This entry allows `kprop` and other Kerberized applications to function. Note that when the principal instance is a host name, the FQDN must be specified in lowercase letters, regardless of the case of the domain name in the `/etc/resolv.conf` file.

```
kadmin: ktadd host/kdc2.example.com
Entry for principal host/kdc2.example.com with kvno 3, encryption type AES-256 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc2.example.com with kvno 3, encryption type AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc2.example.com with kvno 3, encryption type Triple DES cbc mode with HMAC/shal added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc2.example.com with kvno 3, encryption type ArcFour with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc2.example.com with kvno 3, encryption type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
```

b. Add the `kiprop` principal to the slave KDC's keytab file.

Adding the `kiprop` principal to the `krb5.keytab` file allows the `kpropd` command to authenticate itself when incremental propagation is started.

```
kadmin: ktadd kiprop/kdc2.example.com
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type AES-256 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type Triple DES cbc mode with HMAC/shal added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type ArcFour with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
```

c. Quit `kadmin`.

```
kadmin: quit
```

11 On the new slave, start the Kerberos propagation daemon.

```
kdc2 # svcadm enable network/security/krb5_prop
```
12 On the new slave, create a stash file by using kdb5_util.

    kdb5_util: Cannot find/read stored master key while reading master key
    kdb5_util: Warning: proceeding without master key

    Enter KDC database master key: <Type the key>

13 (Optional) On the new slave KDC, synchronize the master KDC’s clock by using NTP or another clock synchronization mechanism.

Installing and using the Network Time Protocol (NTP) is not required. However, every clock must be within the default time that is defined in the l1defaults section of the krb5.conf file for authentication to succeed. See “Synchronizing Clocks Between KDCs and Kerberos Clients” on page 437 for information about NTP.

14 On the new slave, start the KDC daemon (krb5kdc).

    kdc2 # svcadm enable network/security/krb5kdc

▼ How to Refresh the Ticket Granting Service Keys on a Master Server

When the Ticket Granting Service (TGS) principal only has a DES key, which is the case for Solaris KDC servers created prior to the Solaris 10 release, the key restricts the encryption type of the Ticket Granting Ticket (TGT) session key to DES. If a Solaris KDC is updated to Solaris 10 which supports additional, stronger encryption types, the administrator may expect that stronger encryption will be used for all session keys generated by the KDC. However if the existing TGS principal does not have its keys refreshed to include the new encryption types, then the TGT session key will be continue to be limited to DES. The following procedure refreshes the key so that additional encryption types may be used.

- Refresh the TGS service principal key.

    kdb1 % /usr/sbin/kadmin -p kws/admin
    Enter password: <Type kws/admin password>
    kadmin: cpw -randkey krbtgt/EXAMPLE.COM@EXAMPLE.COM

Example 23–2 Refreshing the Principal Keys from a Master Server

If you are logged onto the KDC master as root, you can refresh the TGS service principal with the following command:

    kdc1 # kadmin.local -q 'cpw -randkey krbtgt/EXAMPLE.COM@EXAMPLE.COM'
Configuring Cross-Realm Authentication

You have several ways of linking realms together so that users in one realm can be authenticated in another realm. Cross-realm authentication is accomplished by establishing a secret key that is shared between the two realms. The relationship of the realms can be either hierarchal or directional (see “Realm Hierarchy” on page 379).

▼ How to Establish Hierarchical Cross-Realm Authentication

The example in this procedure uses two realms, ENG.EAST.EXAMPLE.COM and EAST.EXAMPLE.COM. Cross-realm authentication will be established in both directions. This procedure must be completed on the master KDC in both realms.

Before You Begin

The master KDC for each realm must be configured. To fully test the authentication process, several Kerberos clients must be configured.

1 Become superuser on the first master KDC.

2 Create ticket-granting ticket service principals for the two realms.

You must log in with one of the admin principal names that was created when you configured the master KDC.

```bash
# /usr/sbin/kadmin -p kws/admin
Enter password: <Type kws/admin password>
kadmin: addprinc krbtgt/ENG.EAST.EXAMPLE.COM@EAST.EXAMPLE.COM
Enter password for principal krbtgt/ENG.EAST.EXAMPLE.COM@EAST.EXAMPLE.COM: <Type password>
kadmin: addprinc krbtgt/EAST.EXAMPLE.COM@ENG.EAST.EXAMPLE.COM
Enter password for principal krbtgt/EAST.EXAMPLE.COM@ENG.EAST.EXAMPLE.COM: <Type password>
kadmin: quit
```

Note – The password that is specified for each service principal must be identical in both KDCs. Thus, the password for the service principal krbtgt/ENG.EAST.EXAMPLE.COM@EAST.EXAMPLE.COM must be the same in both realms.

3 Add entries to the Kerberos configuration file (krb5.conf) to define domain names for every realm.

```bash
# cat /etc/krb5/krb5.conf
[libdefaults]
.
.
[domain_realm]
```

Chapter 23 • Configuring the Kerberos Service (Tasks)
In this example, domain names for the ENG.EAST.EXAMPLE.COM and EAST.EXAMPLE.COM realms are defined. It is important to include the subdomain first, because the file is searched top down.

4 Copy the Kerberos configuration file to all clients in this realm.
For cross-realm authentication to work, all systems (including slave KDCs and other servers) must have the new version of the Kerberos configuration file (/etc/krb5/krb5.conf) installed.

5 Repeat all of these steps in the second realm.

▼ How to Establish Direct Cross-Realm Authentication

The example in this procedure uses two realms, ENG.EAST.EXAMPLE.COM and SALES.WEST.EXAMPLE.COM. Cross-realm authentication will be established in both directions. This procedure must be completed on the master KDC in both realms.

Before You Begin

The master KDC for each realm must be configured. To fully test the authentication process, several Kerberos clients must be configured.

1 Become superuser on one of the master KDC servers.

2 Create ticket-granting ticket service principals for the two realms.
You must log in with one of the admin principal names that was created when you configured the master KDC.

```
# /usr/sbin/kadmin -p kws/admin
Enter password: <Type kws/admin password>
kadmin: addprinc krbtgt/ENG.EAST.EXAMPLE.COM@SALES.WEST.EXAMPLE.COM
```

Enter password for principal:
```
krtgt/ENG.EAST.EXAMPLE.COM@SALES.WEST.EXAMPLE.COM: <Type the password>
kadmin: addprinc krbtgt/SALES.WEST.EXAMPLE.COM@ENG.EAST.EXAMPLE.COM
```

Enter password for principal:
```
krgtgt/SALES.WEST.EXAMPLE.COM@ENG.EAST.EXAMPLE.COM: <Type the password>
kadmin: quit
```

Note – The password that is specified for each service principal must be identical in both KDCs. Thus, the password for the service principal krbtgt/ENG.EAST.EXAMPLE.COM@SALES.WEST.EXAMPLE.COM must be the same in both realms.
3 Add entries in the Kerberos configuration file to define the direct path to the remote realm.
This example shows the clients in the ENG.EAST.EXAMPLE.COM realm. You would need to swap the realm names to get the appropriate definitions in the SALES.WEST.EXAMPLE.COM realm.

```
# cat /etc/krb5/krb5.conf
[libdefaults]
.
.
[capaths]
  ENG.EAST.EXAMPLE.COM = {
    SALES.WEST.EXAMPLE.COM = .
  }
  SALES.WEST.EXAMPLE.COM = {
    ENG.EAST.EXAMPLE.COM = .
  }
```

4 Copy the Kerberos configuration file to all clients in the current realm.
For cross-realm authentication to work, all systems (including slave KDCs and other servers) must have the new version of the Kerberos configuration file (/etc/krb5/krb5.conf) installed.

5 Repeat all of these steps for the second realm.

## Configuring Kerberos Network Application Servers

Network application servers are hosts that provide access using one or more of the following network applications: ftp, rcp, rlogin, rsh, ssh, and telnet. Only a few steps are required to enable the Kerberos version of these commands on a server.

### ▼ How to Configure a Kerberos Network Application Server

This procedure uses the following configuration parameters:

- Application server = boston
- admin principal = kws/admin
- DNS domain name = example.com
- Realm name = EXAMPLE.COM

**Before You Begin** This procedure requires that the master KDC has been configured. To fully test the process, several Kerberos clients must be configured.
1. **(Optional) Install the NTP client or another clock synchronization mechanism.**
   
   See “Synchronizing Clocks Between KDCs and Kerberos Clients” on page 437 for information about NTP.

2. **Add principals for the new server and update the server’s keytab.**
   
   The following command reports the existence of the host principal:
   
   ```bash
   boston # klist -k | grep host
   4 host/boston.example.com@EXAMPLE.COM
   4 host/boston.example.com@EXAMPLE.COM
   4 host/boston.example.com@EXAMPLE.COM
   4 host/boston.example.com@EXAMPLE.COM
   ```

   If the command does not return a principal, then create new principals using the following steps.

   How to use the Graphical Kerberos Administration Tool to add a principal is explained in “How to Create a New Kerberos Principal” on page 489. The example in the following steps shows how to add the required principals using the command line. You must login with one of the admin principal names that you created when configuring the master KDC.

   ```bash
   boston # /usr/sbin/kadmin -p kws/admin
   Enter password: <Type kws/admin password>
   kadmin:
   ```

   a. **Create the server’s host principal.**
      
      The host principal is used:
      
      - To authenticate traffic when using the remote commands, such as `rsh` and `ssh`.
      - By `pam_krb5` to prevent KDC spoofing attacks by using the host principal to verify that a user’s Kerberos credential was obtained from a trusted KDC.
      - To allow the root user to automatically acquire a Kerberos credential without requiring that a root principal exist. This can be useful when doing a manual NFS mount where the share requires a Kerberos credential.

      This principal is required if traffic using the remote application is to be authenticated using the Kerberos service. If the server has multiple hostnames associated with it, then create a principal for each hostname using the FQDN form of the hostname.

      ```bash
      kadmin: addprinc -randkey host/boston.example.com
      Principal "host/boston.example.com" created.
      kadmin:
      ```

   b. **Add the server’s host principal to the server’s keytab.**
      
      If the `kadmin` command is not running, restart it with a command similar to the following:

      ```bash
      /usr/sbin/kadmin -p kws/admin
      ```
If the server has multiple hostnames associated with it, then add a principal to the keytab for each hostname.

```
kadmin: ktadd host/boston.example.com
```

Entry for principal host/boston.example.com with kvno 3, encryption type AES-256 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.

Entry for principal host/boston.example.com with kvno 3, encryption type AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.

Entry for principal host/boston.example.com with kvno 3, encryption type Triple DES cbc mode with HMAC/shal added to keytab WRFILE:/etc/krb5/krb5.keytab.

Entry for principal host/boston.example.com with kvno 3, encryption type ArcFour with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.

Entry for principal host/boston.example.com with kvno 3, encryption type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.

```
kadmin: quit
```

---

**Configuring Kerberos NFS Servers**

NFS services use UNIX user IDs (UIDs) to identify a user and cannot directly use GSS credentials. To translate the credential to a UID, a credential table that maps user credentials to UNIX UIDs might need to be created. See “Mapping GSS Credentials to UNIX Credentials” on page 381 for more information on the default credential mapping. The procedures in this section focus on the tasks that are necessary to configure a Kerberos NFS server, to administer the credential table, and to initiate Kerberos security modes for NFS-mounted file systems. The following task map describes the tasks that are covered in this section.

### TABLE 23–2 Configuring Kerberos NFS Servers (Task Map)

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure a Kerberos NFS server.</td>
<td>Enables a server to share a file system that requires Kerberos authentication.</td>
<td>“How to Configure Kerberos NFS Servers” on page 414</td>
</tr>
<tr>
<td>Create a credential table.</td>
<td>Generates a credential table which can be used to provide mapping from GSS credentials to UNIX user IDs, if the default mapping is not sufficient.</td>
<td>“How to Create a Credential Table” on page 415</td>
</tr>
<tr>
<td>Change the credential table that maps user credentials to UNIX UIDs.</td>
<td>Updates information in the credential table.</td>
<td>“How to Add a Single Entry to the Credential Table” on page 416</td>
</tr>
<tr>
<td>Create credential mappings between two like realms.</td>
<td>Provides instructions on how to map UIDs from one realm to another if the realms share a password file.</td>
<td>“How to Provide Credential Mapping Between Realms” on page 417</td>
</tr>
</tbody>
</table>
TABLE 23–2 Configuring Kerberos NFS Servers (Task Map) (Continued)

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Share a file system with Kerberos authentication.</td>
<td>Shares a file system with security modes so that Kerberos authentication is required.</td>
<td>“How to Set Up a Secure NFS Environment With Multiple Kerberos Security Modes” on page 418</td>
</tr>
</tbody>
</table>

▼ How to Configure Kerberos NFS Servers

In this procedure, the following configuration parameters are used:

- Realm name = EXAMPLE.COM
- DNS domain name = example.com
- NFS server = denver.example.com
- admin principal = kws/admin

1 Complete the prerequisites for configuring a Kerberos NFS server.

The master KDC must be configured. To fully test the process, you need several clients.

2 (Optional) Install the NTP client or another clock synchronization mechanism.

Installing and using the Network Time Protocol (NTP) is not required. However, every clock must be synchronized with the time on the KDC server within a maximum difference defined by the `clockskew` relation in the `krb5.conf` file for authentication to succeed. See “Synchronizing Clocks Between KDCs and Kerberos Clients” on page 437 for information about NTP.

3 Configure the NFS server as a Kerberos client.

Follow the instructions in “Configuring Kerberos Clients” on page 419.

4 Start `kadmin`.

You can use the Graphical Kerberos Administration Tool to add a principal, as explained in “How to Create a New Kerberos Principal” on page 489. To do so, you must log in with one of the `admin` principal names that you created when you configured the master KDC. However, the following example shows how to add the required principals by using the command line.

```
denver # /usr/sbin/kadmin -p kws/admin
Enter password: <Type kws/admin password>
kadmin:
```

a. Create the server’s NFS service principal.

Note that when the principal instance is a host name, the FQDN must be specified in lowercase letters, regardless of the case of the domain name in the `/etc/resolv.conf` file.
Repeat this step for each unique interface on the system that might be used to access NFS data. If a host has multiple interfaces with unique names, each unique name must have its own NFS service principal.

```
    kadmin: addprinc -randkey nfs/denver.example.com
    Principal "nfs/denver.example.com" created.
```

**b. Add the server's NFS service principal to the server's keytab file.**

Repeat this step for each unique service principal created in Step a.

```
    kadmin: ktadd nfs/denver.example.com
    Entry for principal nfs/denver.example.com with kvno 3, encryption type AES-256 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
    Entry for principal nfs/denver.example.com with kvno 3, encryption type AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
    Entry for principal nfs/denver.example.com with kvno 3, encryption type Triple DES cbc mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
    Entry for principal nfs/denver.example.com with kvno 3, encryption type ArcFour with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
    Entry for principal nfs/denver.example.com with kvno 3, encryption type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
```

```
    kadmin: quit
```

**c. Quit kadmin.**

```
    kadmin: quit
```

5  *(Optional) Create special GSS credential maps, if needed.*

Normally, the Kerberos service generates appropriate maps between the GSS credentials and the UNIX UIDs. The default mapping is described in "Mapping GSS Credentials to UNIX Credentials" on page 381. If the default mapping is not sufficient, see "How to Create a Credential Table" on page 415 for more information.

6  Share the NFS file system with Kerberos security modes.

See “How to Set Up a Secure NFS Environment With Multiple Kerberos Security Modes” on page 418 for more information.

▼  **How to Create a Credential Table**

The gsscred credential table is used by an NFS server to map Kerberos credentials to a UID. By default, the primary part of the principal name is matched to a UNIX login name. For NFS clients to mount file systems from an NFS server with Kerberos authentication, this table must be created if the default mapping is not sufficient.
Edit /etc/gss/gsscred.conf and change the security mechanism.
Change the mechanism to files.

Create the credential table by using the gsscred command.

```
# gsscred -m kerberos_v5 -a
```

The gsscred command gathers information from all sources that are listed with the passwd entry in the /etc/nsswitch.conf file. You might need to temporarily remove the files entry, if you do not want the local password entries included in the credential table. See the gsscred(1M) man page for more information.

How to Add a Single Entry to the Credential Table

Before You Begin
This procedure requires that the gsscred table has already been created on the NFS server. See “How to Create a Credential Table” on page 415 for instructions.


2. Add an entry to the credential table by using the gsscred command.

```
# gsscred -m mech [-n name [-u uid ]] -a
```

- **mech**: Defines the security mechanism to be used.
- **name**: Defines the principal name for the user, as defined in the KDC.
- **uid**: Defines the UID for the user, as defined in the password database.
- **-a**: Adds the UID to principal name mapping.

Example 23–3
Adding a Multiple Component Principal to the Credential Table
In the following example, an entry is added for a principal named sandy/admin, which is mapped to UID 3736.

```
# gsscred -m kerberos_v5 -n sandy/admin -u 3736 -a
```

Example 23–4
Adding a Principal in a Different Domain to the Credential Table
In the following example, an entry is added for a principal named sandy/admin@EXAMPLE.COM, which is mapped to UID 3736.

```
# gsscred -m kerberos_v5 -n sandy/admin@EXAMPLE.COM -u 3736 -a
```
How to Provide Credential Mapping Between Realms

This procedure provides appropriate credential mapping between realms that use the same password file. In this example, the realms CORP.EXAMPLE.COM and SALES.EXAMPLE.COM use the same password file. The credentials for bob@CORP.EXAMPLE.COM and bob@SALES.EXAMPLE.COM are mapped to the same UID.

1. Become superuser.

2. On the client system, add entries to the krb5.conf file.

   # cat /etc/krb5/krb5.conf
   [libdefaults]
       default_realm = CORP.EXAMPLE.COM
   .
   [realms]
     CORP.EXAMPLE.COM = {
       .
       auth_to_local_realm = SALES.EXAMPLE.COM
     .
   }

Example 23–5 Mapping Credentials Between Realms Using the Same Password File

This example provides appropriate credential mapping between realms that use the same password file. In this example, the realms CORP.EXAMPLE.COM and SALES.EXAMPLE.COM use the same password file. The credentials for bob@CORP.EXAMPLE.COM and bob@SALES.EXAMPLE.COM are mapped to the same UID. On the client system, add entries to the krb5.conf file.

   # cat /etc/krb5/krb5.conf
   [libdefaults]
       default_realm = CORP.EXAMPLE.COM
   .
   [realms]
     CORP.EXAMPLE.COM = {
       .
       auth_to_local_realm = SALES.EXAMPLE.COM
     .
   }

Troubleshooting See “Observing Mapping from GSS Credentials to UNIX Credentials” on page 477 to help with the process of troubleshooting credential mapping problems.
How to Set Up a Secure NFS Environment With Multiple Kerberos Security Modes

This procedure enables a NFS server to provide secure NFS access using different security modes or flavors. When a client negotiates a security flavor with the NFS server, the first flavor that is offered by the server that the client has access to is used. This flavor is used for all subsequent client requests of the file system shared by the NFS server.

1 Become superuser on the NFS server.

2 Verify that there is an NFS service principal in the keytab file.

   The klist command reports if there is a keytab file and displays the principals. If the results show that no keytab file exists or that no NFS service principal exists, you need to verify the completion of all the steps in “How to Configure Kerberos NFS Servers” on page 414.

   # klist -k
   Keytab name: FILE:/etc/krb5/krb5.keytab
   KVNO Principal
   --- ---------------------------------------------------------
   3 nfs/denver.example.com@EXAMPLE.COM
   3 nfs/denver.example.com@EXAMPLE.COM
   3 nfs/denver.example.com@EXAMPLE.COM
   3 nfs/denver.example.com@EXAMPLE.COM

3 Enable Kerberos security modes in the /etc/nfssec.conf file.

   Edit the /etc/nfssec.conf file and remove the “#” that is placed in front of the Kerberos security modes.

   # cat /etc/nfssec.conf
   #
   # Uncomment the following lines to use Kerberos V5 with NFS
   #
   kb5 390003 kerberos_v5 default - # RPCSEC_GSS
   krb5i 390004 kerberos_v5 default integrity # RPCSEC_GSS
   krb5p 390005 kerberos_v5 default privacy # RPCSEC_GSS

4 Share the file systems with the appropriate security modes.

   sharemgr share -o sec=mode file-system

   mode Specifies the security modes to be used when sharing the file system. When using multiple security modes, the first mode in the list is used as the default.

   file-system Defines the path to the file system to be shared.
All clients that attempt to access files from the named file system require Kerberos authentication. To access files, the user principal on the NFS client should be authenticated.

5  (Optional) If the automounter is being used, edit the auto_master database to select a security mode other than the default.
You need not follow this procedure if you are not using the automounter to access the file system or if the default selection for the security mode is acceptable.

```
file-system auto_home -nosuid,sec=mode
```

6  (Optional) Manually issue the mount command to access the file system by using a non-default mode.
Alternatively, you could use the mount command to specify the security mode, but this alternative does not take advantage of the automounter.

```
# mount -F nfs -o sec=mode file-system
```

Example 23–6 Sharing a File System With One Kerberos Security Mode
In this example, Kerberos authentication must succeed before any files can be accessed through the NFS service.

```
# sharemgr share -F nfs -p -o sec=krb5 /export/home
```

Example 23–7 Sharing a File System With Multiple Kerberos Security Modes
In this example, all three Kerberos security modes have been selected. Which mode is used is negotiated between the client and the NFS server. If the first mode in the command fails, then the next is tried. See the nfssec(5) man page for more information.

```
# sharemgr share -F nfs -p -o sec=krb5:krb5i:krb5p /export/home
```

**Configuring Kerberos Clients**

Kerberos clients include any host, that is not a KDC server, on the network that needs to use Kerberos services. This section provides procedures for installing a Kerberos client, as well as specific information about using root authentication to mount NFS file systems.
# Configuring Kerberos Clients (Task Map)

The following task map includes all of the procedures associated with setting up Kerberos clients. Each row includes a task identifier, a description of why you would want to do that task, followed by a link to the task.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Establish a Kerberos client installation profile.</td>
<td>Generates a client installation profile that can be used to automatically install a Kerberos client.</td>
<td>&quot;How to Create a Kerberos Client Installation Profile&quot; on page 420</td>
</tr>
<tr>
<td>Configure a Kerberos client.</td>
<td>Manually installs a Kerberos client. Use this procedure if each client installation requires unique installation parameters.</td>
<td>&quot;How to Manually Configure a Kerberos Client&quot; on page 427</td>
</tr>
<tr>
<td></td>
<td>Automatically installs a Kerberos client. Use this procedure if the installation parameters for each client are the same.</td>
<td>&quot;How to Automatically Configure a Kerberos Client&quot; on page 421</td>
</tr>
<tr>
<td></td>
<td>Interactively installs a Kerberos client. Use this procedure if only a few of the installation parameters need to change.</td>
<td>&quot;How to Interactively Configure a Kerberos Client&quot; on page 422</td>
</tr>
<tr>
<td></td>
<td>Automatically installs a Kerberos client of an Active Directory server.</td>
<td>&quot;How to Configure a Kerberos Client for an Active Directory Server&quot; on page 426</td>
</tr>
<tr>
<td>Allow a client to access a NFS file system as the root user</td>
<td>Creates a root principal on the client, so that the client can mount a NFS file system shared with root access. Also, allows for the client to set up non-interactive root access to the NFS file system, so that cron jobs can run.</td>
<td>&quot;How to Access a Kerberos Protected NFS File System as the root User&quot; on page 433</td>
</tr>
<tr>
<td>Disable verification of the KDC that issued a client Ticket Granting Ticket (TGT).</td>
<td>Allows clients that do not have a host principal stored in the local keytab file to skip the security check that verifies that the KDC that issued the TGT is the same server that issued the host principal.</td>
<td>&quot;How to Disable Verification of the Ticket Granting Ticket (TGT)&quot; on page 432</td>
</tr>
</tbody>
</table>

## How to Create a Kerberos Client Installation Profile

This procedure creates a kclient profile that can be used when you install a Kerberos client. By using the kclient profile, you reduce the likelihood of typing errors. Also, using the profile reduces user intervention as compared to the interactive process.

1. Become superuser.
2 Create a kclient installation profile.
   A sample kclient profile could look similar to the following:
   
   client# cat /net/denver.example.com/export/install/profile
   REALM EXAMPLE.COM
   KDC kdc1.example.com
   ADMIN clntconfig
   FILEPATH /net/denver.example.com/export/install/krb5.conf
   NFS 1
   DNSLOOKUP none

▼ How to Automatically Configure a Kerberos Client

Before You Begin
   This procedure uses an installation profile. See “How to Create a Kerberos Client Installation Profile” on page 420.

1 Become superuser.

2 Run the kclient installation script.
   You need to provide the password for the clntconfig principal to complete the process.
   
   client# /usr/sbin/kclient -p /net/denver.example.com/export/install/profile

   Starting client setup
   --------------------------------------------

   kdc1.example.com

   Setting up /etc/krb5/krb5.conf.

   Obtaining TGT for clntconfig/admin ...
   Password for clntconfig/admin@EXAMPLE.COM: <Type the password>

   nfs/client.example.com entry ADDED to KDC database.
   nfs/client.example.com entry ADDED to keytab.

   host/client.example.com entry ADDED to KDC database.
   host/client.example.com entry ADDED to keytab.

   Copied /net/denver.example.com/export/install/krb5.conf.

   --------------------------------------------
   Setup COMPLETE.

   client#
Example 23–8  Automatically Configuring a Kerberos Client With Command-Line Overrides

The following example overrides the DNSARG and the KDC parameters that are set in the installation profile.

```
# /usr/sbin/kclient -p /net/denver.example.com/export/install/profile\
   -d dns_fallback -k kdc2.example.com
```

Starting client setup

```
kdc1.example.com
```

Setting up /etc/krb5/krb5.conf.

```
Obtaining TGT for clntconfig/admin ...
Password for clntconfig/admin@EXAMPLE.COM: <Type the password>
```

```
nfs/client.example.com entry ADDED to KDC database.
nfs/client.example.com entry ADDED to keytab.
```

```
host/client.example.com entry ADDED to KDC database.
host/client.example.com entry ADDED to keytab.
```

```
Copied /net/denver.example.com/export/install/krb5.conf.
```

```
Setup COMPLETE.
```

client#

▼ How to Interactively Configure a Kerberos Client

This procedure uses the kclient installation utility without an installation profile. In build 90 of the Solaris Express Community Edition release, the kclient utility has been expanded to improve ease of use and ability to work with Active Directory servers. See “How to Configure a Kerberos Client for an Active Directory Server” on page 426 for more information. See Example 23–10 for an example of running kclient on a previous Solaris release.

1  Become superuser.

2  Run the kclient installation script.

   You need to provide the following information:

   - Kerberos realm name
   - KDC master host name
- KDC slave host names
- Domains to map to the local realm
- PAM service names and options to use for Kerberos authentication

a. **Indicate if the KDC server is not running a Solaris release.**
If this system is a client of a KDC server that is not running a Solaris release, you need to define the type of server that is running the KDC. The available servers are: Microsoft Active Directory, MIT KDC server, Heimdal KDC server, and Shishi KDC server.

b. **Select if DNS should be used for Kerberos lookups.**
If you use DNS for Kerberos lookups, you need to enter the DNS lookup option that you want to use. Valid options are `dns_lookup_kdc`, `dns_lookup_realm`, and `dns_fallback`. See the `krb5.conf(4)` man page for more information about these values.

c. **Define the name of the Kerberos realm and the master KDC hostname.**
This information is added to the `/etc/krb5/krb5.conf` configuration file.

d. **Select if slave KDCs exist.**
If there are slave KDCs in the realm, then you need to enter the slave KDC host names. This information is used to create additional KDC entries in the client's configuration file.

e. **Indicate if service or host keys are required.**
Normally, service or host keys are not required unless the client system is hosting Kerberized services.

f. **Specify if the client is a member of a cluster.**
If the client is a member of a cluster, then you need to provide the logical name of the cluster. The logical host name is used when creating service keys, which is required when hosting Kerberos services from clusters.

g. **Identify any domains or hosts to map to the current realm.**
This mapping allows other domains to belong in the default realm of the client.

h. **Specify if the client will use Kerberized NFS.**
NFS service keys need to be created if the client will host NFS services using Kerberos.

i. **Indicate if the `/etc/pam.conf` file needs to be updated.**
This allows you to set which PAM services use Kerberos for authentication. You need to enter the service name and a flag indicating how Kerberos authentication is to be used. The valid flag options are:

- **first** — use Kerberos authentication first, and only use UNIX if Kerberos authentication fails
- **only** — use Kerberos authentication only
- **optional** — use Kerberos authentication optionally

### j. **Select if the master /etc/krb5/krb5.conf file should be copied.**

This option allows for specific configuration information to be used when the arguments to `kclient` are not sufficient.

---

**Example 23–9  Running the kclient Installation Utility**

```
client# /usr/sbin/kclient
```

Starting client setup
```

Is this a client of a non-Solaris KDC ? [y/n]: n
No action performed.

Do you want to use DNS for kerberos lookups ? [y/n]: n
No action performed.

Enter the Kerberos realm: **EXAMPLE.COM**

Specify the KDC hostname for the above realm: **kdc1.example.com**

Note, this system and the KDC’s time must be within 5 minutes of each other for Kerberos to function. Both systems should run some form of time synchronization system like Network Time Protocol (NTP).

Do you have any slave KDC(s) ? [y/n]: y
Enter a comma-separated list of slave KDC host names: **kdc2.example.com**

Will this client need service keys ? [y/n]: n
No action performed.

Is this client a member of a cluster that uses a logical host name ? [y/n]: n
No action performed.

Do you have multiple domains/hosts to map to realm ? [y/n]: y
Enter a comma-separated list of domain/hosts to map to the default realm: **engineering.example.com, example.com**

Setting up /etc/krb5/krb5.conf.

Do you plan on doing Kerberized nfs ? [y/n]: y
Do you want to update /etc/pam.conf ? [y/n]: y
Enter a comma-separated list of PAM service names in the following format:
Configuring Kerberos Clients

Example 23–10  Running the kclient Installation Utility on the Solaris 10 Release

The following output shows the results of running the kclient command.

client# /usr/sbin/kclient

Starting client setup

Do you want to use DNS for kerberos lookups ? [y/n]: n
No action performed.

Enter the Kerberos realm: EXAMPLE.COM
Specify the KDC hostname for the above realm: kdc1.example.com

Setting up /etc/krb5/krb5.conf.

Enter the krb5 administrative principal to be used: clntconfig/admin
Obtaining TGT for clntconfig/admin ...
Password for clntconfig/admin@EXAMPLE.COM: <Type the password>

Do you plan on doing Kerberized nfs ? [y/n]: n

host/client.example.com entry ADDED to KDC database.
host/client.example.com entry ADDED to keytab.

Do you want to copy over the master krb5.conf file ? [y/n]: y
Enter the pathname of the file to be copied: \net/denver.example.com/export/install/krb5.conf

Copied /net/denver.example.com/export/install/krb5.conf.

Setup COMPLETE!
How to Configure a Kerberos Client for an Active Directory Server

This procedure uses the kclient installation utility without a installation profile.

1. Become superuser.

2. (Optional) Enable DNS resource record creation for the client.
   The first command causes DNS records to be created when the client is joined to the Active Directory domain. The second command records this change in the running configuration of this service.
   ```
   client# svccfg -s smb/server setprop smbd/ddns_enable=true
   client# svcadm refresh smb/server
   ```

3. Run the kclient installation script.
   You need to provide the following information:
   - Password for the administrative principal

Example 23–11 Configuring a Kerberos Client for an Active Directory Server Using kclient.

The following output shows the results of running the kclient command using the ms_ad (Microsoft Active Directory) server type argument. The client will be joined to the Active Directory domain called EXAMPLE.COM.

```
client# /usr/sbin/kclient -T ms_ad

Starting client setup
-----------------------------------------------

Attempting to join 'CLIENT' to the 'EXAMPLE.COM' domain.
Password for Administrator@EXAMPLE.COM:   <Type the password>
Forest name found: example.com
Looking for local KDCs, DCs and global catalog servers (SVR RRs).

Setting up /etc/krb5/krb5.conf

Creating the machine account in AD via LDAP.
-----------------------------------------------

Setup COMPLETE.
#
```
How to Manually Configure a Kerberos Client

In this procedure, the following configuration parameters are used:

- Realm name = EXAMPLE.COM
- DNS domain name = example.com
- Master KDC = kdc1.example.com
- Slave KDC = kdc2.example.com
- NFS server = denver.example.com
- Client = client.example.com
- admin principal = kws/admin
- User principal = mre

Note – Adjust the URL to point to the “Graphical Kerberos Administration Tool” section, as described in the “Online Help URL in the Graphical Kerberos Administration Tool” on page 385.

1. Become superuser.

2. Edit the Kerberos configuration file (krb5.conf).

To change the file from the Kerberos default version, you need to change the realm names and the server names. You also need to identify the path to the help files for gkadmin.

```
kdc1 # cat /etc/krb5/krb5.conf
[libdefaults]
  default_realm = EXAMPLE.COM

[realms]
  EXAMPLE.COM = {
    kdc = kdc1.example.com
    kdc = kdc2.example.com
    admin_server = kdc1.example.com
  }

[domain_realm]
  .example.com = EXAMPLE.COM
  #
  # if the domain name and realm name are equivalent,
  # this entry is not needed
  #
  [logging]
  default = FILE:/var/krb5/kdc.log
```
kdc = FILE:/var/krb5/kdc.log

[appdefaults]
gkadmin = {
}

Note – If you want to restrict the encryption types, you can set the default_tkt_enctypes or default_tgs_enctypes lines. Refer to “Using Kerberos Encryption Types” on page 545 for a description of the issues involved with restricting the encryption types.

3 (Optional) Change the process used to locate the KDCs.

Starting with the Solaris 10 5/08 and Solaris Express Developer Edition 1/08 releases, by default the Kerberos realm to KDC mapping is determined in the following order:

- The definition in the realms section in krb5.conf.
- By looking up SRV records in DNS.

You can change this behavior by adding dns_lookup_kdc or dns_fallback to the libdefaults section of the krb5.conf file. See the krb5.conf(4) man page for more information. Note that referrals are always tried first.

4 (Optional) Change the process used to determine the realm for a host.

Starting with the Solaris 10 5/08 and Solaris Express Developer Edition 1/08 releases, by default the host to realm mapping is determined in the following order:

- If the KDC supports referrals, then the KDC may inform the client which realm the host belongs to.
- By the definition of domain_realm in the krb5.conf file.
- The DNS domain name of the host.
- The default realm.

You can change this behavior by adding dns_lookup_kdc or dns_fallback to the libdefaults section of the krb5.conf file. See the krb5.conf(4) man page for more information. Note that referrals will always be tried first.

5 (Optional) Synchronize the client’s clock with the master KDC’s clock by using NTP or another clock synchronization mechanism.

Installing and using the Network Time Protocol (NTP) is not required. However, every clock must be synchronized with the time on the KDC server within a maximum difference defined in the clockskeW relation in the krb5.conf file for authentication to succeed. See “Synchronizing Clocks Between KDCs and Kerberos Clients” on page 437 for information about NTP.
6 Start kadmin.

You can use the Graphical Kerberos Administration Tool to add a principal, as explained in “How to Create a New Kerberos Principal” on page 489. To do so, you must log in with one of the admin principal names that you created when you configured the master KDC. However, the following example shows how to add the required principals by using the command line.

denver # /usr/sbin/kadmin -p kws/admin
Enter password:  "Type kws/admin password"

a. (Optional) Create a user principal if a user principal does not already exist.

You need to create a user principal only if the user associated with this host does not already have a principal assigned to him or her.

kadmin: addprinc mre
Enter password for principal mre@example.com:  "Type the password"
Re-enter password for principal mre@example.com:  "Type it again"

b. (Optional) Create a root principal and add the principal to the server’s keytab file.

This step is required so that the client can have root access to file systems mounted using the NFS service. This step is also required if non-interactive root access is needed, such as running cron jobs as root.

If the client does not require root access to a remote file system which is mounted using the NFS service, then you can skip this step. The root principal should be a two component principal with the second component the host name of the Kerberos client system to avoid the creation of a realm wide root principal. Note that when the principal instance is a host name, the FQDN must be specified in lowercase letters, regardless of the case of the domain name in the /etc/resolv.conf file.

kadmin: addprinc -randkey root/client.example.com
Principal "root/client.example.com" created.

kadmin: ktadd root/client.example.com
Entry for principal root/client.example.com with kvno 3, encryption type AES-256 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type Triple DES cbc mode with HMAC/shal added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type ArcFour with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.

kadmin:
c. Create a host principal and add the principal to the server's keytab file.

The host principal is used by remote access services to provide authentication. The principal allows root to acquire a credential, if there is not one already in the keytab file.

\[ \text{kadmin: addprinc -randkey host/denver.example.com} \]
Principal "host/denver.example.com@EXAMPLE.COM" created.

\[ \text{kadmin: ktadd host/denver.example.com} \]
Entry for principal host/denver.example.com with kvno 3, encryption type AES-256 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/denver.example.com with kvno 3, encryption type AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/denver.example.com with kvno 3, encryption type Triple DES cbc mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/denver.example.com with kvno 3, encryption type ArcFour with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/denver.example.com with kvno 3, encryption type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.

\[ \text{kadmin:} \]

d. (Optional) Add the server’s NFS service principal to the server’s keytab file.

This step is only required if the client needs to access NFS file systems using Kerberos authentication.

\[ \text{kadmin: ktadd nfs/denver.example.com} \]
Entry for principal nfs/denver.example.com with kvno 3, encryption type AES-256 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal nfs/denver.example.com with kvno 3, encryption type AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal nfs/denver.example.com with kvno 3, encryption type Triple DES cbc mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal nfs/denver.example.com with kvno 3, encryption type ArcFour with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal nfs/denver.example.com with kvno 3, encryption type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.

\[ \text{kadmin:} \]
e. Quit kadmin.

\[ \text{kadmin: quit} \]

7 (Optional) Enable Kerberos with NFS.

a. Enable Kerberos security modes in the /etc/nfssec.conf file.

Edit the /etc/nfssec.conf file and remove the "#" that is placed in front of the Kerberos security modes.

\[ \text{# cat /etc/nfssec.conf} \]

\[ \text{.} \]
# Uncomment the following lines to use Kerberos V5 with NFS
#
krb5 390003 kerberos_v5 default - # RPCSEC_GSS
krb5i 390004 kerberos_v5 default integrity # RPCSEC_GSS
krb5p 390005 kerberos_v5 default privacy # RPCSEC_GSS

b. Enable DNS.
If the /etc/resolv.conf file has not already been created, then create this file as the service principal canonicalization is dependent upon DNS to do this. See the resolv.conf(4) man page for more information.

c. Restart the gssd service.
After the /etc/resolv.conf file has been created or modified you must then restart the gssd daemon to reread any changes.

# svcadm restart network/rpc/gss

8 If you want the client to automatically renew the TGT or to warn users about Kerberos ticket expiration, create an entry in the /etc/krb5/warn.conf file.
See the warn.conf(4) man page for more information.

Example 23-12 Setting Up a Kerberos Client Using a Non-Solaris KDC
A Kerberos client can be set up to work with a non-Solaris KDC. In this case, a line must be included in the /etc/krb5/krb5.conf file in the realms section. This line changes the protocol that is used when the client is communicating with the Kerberos password-changing server. The format of this line follows.

[realms]

EXAMPLE.COM = {
  kdc = kdc1.example.com
  kdc = kdc2.example.com
  admin_server = kdc1.example.com
  kpasswd_protocol = SET_CHANGE
}

Example 23-13 DNS TXT Records for the Mapping of Host and Domain Name to Kerberos Realm

@ IN SOA kdc1.example.com root.kdc1.example.com ( 1989020501 ;serial 10800 ;refresh 3600 ;retry 3600000 ;expire 86400 ) ;minimum
**Example 23–14  DNS SRV Records for Kerberos Server Locations**

This example defines the records for the location of the KDCs, the admin server, and the kpasswd server, respectively.

```plaintext
@ IN SOA kdc1.example.com root.kdc1.example.com ( 1989020501 ;serial 10800 ;refresh 3600 ;retry 3600000 ;expire 86400 ) ;minimum
kdc1 IN A 192.146.86.20
kdc2 IN A 192.146.86.21
_kerberos.example.com. IN TXT "EXAMPLE.COM"
_kerberos.kdc1.example.com. IN TXT "EXAMPLE.COM"
_kerberos.kdc2.example.com. IN TXT "EXAMPLE.COM"
```

▼ **How to Disable Verification of the Ticket Granting Ticket (TGT)**

This procedure disables the security check that checks that the KDC of the host principal stored in the local `/etc/krb5/krb5.keytab` file is the same KDC that issued the Ticket Granting Ticket. This check prevents DNS spoofing attacks. However, for some client configurations, the host principal may not be available, so this check would need to be disabled to allow the client to function. These are the configurations that require that this check is disabled:

- The client IP address is dynamically assigned. For instance, a DHCP client.
- The client is not configured to host any services, so no host principal was created.
- The host key is not stored on the client.

1. Become superuser.
Change the krb5.conf file.

If the verify_ap_req_nofail option is set to false, the TGT verification process is not enabled. See the krb5.conf(4) man page for more information about this option.

```
client # cat /etc/krb5/krb5.conf
[libdefaults]
default_realm = EXAMPLE.COM
verify_ap_req_nofail = false
...
```

Note – The verify_ap_req_nofail option can be entered in either the [libdefaults] or the [realms] section of the krb5.conf file. If the option is in the [libdefaults] section, the setting is used for all realms. If the option is in the [realms] section, the setting only applies to the defined realm.

How to Access a Kerberos Protected NFS File System as the root User

This procedure allows a client to access a NFS file system that requires Kerberos authentication with the root ID privilege. In particular, when the NFS file system is shared with options like: `-o sec=krb5,root=client1.sun.com`

1. Become superuser.

2. Start kadmin.

You can use the Graphical Kerberos Administration Tool to add a principal, as explained in “How to Create a New Kerberos Principal” on page 489. To do so, you must log in with one of the admin principal names that you created when you configured the master KDC. However, the following example shows how to add the required principals by using the command line.

```
denver # /usr/sbin/kadmin -p kws/admin
Enter password: <Type kws/admin password>
kadmin:
```

a. Create a root principal for the NFS client.

This principal is used to provide root equivalent access to NFS mounted file systems that require Kerberos authentication. The root principal should be a two component principal with the second component the host name of the Kerberos client system to avoid the creation of a realm wide root principal. Note that when the principal instance is a host name, the FQDN must be specified in lowercase letters, regardless of the case of the domain name in the /etc/resolv.conf file.

```
kadmin: addprinc -randkey root/client.example.com
Principal "root/client.example.com" created.
kadmin:
```
b. Add the root principal to the server's keytab file.

This step is required if you added a root principal so that the client can have root access to file systems mounted using the NFS service. This step is also required if non-interactive root access is needed, such as running cron jobs as root.

```bash
kadmin: ktadd root/client.example.com
Entry for principal root/client.example.com with kvno 3, encryption type AES-256 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type Triple DES cbc mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type ArcFour with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
```

kadmin:

c. Quit kadmin.

```bash
kadmin: quit
```

▼ How to Configure Automatic Migration of Users in a Kerberos Realm

Users, who do not have a Kerberos principal, can be automatically migrated to an existing Kerberos realm. The migration is achieved by using the PAM framework for the service in use by stacking the `pam_krb5_migrate` module in the service's authentication stack in `/etc/pam.conf`.

In this example, the `dtlogin` and other PAM service names are configured to use the automatic migration. The following configuration parameters are used:

- Realm name = EXAMPLE.COM
- Master KDC = kdc1.example.com
- Machine hosting the migration service = server1.example.com
- Migration service principal = host/server1.example.com

**Before You Begin** Setup server1 as a Kerberos client of the realm EXAMPLE.COM. See “Configuring Kerberos Clients” on page 419 for more information.
1 Check to see if a host service principal for `server1` exists.

The host service principal in the keytab file of `server1` is used to authenticate the server to the master KDC.

```
server1 # klist -k
Keytab name: FILE:/etc/krb5/krb5.keytab

KVNO Principal
-----------------------------------------------
  3 host/server1.example.com@EXAMPLE.COM
  3 host/server1.example.com@EXAMPLE.COM
  3 host/server1.example.com@EXAMPLE.COM
  3 host/server1.example.com@EXAMPLE.COM
```

2 Make changes to the PAM configuration file.

   a. Add entries for the `dtlogin` service.

```
# cat /etc/pam.conf
.
.
#
# dtlogin service (explicit because of pam_krb5_migrate)
#
dtlogin auth requisite       pam_authtok_get.so.1
dtlogin auth required        pam_dhkeys.so.1
dtlogin auth required        pam_unix_cred.so.1
dtlogin auth sufficient      pam_krb5.so.1
dtlogin auth requisite       pam_unix_auth.so.1
dtlogin auth optional        pam_krb5_migrate.so.1
```

   b. (Optional) Force an immediate password change, if needed.

   The newly created Kerberos accounts can have their password expiration time set to the current time (now), in order to force an immediate Kerberos password change. To set the expiration time to now, add the `expire_pw` option to the lines which use the `pam_krb5_migrate` module. See the `pam_krb5_migrate(5)` man page for more information.

```
# cat /etc/pam.conf
.
.
dtlogin auth optional        pam_krb5_migrate.so.1 expire_pw
```

   c. Add the `pam_krb5` module to the account stack.

   This addition allows for password expiration in Kerberos to block access.

```
# cat /etc/pam.conf
.
.
#
# Default definition for Account management
d. **Add the pam_krb5 module to the password stack.**
This addition allows for passwords to be updated when the password expire.

```bash
# cat /etc/pam.conf
.
.
# Default definition for Password management
# Used when service name is not explicitly mentioned for password management
#
other password required pam_dhkeys.so.1
other password requisite pam_authtok_get.so.1
other password requisite pam_authtok_check.so.1
other password sufficient pam_krb5.so.1
other password required pam_authtok_store.so.1
```

3  **On the master KDC, update the access control file.**
The following entries grant migrate and inquire privileges to the host/server1.example.com service principal for all users, excepting the root user. It is important that users who should not be migrated are listed in the kadm5.acl file using the U privilege. These entries need to be before the permit all or ui entry. See the `kadm5.acl(4)` man page for more information.

```bash
kdc1 # cat /etc/krb5/kadm5.acl
host/server1.example.com@EXAMPLE.COM U root
host/server1.example.com@EXAMPLE.COM ui *
*/admin@EXAMPLE.COM *
```

4  **On the master KDC, restart the Kerberos administration daemon.**
This step allows the kadmin daemon to use the new kadm5.acl entries.

```bash
kdc1 # svcadm restart network/security/kadmin
```

5  **On the master KDC, add entries to the pam.conf file.**
The following entries enable the kadmin daemon to use the k5migrate PAM service, to validate UNIX user password for accounts that require migration.

```bash
# grep k5migrate /etc/pam.conf
k5migrate auth required pam_unix_auth.so.1
k5migrate account required pam_unix_account.so.1
```
Synchronizing Clocks Between KDCs and Kerberos Clients

All hosts that participate in the Kerberos authentication system must have their internal clocks synchronized within a specified maximum amount of time (known as clock skew). This requirement provides another Kerberos security check. If the clock skew is exceeded between any of the participating hosts, client requests are rejected.

The clock skew also determines how long application servers must keep track of all Kerberos protocol messages, in order to recognize and reject replayed requests. So, the longer the clock skew value, the more information that application servers have to collect.

The default value for the maximum clock skew is 300 seconds (five minutes). You can change this default in the `libdefaults` section of the `krb5.conf` file.

**Note** – For security reasons, do not increase the clock skew beyond 300 seconds.

Because maintaining synchronized clocks between the KDCs and Kerberos clients is important, you should use the Network Time Protocol (NTP) software to synchronize them. NTP public domain software from the University of Delaware is included in the Solaris software, starting with the Solaris 2.6 release.

**Note** – Another way to synchronize clocks is to use the `rdate` command and `cron` jobs, a process that can be less involved than using NTP. However, this section focuses on using NTP. And, if you use the network to synchronize the clocks, the clock synchronization protocol must itself be secure.

NTP enables you to manage precise time or network clock synchronization, or both, in a network environment. NTP is basically a server-client implementation. You pick one system to be the master clock (the NTP server). Then, you set up all your other systems (the NTP clients) to synchronize their clocks with the master clock.

To synchronize the clocks, NTP uses the `xntpd` daemon, which sets and maintains a UNIX system time-of-day in agreement with Internet standard time servers. The following shows an example of this server-client NTP implementation.
Ensuring that the KDCs and Kerberos clients maintain synchronized clocks involves implementing the following steps:

1. Setting up an NTP server on your network. This server can be any system, except the master KDC. See “Managing Network Time Protocol (Tasks)” in System Administration Guide: Network Services to find the NTP server task.

2. As you configure the KDCs and Kerberos clients on the network, setting them up to be NTP clients of the NTP server. See “Managing Network Time Protocol (Tasks)” in System Administration Guide: Network Services to find the NTP client task.

### Swapping a Master KDC and a Slave KDC

You should use the procedures in this section to make the swap of a master KDC with a slave KDC easier. You should swap the master KDC with a slave KDC only if the master KDC server fails for some reason, or if the master KDC needs to be re-installed (for example, because new hardware is installed).

#### How to Configure a Swappable Slave KDC

Perform this procedure on the slave KDC server that you want to have available to become the master KDC. This procedure assumes that you are using incremental propagation.
1 **Use alias names for the master KDC and the swappable slave KDC during the KDC installation.**
   When you define the host names for the KDCs, make sure that each system has an alias included in DNS. Also, use the alias names when you define the hosts in the `/etc/krb5/krb5.conf` file.

2 **Follow the steps to install a slave KDC.**
   Prior to any swap, this server should function as any other slave KDC in the realm. See “How to Manually Configure a Slave KDC” on page 404 for instructions.

3 **Move the master KDC commands.**
   To prevent the master KDC commands from being run from this slave KDC, move the `kprop`, `kadmind`, and `kadmin.local` commands to a reserved place.

   ```
   kdc4 # mv /usr/lib/krb5/kprop /usr/lib/krb5/kprop.save
   kdc4 # mv /usr/lib/krb5/kadmind /usr/lib/krb5/kadmind.save
   kdc4 # mv /usr/sbin/kadmin.local /usr/sbin/kadmin.local.save
   ```

### How to Swap a Master KDC and a Slave KDC

In this procedure, the master KDC server that is being swapped out is named `kdc1`. The slave KDC that will become the new master KDC is named `kdc4`. This procedure assumes that you are using incremental propagation.

**Before You Begin**
This procedure requires that the slave KDC server has been set up as a swappable slave. For more information, see “How to Configure a Swappable Slave KDC” on page 438).

1 **On the new master KDC, start `kadmin`.**

   ```
   kdc4 # /usr/sbin/kadmin -p kws/admin
   Enter password: <Type kws/admin password>
   kadmin:
   ```

   a. **Create new principals for the `kadmind` service.**

      The following example shows the first `addprinc` command on two lines, but it should be typed on one line.

      ```
      kadmin: addprinc -randkey -allow_tgs_req +password_changing_service -clearpolicy \
       changepw/kdc4.example.com
      Principal "changepw/kdc4.example.com@ENG.SUN.COM" created.
      kadmin: addprinc -randkey -allow_tgs_req -clearpolicy kadmin/kdc4.example.com
      Principal "kadmin/kdc4.example.com@EXAMPLE.COM" created.
      kadmin:
      ```

   b. **Quit `kadmin`.**

      ```
      kadmin: quit
      ```
2 On the new master KDC, force synchronization.
The following steps force a full KDC update on the slave server.
```
kdc4 # svcadm disable network/security/krb5kdc
kdc4 # rm /var/krb5/principal.ulog
```

3 On the new master KDC, verify that the update is complete.
```
kdc4 # /usr/sbin/kproplog -h
```

4 On the new master KDC, restart the KDC service.
```
kdc4 # svcadm enable -r network/security/krb5kdc
```

5 On the new master KDC, clear the update log.
These steps reinitialize the update log for the new master KDC server.
```
kdc4 # svcadm disable network/security/krb5kdc
kdc4 # rm /var/krb5/principal.ulog
```

6 On the old master KDC, kill the kadmind and krb5kdc processes.
When you kill the kadmind process, you prevent any changes from being made to the KDC database.
```
kdc1 # svcadm disable network/security/kadmin
kdc1 # svcadm disable network/security/krb5kdc
```

7 On the old master KDC, specify the poll time for requesting propagations.
Comment out the sunw_dbprop_master_ulogsize entry in /etc/krb5/kdc.conf and add an entry defining sunw_dbprop_slave_poll. The entry sets the poll time to 2 minutes.
```
kdc1 # cat /etc/krb5/kdc.conf
[kdcdefaults]
kdc_ports = 88,750
[realms]
EXAMPLE.COM= {
    profile = /etc/krb5/krb5.conf
    database_name = /var/krb5/principal
    acl_file = /etc/krb5/kadm5.acl
    kadmind_port = 749
    max_life = 8h 0m 0s
    max_renewable_life = 7d 0h 0m 0s
    sunw_dbprop_enable = true
    #
    sunw_dbprop_master_ulogsize = 1000
    sunw_dbprop_slave_poll = 2m
    }
8 On the old master KDC, move the master KDC commands and the \textit{kadm5.acl} file.
To prevent the master KDC commands from being run, move the kprop, kadmind, and
cadmin.local commands to a reserved place.
\begin{verbatim}
  kdc1 # mv /usr/lib/krb5/kprop /usr/lib/krb5/kprop.save
  kdc1 # mv /usr/lib/krb5/kadmind /usr/lib/krb5/kadmind.save
  kdc1 # mv /usr/sbin/kadmin.local /usr/sbin/kadmin.local.save
  kdc1 # mv /etc/krb5/kadm5.acl /etc/krb5/kadm5.acl.save
\end{verbatim}

9 On the DNS server, change the alias names for the master KDC.
To change the servers, edit the \texttt{example.com} zone file and change the entry for masterkdc.
\begin{verbatim}
masterkdc IN CNAME kdc4
\end{verbatim}

10 On the DNS server, restart the Internet domain name server.
Run the following command to reload the new alias information:
\begin{verbatim}
# svcadm refresh network/dns/server
\end{verbatim}

11 On the new master KDC, move the master KDC commands and the slave \textit{kpropd.acl} file.
\begin{verbatim}
kdc4 # mv /usr/lib/krb5/kprop.save /usr/lib/krb5/kprop
kdc4 # mv /usr/lib/krb5/kadmind.save /usr/lib/krb5/kadmind
kdc4 # mv /usr/sbin/kadmin.local.save /usr/sbin/kadmin.local
kdc4 # mv /etc/krb5/kpropd.acl /etc/krb5/kpropd.acl.save
\end{verbatim}

12 On the new master KDC, create the Kerberos access control list file (\textit{kadm5.acl}).
Once populated, the /etc/krb5/kadm5.acl file should contain all principal names that are allowed to administer the KDC. The file should also list all of the slaves that make requests for incremental propagation. See the \texttt{kadm5.acl(4)} man page for more information.
\begin{verbatim}
kdc4 # cat /etc/krb5/kadm5.acl
kws/admin@EXAMPLE.COM * 
kprop/kdc1.example.com@EXAMPLE.COM p
\end{verbatim}

13 On the new master KDC, specify the update log size in the \textit{kdc.conf} file.
Comment out the \texttt{sunw_dprop_slave_poll} entry and add an entry defining
\texttt{sunw_dprop_master_ulogsiz}. The entry sets the log size to 1000 entries.
\begin{verbatim}
kdc1 # cat /etc/krb5/kdc.conf
[kdcdefaults]
kdc_ports = 88,750
[realms]
EXAMPLE.COM= {
  profile = /etc/krb5/krb5.conf
  database_name = /var/krb5/principal
  acl_file = /etc/krb5/kadm5.acl
  kadmind_port = 749
  max_life = 8h 0m 0s
}\end{verbatim}
max_renewable_life = 7d 0h 0m 0s
sunw_dbprop_enable = true
sunw_dbprop_slave_poll = 2m
sunw_dbprop_master_ulongsize = 1000
}

14 On the new master KDC, start kadmind and krb5kdc.
   kdc4 # svcadm enable -r network/security/krb5kdc
   kdc4 # svcadm enable -r network/security/kadmin

15 On the old master KDC, add the kprop service principal.
   Adding the kprop principal to the krb5.keytab file allows the kpropd daemon to authenticate itself for the incremental propagation service.

   kdc1 # /usr/sbin/kadmin -p kws/admin
   Authenticating as principal kws/admin@EXAMPLE.COM with password.
   Enter password:  <Type kws/admin password>
   kadmin: ktadd kprop/kdc1.example.com
   Entry for principal kprop/kdc1.example.com with kvno 3, encryption type AES-256 CTS mode
   with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
   Entry for principal kprop/kdc1.example.com with kvno 3, encryption type AES-128 CTS mode
   with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
   Entry for principal kprop/kdc1.example.com with kvno 3, encryption type Triple DES cbc
   mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
   Entry for principal kprop/kdc1.example.com with kvno 3, encryption type ArcFour
   with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
   Entry for principal kprop/kdc1.example.com with kvno 3, encryption type DES cbc mode
   with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
   kadmin: quit

16 On the old master KDC, add an entry for each KDC listed in krb5.conf to the propagation configuration file, kpropd.acl.
   kdc1 # cat /etc/krb5/kpropd.acl
   host/kdc1.example.com@EXAMPLE.COM
   host/kdc2.example.com@EXAMPLE.COM
   host/kdc3.example.com@EXAMPLE.COM
   host/kdc4.example.com@EXAMPLE.COM

17 On the old master KDC, start kpropd and krb5kdc.
   kdc1 # svcadm enable -r network/security/krb5_prop
   kdc1 # svcadm enable -r network/security/krb5kdc
Administering the Kerberos Database

The Kerberos database is the backbone of Kerberos and must be maintained properly. This section provides some procedures on how to administer the Kerberos database, such as backing up and restoring the database, setting up incremental or parallel propagation, and administering the stash file. The steps to initially set up the database are in “How to Manually Configure a Master KDC” on page 391.

Backing Up and Propagating the Kerberos Database

Propagating the Kerberos database from the master KDC to the slave KDCs is one of the most important configuration tasks. If propagation doesn’t happen often enough, the master KDC and the slave KDCs will lose synchronization. So, if the master KDC goes down, the slave KDCs will not have the most recent database information. Also, if a slave KDC has been configured as a master KDC for purposes of load balancing, the clients that use that slave KDC as a master KDC will not have the latest information. Therefore, you must make sure that propagation occurs often enough or else configure the servers for incremental propagation, based on how often you change the Kerberos database. Incremental propagation is preferred over manual propagation because there is more administrative overhead when you manually propagate the database. Also, there are inefficiencies when you do full propagation of the database.

When you configure the master KDC, you set up the kprop_script command in a cron job to automatically back up the Kerberos database to the /var/krb5/slave_datatrans dump file and propagate it to the slave KDCs. But, as with any file, the Kerberos database can become corrupted. If data corruption occurs on a slave KDC, you might never notice, because the next automatic propagation of the database installs a fresh copy. However, if corruption occurs on the master KDC, the corrupted database is propagated to all of the slave KDCs during the next propagation. And, the corrupted backup overwrites the previous uncorrupted backup file on the master KDC.

Because there is no “safe” backup copy in this scenario, you should also set up a cron job to periodically copy the slave_datatrans dump file to another location or to create another separate backup copy by using the dump command of kdb5_util. Then, if your database becomes corrupted, you can restore the most recent backup on the master KDC by using the load command of kdb5_util.

Another important note: Because the database dump file contains principal keys, you need to protect the file from being accessed by unauthorized users. By default, the database dump file has read and write permissions only as root. To protect against unauthorized access, use only the kprop command to propagate the database dump file, which encrypts the data that is being transferred. Also, kprop propagates the data only to the slave KDCs, which minimizes the chance of accidentally sending the database dump file to unauthorized hosts.
Caution – If the Kerberos database is updated after it has been propagated and if the database subsequently is corrupted before the next propagation, the KDC slaves will not contain the updates. The updates will be lost. For this reason, if you add significant updates to the Kerberos database before a regularly scheduled propagation, you should manually propagate the database to avoid data loss.

The kpropd.acl File

The kpropd.acl file on a slave KDC provides a list of host principal names, one name per line, that specifies the systems from which the KDC can receive an updated database through propagation. If the master KDC is used to propagate all the slave KDCs, the kpropd.acl file on each slave needs to contain only the host principal name of the master KDC.

However, the Kerberos installation and subsequent configuration steps in this book instruct you to add the same kpropd.acl file to the master KDC and the slave KDCs. This file contains all the KDC host principal names. This configuration enables you to propagate from any KDC, in case the propagating KDCs become temporarily unavailable. And, by keeping an identical copy on all KDCs, you make the configuration easy to maintain.

The kprop_script Command

The kprop_script command uses the kprop command to propagate the Kerberos database to other KDCs. If the kprop_script command is run on a slave KDC, it propagates the slave KDC's copy of the Kerberos database to other KDCs. The kprop_script accepts a list of host names for arguments, separated by spaces, which denote the KDCs to propagate.

When kprop_script is run, it creates a backup of the Kerberos database to the /var/krb5/slave_datatrans file and copies the file to the specified KDCs. The Kerberos database is locked until the propagation is finished.

How to Back Up the Kerberos Database

1. Become superuser on the master KDC.

2. Back up the Kerberos database by using the dump command of the kdb5_util command.

   # /usr/sbin/kdb5_util dump [-verbose] [-d dbname] [filename [principals...]]

   -verbose        Prints the name of each principal and policy that is being backed up.

   dbname          Defines the name of the database to back up. Note that you can specify an absolute path for the file. If the -d option is not specified, the default database name is /var/krb5/principal.
filename  Defines the file that is used to back up the database. You can specify an absolute path for the file. If you don't specify a file, the database is dumped to standard output.

principals  Defines a list of one or more principals (separated by a space) to back up. You must use fully qualified principal names. If you don't specify any principals, the entire database is backed up.

Example 23–15  Backing Up the Kerberos Database

In the following example, the Kerberos database is backed up to a file called dumpfile. Because the -verbose option is specified, each principal is printed as it is backed up.

```bash
# kdb5_util dump -verbose dumpfile
kadmin/kdc1.eng.example.com@ENG.EXAMPLE.COM
krbtgt/ENG.EXAMPLE.COM@ENG.EXAMPLE.COM
kadmin/history@ENG.EXAMPLE.COM
pak/admin@ENG.EXAMPLE.COM
pak@ENG.EXAMPLE.COM
changepw/kdc1.eng.example.com@ENG.EXAMPLE.COM
```

In the following example, the pak and pak/admin principals from the Kerberos database are backed up.

```bash
# kdb5_util dump -verbose dumpfile pak/admin@ENG.EXAMPLE.COM pak@ENG.EXAMPLE.COM
pak/admin@ENG.EXAMPLE.COM
pak@ENG.EXAMPLE.COM
```

How to Restore the Kerberos Database

1  Become superuser on the master KDC.

2  On the master, stop the KDC daemons.
   kc1  # svcadm disable network/security/krb5kdc
   kc1  # svcadm disable network/security/kadmin

3  Restore the Kerberos database by using the load command of the kdb5_util command.
   # /usr/sbin/kdb5_util load [-verbose] [-d dbname] [-update] [filename]
   -verbose  Prints the name of each principal and policy that is being restored.
   dbname  Defines the name of the database to restore. Note you can specify an absolute path for the file. If the -d option is not specified, the default database name is /var/krb5/principal.
- update Updates the existing database. Otherwise, a new database is created or the existing database is overwritten.

filename Defines the file from which to restore the database. You can specify an absolute path for the file.

4 Start the KDC daemons.
   kdc1 # svcadm enable -r network/security/krb5kdc
   kdc1 # svcadm enable -r network/security/kadmin

Example 23–16 Restoring the Kerberos Database

In the following example, the database called database1 is restored into the current directory from the dumpfile file. Because the -update option isn't specified, a new database is created by the restore.

   # kdb5_util load -d database1 dumpfile

### How to Convert a Kerberos Database After a Server Upgrade

If your KDC database was created on a server running the Solaris 8 or Solaris 9 release, converting the database allows you to take advantage of the improved database format.

**Before You Begin** Make sure that the database is using an older format.

1 **On the master, stop the KDC daemons.**
   kdc1 # svcadm disable network/security/krb5kdc
   kdc1 # svcadm disable network/security/kadmin

2 **Create a directory to store a temporary copy of the database.**
   kdc1 # mkdir /var/krb5/tmp
   kdc1 # chmod 700 /var/krb5/tmp

3 **Dump the KDC database.**
   kdc1 # kdb5_util dump /var/krb5/tmp/prdb.txt

4 **Save copies of the current database files.**
   kdc1 # cd /var/krb5
   kdc1 # mv princ* tmp/
5 Load the database.
   kdc1 # kdb5_util load /var/krb5/tmp/prdb.txt

6 Start the KDC daemons.
   kdc1 # svcadm enable -r network/security/krb5kdc
   kdc1 # svcadm enable -r network/security/kadmin

▼ How to Reconfigure a Master KDC to Use Incremental Propagation

The steps in this procedure can be used to reconfigure an existing master KDC to use incremental propagation. In this procedure, the following configuration parameters are used:

- Realm name = EXAMPLE.COM
- DNS domain name = example.com
- Master KDC = kdc1.example.com
- Slave KDC = kdc2.example.com
- admin principal = kws/admin

1 Add entries to kdc.conf.

You need to enable incremental propagation and select the number of updates the KDC master keeps in the log. See the kdc.conf(4) man page for more information.

   kdc1 # cat /etc/krb5/kdc.conf
   [kdcdefaults]
      kdc_ports = 88,750
   [realms]
      EXAMPLE.COM =
         profile = /etc/krb5/krb5.conf
         database_name = /var/krb5/principal
         acl_file = /etc/krb5/kadm5.acl
         kadmind_port = 749
         max_life = 8h 0m 0s
         max_renewable_life = 7d 0h 0m 0s
         sunw_dbprop_enable = true
         sunw_dbprop_master__ulongsize = 1000
2 Create the kprop principal.
The kprop principal is used to authenticate the master KDC server and to authorize updates from the master KDC.

```
kdc1 # /usr/sbin/kadmin -p kws/admin
Enter password: <Type kws/admin password>
kadmin: addprinc -randkey kiprop/kdc1.example.com
Principal "kiprop/kdc1.example.com@EXAMPLE.COM" created.
kadmin: addprinc -randkey kiprop/kdc2.example.com
Principal "kiprop/kdc2.example.com@EXAMPLE.COM" created.
kadmin:
```

3 On the master KDC, add a kprop entry to kadm5.acl
   This entry allows the master KDC to receive requests for incremental propagation from the kdc2 server.

```
kdc1 # cat /etc/krb5/kadm5.acl
*/admin@EXAMPLE.COM *
kprop/kdc2.example.com@EXAMPLE.COM p
```

4 Comment out the kprop line in the root crontab file.
   This step prevents the master KDC from propagating its copy of the KDC database.

```
kdc1 # crontab -e
#ident "@(#)root 1.20 01/11/06 SMI"
#
# The root crontab should be used to perform accounting data collection.
#
# The rtc command is run to adjust the real time clock if and when
# daylight savings time changes.
#
10 3 * * * /usr/sbin/logadm
15 3 * * 0 /usr/lib/fs/nfs/nfsfind
1 2 * * * [ -x /usr/sbin/rtc ] && /usr/sbin/rtc -c > /dev/null 2>&1
30 3 * * * [ -x /usr/lib/gss/gsscred_clean ] && /usr/lib/gss/gsscred_clean
#10 3 * * * /usr/lib/krb5kprop_script kdc2.example.sun.com #SUNWkr5ma
```

5 Restart kadmind.
   kdc1 # svcadm restart network/security/kadmin

6 Reconfigure all slave KDC servers that use incremental propagation.
   See “How to Reconfigure a Slave KDC to Use Incremental Propagation” on page 449 for complete instructions.
How to Reconfigure a Slave KDC to Use Incremental Propagation

1. Add entries to krb5.conf.
   The new entries enable incremental propagation and set the poll time to 2 minutes.
   ```
   kdc2 # cat /etc/krb5/krb5.conf
   [kdcdefaults]
   kdc_ports = 88,750
   
   [realms]
   EXAMPLE.COM = {
      profile = /etc/krb5/krb5.conf
      database_name = /var/krb5/principal
      acl_file = /etc/krb5/kadm5.acl
      kadmind_port = 749
      max_life = 8h 0m 0s
      max_renewable_life = 7d 0h 0m 0s
      sunw_dbprop_enable = true
      sunw_dbprop_slave_poll = 2m
   }
   ```

2. Add the kprop principal to the krb5.keytab file.
   ```
   kdc2 /usr/sbin/kadmin -p kws/admin
   Enter password: <Type kws/admin password>
   kadmin: ktadd kprop/kdc2.example.com
   Entry for principal kprop/kdc2.example.com with kvno 3, encryption type AES-256 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
   Entry for principal kprop/kdc2.example.com with kvno 3, encryption type AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
   Entry for principal kprop/kdc2.example.com with kvno 3, encryption type Triple DES cbc mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
   Entry for principal kprop/kdc2.example.com with kvno 3, encryption type ArcFour with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
   Entry for principal kprop/kdc2.example.com with kvno 3, encryption type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
   kadmin: quit
   ```

3. Restart kpropd.
   ```
   kdc2 # svcadm restart network/security/krb5_prop
   ```
How to Configure a Slave KDC to Use Full Propagation

This procedure shows how to reconfigure a slave KDC server running the Solaris 10 release to use full propagation. Normally, the procedure would only need to be used if the master KDC server is running either the Solaris 9 release or an earlier release. In this case, the master KDC server can not support incremental propagation, so the slave needs to be configured to allow propagation to work.

In this procedure, a slave KDC named kdc3 is configured. This procedure uses the following configuration parameters:

- Realm name = EXAMPLE.COM
- DNS domain name = example.com
- Master KDC = kdc1.example.com
- Slave KDC = kdc2.example.com and kdc3.example.com
- admin principal = kws/admin

Note – Adjust the URL to point to the “Graphical Kerberos Administration Tool” section, as described in the “Online Help URL in the Graphical Kerberos Administration Tool” on page 385.

Before You Begin

The master KDC must be configured. For specific instructions if this slave is to be swappable, see “Swapping a Master KDC and a Slave KDC” on page 438.

1 On the master KDC, become superuser.

2 On the master KDC, start kadmin.

You must log in with one of the admin principal names that you created when you configured the master KDC.

    kdc1 # /usr/sbin/kadmin -p kws/admin
    Enter password: <Type kws/admin password>
    kadmin:

    a. On the master KDC, add slave host principals to the database, if not already done.

    For the slave to function, it must have a host principal. Note that when the principal instance is a host name, the FQDN must be specified in lowercase letters, regardless of the case of the domain name in the /etc/resolv.conf file.

    kadmin: addprinc -randkey host/kdc3.example.com
    Principal “host/kdc3@EXAMPLE.COM” created.
    kadmin:
b. Quit kadmin.
   
   kadmin: quit

3 On the master KDC, edit the Kerberos configuration file (`krb5.conf`).
   You need to add an entry for each slave. See the `krb5.conf(4)` man page for a full description of this file.
   
   kdc1 # cat /etc/krb5/krb5.conf

   [realms]
   EXAMPLE.COM = {
     kdc = kdc1.example.com
     kdc = kdc2.example.com
     kdc = kdc3.example.com
     admin_server = kdc1.example.com
   }

4 On the master KDC, add an entry for the master KDC and each slave KDC into the `kpropd.acl` file.
   See the `kprop(1M)` man page for a full description of this file.
   
   kdc1 # cat /etc/krb5/kpropd.acl
   host/kdc1.example.com@EXAMPLE.COM
   host/kdc2.example.com@EXAMPLE.COM
   host/kdc3.example.com@EXAMPLE.COM

5 On all slave KDCs, copy the KDC administration files from the master KDC server.
   This step needs to be followed on all slave KDCs, because the master KDC server has updated information that each KDC server needs. You can use `ftp` or a similar transfer mechanism to grab copies of the following files from the master KDC:

   - `/etc/krb5/krb5.conf`
   - `/etc/krb5/kdc.conf`
   - `/etc/krb5/kpropd.acl`

6 On all slave KDCs, make sure that the Kerberos access control list file, `kadm5.acl`, is not populated.
   An unmodified `kadm5.acl` file would look like:
   
   kdc2 # cat /etc/krb5/kadm5.acl
   */admin@___default_realm___ *

   If the file has `kiprop` entries, remove them.
7 On the new slave, start the kadmin command.

You must log in with one of the admin principal names that you created when you configured the master KDC.

```
kdc2 # /usr/sbin/kadmin -p kws/admin
```
Enter password: <Type kws/admin password>

```
kadmin:
```

a. Add the slave's host principal to the slave's keytab file by using kadmin.

This entry allows kprop and other Kerberized applications to function. Note that when the principal instance is a host name, the FQDN must be specified in lowercase letters, regardless of the case of the domain name in the /etc/resolv.conf file.

```
kadmin: ktadd host/kdc3.example.com
```
Entry for principal host/kdc3.example.com with kvno 3, encryption type AES-256 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc3.example.com with kvno 3, encryption type AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc3.example.com with kvno 3, encryption type Triple DES cbc mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc3.example.com with kvno 3, encryption type ArcFour with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc3.example.com with kvno 3, encryption type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.

```
kadmin:
```

b. Quit kadmin.

```
kadmin: quit
```

8 On the master KDC, add the slave KDC name to the cron job, which automatically runs the backups, by running crontab -e.

Add the name of each slave KDC server at the end of the kprop_script line.

```
10 3 * * * /usr/lib/krb5/kprop_script kdc2.example.com kdc3.example.com
```

You might also want to change the time of the backups. This entry starts the backup process every day at 3:10 AM.

9 On the new slave, start the Kerberos propagation daemon.

```
kdc3 # svcadm enable network/security/krb5_prop
```

Administering the Kerberos Database
10 **On the master KDC, back up and propagate the database by using** `kprop_script`.
   If a backup copy of the database is already available, it is not necessary to complete another backup. See “How to Manually Propagate the Kerberos Database to the Slave KDCs” on page 454 for further instructions.
   ```
   kdc1 # /usr/lib/krb5/kprop_script kdc3.example.com
   Database propagation to kdc3.example.com: SUCCEEDED
   ```

11 **On the new slave, create a stash file by using** `kdb5_util`.
   ```
   kdc3 # /usr/sbin/kdb5_util stash
   kdb5_util: Cannot find/read stored master key while reading master key
   kdb5_util: Warning: proceeding without master key
   Enter KDC database master key:  <Type the key>
   ```

12 **(Optional) On the new slave KDC, synchronize the master KDCs clock by using NTP or another clock synchronization mechanism.**
   Installing and using the Network Time Protocol (NTP) is not required. However, every clock must be within the default time that is defined in the `libdefaults` section of the `krb5.conf` file for authentication to succeed. See “Synchronizing Clocks Between KDCs and Kerberos Clients” on page 437 for information about NTP.

13 **On the new slave, start the KDC daemon (krb5kdc).**
   ```
   kdc3 # svcadm enable network/security/krb5kdc
   ```

**How to Verify That the KDC Servers Are Synchronized**

If incremental propagation has been configured, this procedure ensures that the information on the slave KDC has been updated.

1 **On the KDC master server, run the kproplog command.**
   ```
   kdc1 # /usr/sbin/kproplog -h
   ```

2 **On a KDC slave server, run the kproplog command.**
   ```
   kdc2 # /usr/sbin/kproplog -h
   ```

3 **Check that the last serial # and the last timestamp values match.**

**Example 23–17** Verifying That the KDC Servers Are Synchronized

The following is a sample of results from running the `kproplog` command on the master KDC server.
The following is a sample of results from running the `kproplog` command on a slave KDC server.

```
kdc2 # /usr/sbin/kproplog -h
```

Kerberos update log (/var/krb5/principal.ulog)
Update log dump:
  Log version #: 1
  Log state: Stable
  Entry block size: 2048
  Number of entries: 0
  First serial #: None
  Last serial #: 140465
  First time stamp: None
  Last time stamp: Fri Nov 28 01:06:13 2004

Notice that the values for the last serial number and the last timestamp are identical, which indicates that the slave is synchronized with the master KDC server.

In the slave KDC server output, notice that no update entries exist in the slave KDC server's update log. No entries exist because the slave KDC server does not keep a set of updates, unlike the master KDC server. Also, the KDC slave server does not include information on the first serial number or the first timestamp because this is not relevant information.

▼ How to Manually Propagate the Kerberos Database to the Slave KDCs

This procedure shows you how to propagate the Kerberos database by using the `kprop` command. Use this procedure if you need to synchronize a slave KDC with the master KDC outside the periodic `cron` job. Unlike the `kprop_script`, you can use `kprop` to propagate just the current database backup without first making a new backup of the Kerberos database.
Note – Do not use this procedure if you are using incremental propagation.

1 Become superuser on the master KDC.

2 (Optional) Back up the database by using the `kdb5_util` command.
   
   ```
   # /usr/sbin/kdb5_util dump /var/krb5/slave_datatrans
   ```

3 Propagate the database to a slave KDC by using the `kprop` command.
   
   ```
   # /usr/lib/krb5/kprop -f /var/krb5/slave_datatrans slave-KDC
   ```

**Example 23–18** Manually Propagating the Kerberos Database to the Slave KDCs Using `kprop_script`

If you want to back up the database and propagate it to a slave KDC outside the periodic `cron` job, you can also use the `kprop_script` command as follows:

```
# /usr/lib/krb5/kprop_script slave-KDC
```

**Setting Up Parallel Propagation**

In most cases, the master KDC is used exclusively to propagate its Kerberos database to the slave KDCs. However, if your site has many slave KDCs, you might consider load-sharing the propagation process, known as parallel propagation.

Note – Do not use this procedure if you are using incremental propagation.

Parallel propagation allows specific slave KDCs to share the propagation duties with the master KDC. This sharing of duties enables the propagation to be done faster and to lighten the work for the master KDC.

For example, say your site has one master KDC and six slave KDCs (shown in Figure 23–2), where slave-1 through slave-3 consist of one logical grouping and slave-4 through slave-6 consist of another logical grouping. To set up parallel propagation, you could have the master KDC propagate the database to slave-1 and slave-4. In turn, those KDC slaves could propagate the database to the KDC slaves in their group.
Configuration Steps for Setting Up Parallel Propagation

The following is not a detailed step-by-step procedure, but a high-level list of configuration steps to enable parallel propagation. These steps involve the following:

1. On the master KDC, changing the `kprop_script` entry in its cron job to include arguments for only the KDC slaves that will perform the succeeding propagation (the propagation slaves).

2. On each propagation slave, adding a `kprop_script` entry to its cron job, which must include arguments for the slaves to propagate. To successfully propagate in parallel, the cron job should be set up to run after the propagation slave is itself propagated with the new Kerberos database.

3. On each slave KDC, setting up the appropriate permissions to be propagated. This step is done by adding the host principal name of its propagating KDC to its `kpropd.acl` file.

**Note** – How long it will take for a propagation slave to be propagated depends on factors such as network bandwidth and the size of the Kerberos database.

Using the example in Figure 23–2, the master KDC's `kprop_script` entry would look similar to the following:

```
0 3 * * * /usr/lib/krb5/kprop_script slave-1.example.com slave-4.example.com
```

The slave-1's `kprop_script` entry would look similar to the following:

```
0 4 * * * /usr/lib/krb5/kprop_script slave-2.example.com slave-3.example.com
```

Note that the propagation on the slave starts an hour after it is propagated by the master.
The `kpropd.acl` file on the propagation slaves would contain the following entry:

`host/master.example.com@EXAMPLE.COM`

The `kpropd.acl` file on the KDC slaves being propagated by `slave-1` would contain the following entry:

`host/slave-1.example.com@EXAMPLE.COM`

**Administering the Stash File**

The *stash file* contains the master key for the Kerberos database, which is automatically created when you create a Kerberos database. If the stash file gets corrupted, you can use the `stash` command of the `kdb5_util` utility to replace the corrupted file. The only time you should need to remove a stash file is after removing the Kerberos database with the `destroy` command of `kdb5_util`. Because the stash file is not automatically removed with the database, you have to remove the stash file to finish the cleanup.

### How to Remove a Stash File

1. Become superuser on the KDC that contains the stash file.

2. Remove the stash file.

   ```
   # rm stash-file
   ```

   Where `stash-file` is the path to the stash file. By default, the stash file is located at `/var/krb5/.k5.realm`.

   **Note** – If you need to re-create the stash file, you can use the `-f` option of the `kdb5_util` command.

**Managing a KDC on an LDAP Directory Server**

Most of the KDC administration tasks using an LDAP Directory Server are the same as those for the DB2 server. There are some new tasks that are specific to working with LDAP.
Managing a KDC on an LDAP Directory Server

### TABLE 23–3 Configuring KDC Servers to Use LDAP (Task Map)

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuring a Master KDC</td>
<td>Configures and builds the master KDC server and database for a realm using a manual process and using LDAP for the KDC.</td>
<td>“How to Configure a KDC to Use an LDAP Data Server” on page 396</td>
</tr>
<tr>
<td>Mix Kerberos principal attributes with non-Kerberos object class types.</td>
<td>Allows information stored with the Kerberos records to be shared with other LDAP databases.</td>
<td>“How to Mix Kerberos Principal Attributes in a Non-Kerberos Object Class Type” on page 458</td>
</tr>
<tr>
<td>Destroy a Realm</td>
<td>Removes all of the data associated with a realm</td>
<td>“How to Destroy a Realm on an LDAP Directory Server” on page 459</td>
</tr>
</tbody>
</table>

### How to Mix Kerberos Principal Attributes in a Non-Kerberos Object Class Type

This procedure allows for Kerberos principal attributes to be associated with non-Kerberos object class types. In this procedure the krbprincipalaux, krbTicketPolicyAux and krbPrincipalName attributes are associated with the people object class.

In this procedure, the following configuration parameters are used:

- Directory Server = dsserver.example.com
- user principal = willf@EXAMPLE.COM

1. **Become superuser.**

2. **Prepare each entry in the people object class.**
   
   Repeat this step for each entry.
   
   ```bash
   cat << EOF | ldapmodify -h dsserver.example.com -D "cn=directory manager"
   dn: uid=willf,ou=people,dc=example,dc=com
   changetype: modify
   objectClass: krbprincipalaux
   objectClass: krbTicketPolicyAux
   krbPrincipalName: willf@EXAMPLE.COM
   EOF
   ```

3. **Add a subtree attribute to the realm container.**
   
   This step allows for searching of principal entries in the ou=people,dc=example,dc=com container, as well as in the default EXAMPLE.COM container.
   
   ```bash
   # kdb5_ldap_util -D "cn=directory manager" modify \
   -subtrees 'ou=people,dc=example,dc=com' -r EXAMPLE.COM
   ```
4 (Optional) If the KDC records are stored in DB2, migrate DB2 entries.
   a. Dump the DB2 entries.
      # kdb5_util dump > dumpfile
   b. Load the database into the LDAP server.
      # kdb5_util load -update dumpfile

5 (Optional) Add the principal attributes to the KDC.
   # kadmin.local -q 'addprinc willf'

▼ How to Destroy a Realm on an LDAP Directory Server

This procedure can be used if a different LDAP Directory Server has been configured to handle a realm.

1 Become superuser.

2 Destroy the realm.
   # kdb5_ldap_util -D "cn=directory manager" destroy

### Increasing Security on Kerberos Servers

Follow these steps to increase security on Kerberos application servers and on KDC servers.

**TABLE 23–4** Increasing Security on Kerberos Servers (Task Map)

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabling access using Kerberos authentication</td>
<td>Restrict network access to a server to allow Kerberos authentication only</td>
<td>&quot;How to Enable Only Kerberized Applications&quot; on page 459</td>
</tr>
<tr>
<td>Restricting access to the KDC servers</td>
<td>Increases the security of the KDC servers and their data.</td>
<td>&quot;How to Restrict Access to KDC Servers&quot; on page 460</td>
</tr>
<tr>
<td>Increasing password security by using a dictionary file</td>
<td>Increases the security of any new passwords by checking the new password against a dictionary.</td>
<td>&quot;How to Use a Dictionary File to Increase Password Security&quot; on page 461</td>
</tr>
</tbody>
</table>

▼ How to Enable Only Kerberized Applications

This procedure restricts network access to the server that is running `telnet`, `ftp`, `rcp`, `rsh`, and `rlogin` to use Kerberos authenticated transactions only.
1 Change the *exec* property for the telnet service.
   Add the -a user option to the exec property for telnet to restrict access to those users who can provide valid authentication information.
   
   ```
   # inetadm -m svc:/network/telnet:default exec="/usr/sbin/in.telnetd -a user"
   ```

2 *(Optional) If not already configured, change the exec property for the telnet service.*
   Add the -a option to the exec property for ftp to permit only Kerberos authenticated connections.
   
   ```
   # inetadm -m svc:/network/ftp:default exec="/usr/sbin/in.ftpd -a"
   ```

3 Disable other services.
   The *in.rshd* and *in.rlogind* daemons should be disabled.
   
   ```
   # svcadm disable network/shell
   # svcadm disable network/login:rlogin
   ```

▼ **How to Restrict Access to KDC Servers**

Both master KDC servers and slave KDC servers have copies of the KDC database stored locally. Restricting access to these servers so that the databases are secure is important to the overall security of the Kerberos installation.

1 Disable remote services, as needed.

   To provide a secure KDC server, all nonessential network services should be disabled. Depending on your configuration, some of these services may already be disabled. Check the service status with the `svcs` command. In most circumstances, the only services that would need to run would be `krb5kdc` and `krdb5_kprop` if the KDC is a slave or only `kadmin` if the KDC is a master. In addition, any services that use loopback lls (tlcts, ticots0rd, and ticots) can be left enabled.

   ```
   # svcadm disable network/comsat
   # svcadm disable network/dtspc/tcp
   # svcadm disable network/finger
   # svcadm disable network/login:rlogin
   # svcadm disable network/reexec
   # svcadm disable network/shell
   # svcadm disable network/talk
   # svcadm disable network/tnames
   # svcadm disable network/uucp
   # svcadm disable network/rpc_100068_2-5/rpc_udp
   ```

2 Restrict access to the hardware that supports the KDC.

   To restrict physical access, make sure that the KDC server and its monitor are located in a secure facility. Users should not be able to access this server in any way.
3 Store KDC database backups on local disks or on the KDC slaves.
   Make tape backups of your KDC only if the tapes are stored securely. Follow the same practice
   for copies of keytab files. It would be best to store these files on a local file system that is not
   shared with other systems. The storage file system can be on either the master KDC server or
   any of the slave KDCs.

▼ How to Use a Dictionary File to Increase Password Security

A dictionary file can be used by the Kerberos service to prevent words in the dictionary from
being used as passwords when creating new credentials. Preventing the use of dictionary terms
as passwords makes it harder for someone else to guess any password. By default the
/var/krb5/kadm5.dict file is used, but it is empty.

1 Become superuser on the master KDC.

2 Edit the KDC configuration file (kdc.conf).
   You need add a line to instruct the service to use a dictionary file. In this example, the dictionary
   that is included with the spell utility is used. See the kdc.conf(4) man page for a full
description of the configuration file.
   kdc1 # cat /etc/krb5/kdc.conf
   [kdcdefaults]
     kdc_ports = 88,750
   [realms]
     EXAMPLE.COM = {
       profile = /etc/krb5/krb5.conf
       database_name = /var/krb5/principal
       acl_file = /etc/krb5/kadm5.acl
       kadmin_port = 749
       max_life = 8h 0m 0s
       max_renewable_life = 7d 0h 0m 0s
       sunw_dbprop_enable = true
       sunw_dbprop_master_ ulogsize = 1000
       dict_file = /usr/share/lib/dict/words
     }

3 Restart the Kerberos daemons.
   kdc1 # svcadm restart -r network/security/krb5kdc
   kdc1 # svcadm restart -r network/security/kadmin
Kerberos Error Messages and Troubleshooting

This chapter provides resolutions for error messages that you might receive when you use the Kerberos service. This chapter also provides some troubleshooting tips for various problems. This is a list of the error message and troubleshooting information in this chapter.

- “SEAM Administration Tool Error Messages” on page 463
- “Common Kerberos Error Messages (A-M)” on page 464
- “Common Kerberos Error Messages (N-Z)” on page 472
- “Problems With the Format of the krb5.conf File” on page 475
- “Problems Propagating the Kerberos Database” on page 475
- “Problems Mounting a Kerberized NFS File System” on page 476
- “Problems Authenticating as root” on page 477
- “Observing Mapping from GSS Credentials to UNIX Credentials” on page 477

Kerberos Error Messages

This section provides information about Kerberos error messages, including why each error occurs and a way to fix it.

SEAM Administration Tool Error Messages

Unable to view the list of principals or policies; use the Name field.

Cause: The admin principal that you logged in with does not have the list privilege (l) in the Kerberos ACL file (kadms5.ac1). So, you cannot view the principal list or policy list.

Solution: You must type the principal and policy names in the Name field to work on them, or you need to log in with a principal that has the appropriate privileges.
JNI: Java array creation failed
JNI: Java class lookup failed
JNI: Java field lookup failed
JNI: Java method lookup failed
JNI: Java object lookup failed
JNI: Java object field lookup failed
JNI: Java string access failed
JNI: Java string creation failed

Cause: A serious problem exists with the Java Native Interface that is used by the SEAM Administration Tool (gkadmin).

Solution: Exit gkadmin and restart it. If the problem persists, please report a bug.

Common Kerberos Error Messages (A-M)

This section provides an alphabetical list (A-M) of common error messages for the Kerberos commands, Kerberos daemons, PAM framework, GSS interface, the NFS service, and the Kerberos library.

All authentication systems disabled; connection refused

Cause: This version of rlogind does not support any authentication mechanism.

Solution: Make sure that rlogind is invoked with the -k option.

Another authentication mechanism must be used to access this host

Cause: Authentication could not be done.

Solution: Make sure that the client is using Kerberos V5 mechanism for authentication.

Authentication negotiation has failed, which is required for encryption. Good bye.

Cause: Authentication could not be negotiated with the server.

Solution: Start authentication debugging by invoking the telnet command with the toggle authdebug command and look at the debug messages for further clues. Also, make sure that you have valid credentials.

Bad krb5 admin server hostname while initializing kadmin interface

Cause: An invalid host name is configured for admin_server in the krb5.conf file.

Solution: Make sure that the correct host name for the master KDC is specified on the admin_server line in the krb5.conf file.
Bad lifetime value
  **Cause:** The lifetime value provided is not valid or incorrectly formatted.
  
  **Solution:** Make sure that the value provided is consistent with the Time Formats section in the `kinit(1)` man page.

Bad start time value
  **Cause:** The start time value provided is not valid or incorrectly formatted.
  
  **Solution:** Make sure that the value provided is consistent with the Time Formats section in the `kinit(1)` man page.

Cannot contact any KDC for requested realm
  **Cause:** No KDC responded in the requested realm.
  
  **Solution:** Make sure that at least one KDC (either the master or a slave) is reachable or that the `krb5kdc` daemon is running on the KDCs. Check the `/etc/krb5/krb5.conf` file for the list of configured KDCs (`kdc = kdc-name`).

Cannot determine realm for host
  **Cause:** Kerberos cannot determine the realm name for the host.
  
  **Solution:** Make sure that there is a default realm name, or that the domain name mappings are set up in the Kerberos configuration file (`krb5.conf`).

Cannot find KDC for requested realm
  **Cause:** No KDC was found in the requested realm.
  
  **Solution:** Make sure that the Kerberos configuration file (`krb5.conf`) specifies a KDC in the `realm` section.

cannot initialize realm realm-name
  **Cause:** The KDC might not have a stash file.
  
  **Solution:** Make sure that the KDC has a stash file. If not, create a stash file by using the `kdb5_util` command, and try restarting the `krb5kdc` command.

Cannot resolve KDC for requested realm
  **Cause:** Kerberos cannot determine any KDC for the realm.
  
  **Solution:** Make sure that the Kerberos configuration file (`krb5.conf`) specifies a KDC in the `realm` section.

Cannot reuse password
  **Cause:** The password that you specified has been used before by this principal.
**Solution:** Choose a password that has not been chosen before, at least not within the number of passwords that are kept in the KDC database for each principal. This policy is enforced by the principal's policy.

Can't get forwarded credentials  
**Cause:** Credential forwarding could not be established.  
**Solution:** Make sure that the principal has forwardable credentials.

Can't open/find Kerberos configuration file  
**Cause:** The Kerberos configuration file (`krb5.conf`) was unavailable.  
**Solution:** Make sure that the `krb5.conf` file is available in the correct location and has the correct permissions. This file should be writable by `root` and readable by everyone else.

Client did not supply required checksum--connection rejected  
**Cause:** Authentication with checksum was not negotiated with the client. The client might be using an old Kerberos V5 protocol that does not support initial connection support.  
**Solution:** Make sure that the client is using a Kerberos V5 protocol that supports initial connection support.

Client/server realm mismatch in initial ticket request  
**Cause:** A realm mismatch between the client and server occurred in the initial ticket request.  
**Solution:** Make sure that the server you are communicating with is in the same realm as the client, or that the realm configurations are correct.

Client or server has a null key  
**Cause:** The principal has a null key.  
**Solution:** Modify the principal to have a non-null key by using the `cpw` command of `kadmin`.

Communication failure with server while initializing `kadmin` interface  
**Cause:** The host that was specified for the admin server, also called the master KDC, did not have the `kadmind` daemon running.  
**Solution:** Make sure that you specified the correct host name for the master KDC. If you specified the correct host name, make sure that `kadmind` is running on the master KDC that you specified.

Credentials cache file permissions incorrect  
**Cause:** You do not have the appropriate read or write permissions on the credentials cache (`/tmp/krb5cc_uid`).  
**Solution:** Make sure that you have read and write permissions on the credentials cache.
Credentials cache I/O operation failed XXX

**Cause:** Kerberos had a problem writing to the system's credentials cache (/tmp/krb5cc_uid).

**Solution:** Make sure that the credentials cache has not been removed, and that there is space left on the device by using the `df` command.

Decrypt integrity check failed

**Cause:** You might have an invalid ticket.

**Solution:** Verify both of these conditions:

- Make sure that your credentials are valid. Destroy your tickets with `kdestroy`, and create new tickets with `kinit`.
- Make sure that the target host has a keytab file with the correct version of the service key. Use `kadmin` to view the key version number of the service principal (for example, `host/FQDN-hostname`) in the Kerberos database. Also, use `klist -k` on the target host to make sure that it has the same key version number.

Encryption could not be enabled. Goodbye.

**Cause:** Encryption could not be negotiated with the server.

**Solution:** Start authentication debugging by invoking the `telnet` command with the `toggle encdebug` command and look at the debug messages for further clues.

failed to obtain credentials cache

**Cause:** During `kadmin` initialization, a failure occurred when `kadmin` tried to obtain credentials for the admin principal.

**Solution:** Make sure that you used the correct principal and password when you executed `kadmin`.

Field is too long for this implementation

**Cause:** The message size that was being sent by a Kerberized application was too long. This error could be generated if the transport protocol is UDP, which has a default maximum message size 65535 bytes. In addition, there are limits on individual fields within a protocol message that is sent by the Kerberos service.

**Solution:** Verify that you have not restricted the transport to UDP in the KDC server's `/etc/krb5/kdc.conf` file.

GSS-API (or Kerberos) error

**Cause:** This message is a generic GSS-API or Kerberos error message and can be caused by several different problems.

**Solution:** Check the `/var/krb5/kdc.log` file to find the more specific error message that was logged when this error occurred.
Hostname cannot be canonicalized

**Cause:** Kerberos cannot make the host name fully qualified.

**Solution:** Make sure that the host name is defined in DNS and that the host-name-to-address and address-to-host-name mappings are consistent.

Illegal cross-realm ticket

**Cause:** The ticket sent did not have the correct cross-realms. The realms might not have the correct trust relationships set up.

**Solution:** Make sure that the realms you are using have the correct trust relationships.

Improper format of Kerberos configuration file

**Cause:** The Kerberos configuration file has invalid entries.

**Solution:** Make sure that all the relations in the krb5.conf file are followed by the “=” sign and a value. Also, verify that the brackets are present in pairs for each subsection.

Inappropriate type of checksum in message

**Cause:** The message contained an invalid checksum type.

**Solution:** Check which valid checksum types are specified in the krb5.conf and kdc.conf files.

Incorrect net address

**Cause:** There was a mismatch in the network address. The network address in the ticket that was being forwarded was different from the network address where the ticket was processed. This message might occur when tickets are being forwarded.

**Solution:** Make sure that the network addresses are correct. Destroy your tickets with kdestroy, and create new tickets with kinit.

Invalid credential was supplied

Service key not available

**Cause:** The service ticket in the credentials cache may be incorrect.

**Solution:** Destroy current credential cache and rerun kinit before trying to use this service.

Invalid flag for file lock mode

**Cause:** An internal Kerberos error occurred.

**Solution:** Please report a bug.

Invalid message type specified for encoding

**Cause:** Kerberos could not recognize the message type that was sent by the Kerberized application.
Solution: If you are using a Kerberized application that was developed by your site or a vendor, make sure that it is using Kerberos correctly.

Invalid number of character classes
Cause: The password that you specified for the principal does not contain enough password classes, as enforced by the principal’s policy.

Solution: Make sure that you specify a password with the minimum number of password classes that the policy requires.

KADM err: Memory allocation failure
Cause: There is insufficient memory to run kadmin.

Solution: Free up memory and try running kadmin again.

kadmin: Bad encryption type while changing host/<FQDN>’s key
Cause: More default encryption types are included in the base release in the Solaris 10 8/07 release. Clients can request encryption types that may not be supported by a KDC running an older version of the Solaris software.

Solution: Several solutions exist to fix this problem. The easiest one to implement is listed first:
1. Add the SUNWcry and SUNWcryr packages to the KDC server. This increases the number of encryption types supported by the KDC.
2. Set permitted_enctypes in krb5.conf on the client to not include the aes256 encryption type. This step will need to be done on each new client.

KDC can’t fulfill requested option
Cause: The KDC did not allow the requested option. A possible problem might be that postdating or forwardable options were being requested, and the KDC did not allow them. Another problem might be that you requested the renewal of a TGT, but you didn’t have a renewable TGT.

Solution: Determine if you are either requesting an option that the KDC does not allow or a type of ticket that is not available.

KDC policy rejects request
Cause: The KDC policy did not allow the request. For example, the request to the KDC did not have an IP address in its request. Or forwarding was requested, but the KDC did not allow it.

Solution: Make sure that you are using kinit with the correct options. If necessary, modify the policy that is associated with the principal or change the principal’s attributes to allow the request. You can modify the policy or principal by using kadmin.
KDC reply did not match expectations

**Cause:** The KDC reply did not contain the expected principal name, or other values in the response were incorrect.

**Solution:** Make sure that the KDC you are communicating with complies with RFC1510, that the request you are sending is a Kerberos V5 request, or that the KDC is available.

kdestroy: Could not obtain principal name from cache

**Cause:** The credentials cache is missing or corrupted.

**Solution:** Check that the cache location provided is correct. Remove and obtain a new TGT using `kinit`, if necessary.

kdestroy: No credentials cache file found while destroying cache

**Cause:** The credentials cache (`/tmp/krb5c_<uid>`) is missing or corrupted.

**Solution:** Check that the cache location provided is correct. Remove and obtain a new TGT using `kinit`, if necessary.

kdestroy: TGT expire warning NOT deleted

**Cause:** The credentials cache is missing or corrupted.

**Solution:** Check that the cache location provided is correct. Remove and obtain a new TGT using `kinit`, if necessary.

Kerberos authentication failed

**Cause:** The Kerberos password is either incorrect or the password might not be synchronized with the UNIX password.

**Solution:** If the password are not synchronized, then you must specify a different password to complete Kerberos authentication. It is possible that the user has forgotten their original password.

Kerberos V5 refuses authentication

**Cause:** Authentication could not be negotiated with the server.

**Solution:** Start authentication debugging by invoking the `telnet` command with the `toggle authdebug` command and look at the debug messages for further clues. Also, make sure that you have valid credentials.

Key table entry not found

**Cause:** No entry exists for the service principal in the network application server’s keytab file.

**Solution:** Add the appropriate service principal to the server’s keytab file so that it can provide the Kerberized service.
Key version number for principal in key table is incorrect

**Cause:** A principal’s key version in the keytab file is different from the version in the Kerberos database. Either a service’s key has been changed, or you might be using an old service ticket.

**Solution:** If a service’s key has been changed (for example, by using kadmin), you need to extract the new key and store it in the host’s keytab file where the service is running.

Alternately, you might be using an old service ticket that has an older key. You might want to run the kdestroy command and then the kinit command again.

**kinit: gethostname failed**

**Cause:** An error in the local network configuration is causing kinit to fail.

**Solution:** Make sure that the host is configured correctly.

**login: load_modules: can not open module /usr/lib/security/pam_krb5.so.1**

**Cause:** Either the Kerberos PAM module is missing or it is not a valid executable binary.

**Solution:** Make sure that the Kerberos PAM module is in the /usr/lib/security directory and that it is a valid executable binary. Also, make sure that the /etc/pam.conf file contains the correct path to pam_krb5.so.1.

Looping detected inside krb5_get_in_tkt

**Cause:** Kerberos made several attempts to get the initial tickets but failed.

**Solution:** Make sure that at least one KDC is responding to authentication requests.

**Master key does not match database**

**Cause:** The loaded database dump was not created from a database that contains the master key. The master key is located in /var/krb5/.k5.REALM.

**Solution:** Make sure that the master key in the loaded database dump matches the master key that is located in /var/krb5/.k5.REALM.

Matching credential not found

**Cause:** The matching credential for your request was not found. Your request requires credentials that are unavailable in the credentials cache.

**Solution:** Destroy your tickets with kdestroy, and create new tickets with kinit.

Message out of order

**Cause:** Messages that were sent using sequential-order privacy arrived out of order. Some messages might have been lost in transit.

**Solution:** You should reinitialize the Kerberos session.
Message stream modified

**Cause:** There was a mismatch between the computed checksum and the message checksum. The message might have been modified while in transit, which can indicate a security leak.

**Solution:** Make sure that the messages are being sent across the network correctly. Because this message can also indicate the possible tampering of messages while they are being sent, destroy your tickets using `kdestroy` and reinitialize the Kerberos services that you are using.

**Common Kerberos Error Messages (N-Z)**

This section provides an alphabetical list (N-Z) of common error messages for the Kerberos commands, Kerberos daemons, PAM framework, GSS interface, the NFS service, and the Kerberos library.

No credentials cache file found

**Cause:** Kerberos could not find the credentials cache (/tmp/krb5cc_uid).

**Solution:** Make sure that the credential file exists and is readable. If it isn’t, try performing `kinit` again.

No credentials were supplied, or the credentials were unavailable or inaccessible

No credential cache found

**Cause:** The user’s credential cache is incorrect or does not exist.

**Solution:** The user should run `kinit` before trying to start the service.

No credentials were supplied, or the credentials were unavailable or inaccessible

No principal in keytab matches desired name

**Cause:** An error occurred while trying to authenticate the server.

**Solution:** Make sure that the host or service principal is in the server’s keytab file.

Operation requires “privilege” privilege

**Cause:** The `admin` principal that was being used does not have the appropriate privilege configured in the `kadm5.acl` file.

**Solution:** Use a principal that has the appropriate privileges. Or, configure the principal that was being used to have the appropriate privileges by modifying the `kadm5.acl` file. Usually, a principal with `/admin` as part of its name has the appropriate privileges.
PAM-KRB5 (auth): krb5_verify_init_creds failed: Key table entry not found

**Cause:** The remote application tried to read the host's service principal in the local /etc/krb5/krb5.keytab file, but one does not exist.

**Solution:** Add the host's service principal to the host's keytab file.

Password is in the password dictionary

**Cause:** The password that you specified is in a password dictionary that is being used. Your password is not a good choice for a password.

**Solution:** Choose a password that has a mix of password classes.

Permission denied in replay cache code

**Cause:** The system's replay cache could not be opened. Your server might have been first run under a user ID different than your current user ID.

**Solution:** Make sure that the replay cache has the appropriate permissions. The replay cache is stored on the host where the Kerberized server application is running. The replay cache file is called /var/krb5/rcache/rc_service_name_uid for non-root users. For root users the replay cache file is called /var/krb5/rcache/root/rc_service_name.

Protocol version mismatch

**Cause:** Most likely, a Kerberos V4 request was sent to the KDC. The Kerberos service supports only the Kerberos V5 protocol.

**Solution:** Make sure that your applications are using the Kerberos V5 protocol.

Request is a replay

**Cause:** The request has already been sent to this server and processed. The tickets might have been stolen, and someone else is trying to reuse the tickets.

**Solution:** Wait for a few minutes, and reissue the request.

Requested principal and ticket don’t match

**Cause:** The service principal that you are connecting to and the service ticket that you have do not match.

**Solution:** Make sure that DNS is functioning properly. If you are using another vendor's software, make sure that the software is using principal names correctly.

Requested protocol version not supported

**Cause:** Most likely, a Kerberos V4 request was sent to the KDC. The Kerberos service supports only the Kerberos V5 protocol.

**Solution:** Make sure that your applications are using the Kerberos V5 protocol.
Server refused to negotiate authentication, which is required for encryption. Good bye.

**Cause:** The remote application is not capable or has been configured not to accept Kerberos authentication from the client.

**Solution:** Provide a remote application that can negotiate authentication or configure the application to use the appropriate flags to turn on authentication.

Server refused to negotiate encryption. Good bye.

**Cause:** Encryption could not be negotiated with the server.

**Solution:** Start authentication debugging by invoking the `telnet` command with the `toggle encdebug` command and look at the debug messages for further clues.

Server rejected authentication (during sendauth exchange)

**Cause:** The server that you are trying to communicate with rejected the authentication. Most often, this error occurs during Kerberos database propagation. Some common causes might be problems with the `kpropd.acl` file, DNS, or the keytab file.

**Solution:** If you get this error when you are running applications other than `kprop`, investigate whether the server’s keytab file is correct.

The ticket isn’t for us

**Ticket/authenticator don’t match**

**Cause:** There was a mismatch between the ticket and the authenticator. The principal name in the request might not have matched the service principal’s name. Either because the ticket was being sent with an FQDN name of the principal while the service expected a non-FQDN name, or a non-FQDN name was sent when the service expected an FQDN name.

**Solution:** If you get this error when you are running applications other than `kprop`, investigate whether the server’s keytab file is correct.

**Ticket expired**

**Cause:** Your ticket times have expired.

**Solution:** Destroy your tickets with `kdestroy`, and create new tickets with `kinit`.

**Ticket is ineligible for postdating**

**Cause:** The principal does not allow its tickets to be postdated.

**Solution:** Modify the principal with `kadmin` to allow postdating.

**Ticket not yet valid**

**Cause:** The postdated ticket is not valid yet.

**Solution:** Create a new ticket with the correct date, or wait until the current ticket is valid.
Truncated input file detected
   **Cause:** The database dump file that was being used in the operation is not a complete dump file.

   **Solution:** Create the dump file again, or use a different database dump file.

Unable to securely authenticate user ... exit
   **Cause:** Authentication could not be negotiated with the server.

   **Solution:** Start authentication debugging by invoking the `telnet` command with the `toggle authdebug` command and look at the debug messages for further clues. Also, make sure that you have valid credentials.

Wrong principal in request
   **Cause:** There was an invalid principal name in the ticket. This error might indicate a DNS or FQDN problem.

   **Solution:** Make sure that the principal of the service matches the principal in the ticket.

---

**Kerberos Troubleshooting**

This section provides troubleshooting information for the Kerberos software.

**Problems With the Format of the** `krb5.conf` **File**

If the `krb5.conf` file is not formatted properly, the `telnet` command will fail. However, the `dtlogin` and `login` commands will still succeed, even if the `krb5.conf` file is specified as required for the commands. If this problem occurs, the following error message is displayed:

```
Error initializing krb5: Improper format of Kerberos configuration
```

In addition, an incorrectly formatted `krb5.conf` file, prevents the applications that use the GSSAPI from using the krb5 mechanisms.

If there is a problem with the format of the `krb5.conf` file, you are vulnerable to security breaches. You should fix the problem before you allow Kerberos features to be used.

**Problems Propagating the Kerberos Database**

If propagating the Kerberos database fails, try `/usr/bin/rlogin -x` between the slave KDC and master KDC, and from the master KDC to the slave KDC server.

If the KDCs have been set up to restrict access, `rlogin` is disabled and cannot be used to troubleshoot this problem. To enable `rlogin` on a KDC, you must enable the `eklogin` service.
# svcadm enable svc:/network/login:eklogin

After you finish troubleshooting the problem, you need to disable the eklogin service.

If rlogin does not work, problems are likely because of the keytab files on the KDCs. If rlogin does work, the problem is not in the keytab file or the name service, because rlogin and the propagation software use the same host/host-name principal. In this case, make sure that the kpropd.acl file is correct.

## Problems Mounting a Kerberized NFS File System

- If mounting a Kerberized NFS file system fails, make sure that the /var/rcache/root file exists on the NFS server. If the file system is not owned by root, remove it and try the mount again.
- If you have a problem accessing a Kerberized NFS file system, make sure that the gssd service is enabled on your system and the NFS server.
- If you see either the invalid argument or bad directory error message when you are trying to access a Kerberized NFS file system, the problem might be that you are not using a fully qualified DNS name when you are trying to mount the NFS file system. The host that is being mounted is not the same as the host name part of the service principal in the server’s keytab file.

This problem might also occur if your server has multiple Ethernet interfaces, and you have set up DNS to use a “name per interface” scheme instead of a “multiple address records per host” scheme. For the Kerberos service, you should set up multiple address records per host as follows:

```plaintext
my.host.name. A 1.2.3.4
A 1.2.4.4
A 1.2.5.4

my-en0.host.name. A 1.2.3.4
my-en1.host.name. A 1.2.4.4
my-en2.host.name. A 1.2.5.4

4.3.2.1 PTR my.host.name.
4.4.2.1 PTR my.host.name.
4.5.2.1 PTR my.host.name.
```

In this example, the setup allows one reference to the different interfaces and a single service principal instead of three service principals in the server’s keytab file.

---

Problems Authenticating as root

If authentication fails when you try to become superuser on your system and you have already added the root principal to your host’s keytab file, there are two potential problems to check. First, make sure that the root principal in the keytab file has a fully qualified host name as its instance. If it does, check the `/etc/resolve.conf` file to make sure that the system is correctly set up as a DNS client.

Observing Mapping from GSS Credentials to UNIX Credentials

To be able to monitor the credential mappings, first uncomment this line from the `/etc/gss/gsscred.conf` file.

```
SYSLOG_UID_MAPPING=yes
```

Next instruct the gssd service to get information from the `/etc/gss/gsscred.conf` file.

```
# pkill -HUP gssd
```

Now you should be able to monitor the credential mappings as gssd requests them. The mappings are recorded by syslogd, if the `syslog.conf` file is configured for the auth system facility with the debug severity level.
This chapter provides procedures for administering principals and the policies that are associated with them. This chapter also shows how to administer a host’s keytab file.

This chapter should be used by anyone who needs to administer principals and policies. Before you use this chapter, you should be familiar with principals and policies, including any planning considerations. Refer to Chapter 21, “Introduction to the Kerberos Service,” and Chapter 22, “Planning for the Kerberos Service,” respectively.

This is a list of the information in this chapter.
- “Ways to Administer Kerberos Principals and Policies” on page 479
- “SEAM Administration Tool” on page 480
- “Administering Kerberos Principals” on page 483
- “Administering Kerberos Policies” on page 497
- “SEAM Tool Reference” on page 505
- “Administering Keytab Files” on page 509

Ways to Administer Kerberos Principals and Policies

The Kerberos database on the master KDC contains all of your realm’s Kerberos principals, their passwords, policies, and other administrative information. To create and delete principals, and to modify their attributes, you can use either the kadmin or gkadmin command.

The kadmin command provides an interactive command-line interface that enables you to maintain Kerberos principals, policies, and keytab files. There are two versions of the kadmin command:
- kadmin – Uses Kerberos authentication to operate securely from anywhere on the network
- kadmin.local – Must be run directly on the master KDC
Other than `kadmin` using Kerberos to authenticate the user, the capabilities of the two versions are identical. The local version is necessary to enable you to set up enough of the database so that you can use the remote version.

Also, the Solaris release provides the SEAM Administration Tool, `gkadmin`, which is an interactive graphical user interface (GUI) that provides essentially the same capabilities as the `kadmin` command. See “SEAM Administration Tool” on page 480 for more information.

**SEAM Administration Tool**

The SEAM Administration Tool (SEAM Tool) is an interactive graphical user interface (GUI) that enables you to maintain Kerberos principals and policies. This tool provides much the same capabilities as the `kadmin` command. However, this tool does not support the management of keytab files. You must use the `kadmin` command to administer keytab files, which is described in “Administering Keytab Files” on page 509.

Similar to the `kadmin` command, the SEAM Tool uses Kerberos authentication and encrypted RPC to operate securely from anywhere on the network. The SEAM Tool enables you to do the following:

- Create new principals that are based on default values or existing principals.
- Create new policies that are based on existing policies.
- Add comments for principals.
- Set up default values for creating new principals.
- Log in as another principal without exiting the tool.
- Print or save principal lists and policy lists.
- View and search principal lists and policy lists.

The SEAM Tool also provides context-sensitive help and general online help.

The following task maps provide pointers to the various tasks that you can do with the SEAM Tool:

- “Administering Kerberos Principals (Task Map)” on page 483
- “Administering Kerberos Policies (Task Map)” on page 497

Also, go to “SEAM Tool Panel Descriptions” on page 505 for descriptions of all the principal attributes and policy attributes that you can either specify or view in the SEAM Tool.

**Command-Line Equivalents of the SEAM Tool**

This section lists the `kadmin` commands that provide the same capabilities as the SEAM Tool. These commands can be used without running an X Window system. Even though most procedures in this chapter use the SEAM Tool, many procedures also provide corresponding examples that use the command-line equivalents.
TABLE 25–1 Command-Line Equivalents of the SEAM Tool

<table>
<thead>
<tr>
<th>SEAM Tool Procedure</th>
<th>Equivalent kadmin Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>View the list of principals.</td>
<td>list_principals or get_principals</td>
</tr>
<tr>
<td>View a principal’s attributes.</td>
<td>get_principal</td>
</tr>
<tr>
<td>Create a new principal.</td>
<td>add_principal</td>
</tr>
<tr>
<td>Duplicate a principal.</td>
<td>No command-line equivalent</td>
</tr>
<tr>
<td>Modify a principal.</td>
<td>modify_principal or change_password</td>
</tr>
<tr>
<td>Delete a principal.</td>
<td>delete_principal</td>
</tr>
<tr>
<td>Set up defaults for creating new principals.</td>
<td>No command-line equivalent</td>
</tr>
<tr>
<td>View the list of policies.</td>
<td>list_policies or get_policies</td>
</tr>
<tr>
<td>View a policy’s attributes.</td>
<td>get_policy</td>
</tr>
<tr>
<td>Create a new policy.</td>
<td>add_policy</td>
</tr>
<tr>
<td>Duplicate a policy.</td>
<td>No command-line equivalent</td>
</tr>
<tr>
<td>Modify a policy.</td>
<td>modify_policy</td>
</tr>
<tr>
<td>Delete a policy.</td>
<td>delete_policy</td>
</tr>
</tbody>
</table>

The Only File Modified by the SEAM Tool

The only file that the SEAM Tool modifies is the $HOME/.gkadmin file. This file contains the default values for creating new principals. You can update this file by choosing Properties from the Edit menu.

Print and Online Help Features of the SEAM Tool

The SEAM Tool provides both print features and online help features. From the Print menu, you can send the following to a printer or a file:

- List of available principals on the specified master KDC
- List of available policies on the specified master KDC
- The currently selected principal or the loaded principal
- The currently selected policy or the loaded policy

From the Help menu, you can access context-sensitive help and general help. When you choose Context-Sensitive Help from the Help menu, the Context-Sensitive Help window is displayed and the tool is switched to help mode. In help mode, when you click on any fields, labels, or buttons on the window, help on that item is displayed in the Help window. To switch back to the tool’s normal mode, click Dismiss in the Help window.
You can also choose Help Contents, which opens an HTML browser that provides pointers to the general overview and task information that is provided in this chapter.

**Working With Large Lists in the SEAM Tool**

As your site starts to accumulate a large number of principals and policies, the time it takes the SEAM Tool to load and display the principal and policy lists will become increasingly longer. Thus, your productivity with the tool will decrease. There are several ways to work around this problem.

First, you can completely eliminate the time to load the lists by not having the SEAM Tool load the lists. You can set this option by choosing Properties from the Edit menu, and unchecking the Show Lists field. Of course, when the tool doesn’t load the lists, it can’t display the lists, and you can no longer use the list panels to select principals or policies. Instead, you must type a principal or policy name in the new Name field that is provided, then select the operation that you want to perform on it. In effect, typing a name is equivalent to selecting an item from the list.

Another way to work with large lists is to cache them. In fact, caching the lists for a limited time is set as the default behavior for the SEAM Tool. The SEAM Tool must still initially load the lists into the cache. But after that, the tool can use the cache rather than retrieve the lists again. This option eliminates the need to keep loading the lists from the server, which is what takes so long.

You can set list caching by choosing Properties from the Edit menu. There are two cache settings. You can choose to cache the list forever, or you can specify a time limit when the tool must reload the lists from the server into the cache.

Caching the lists still enables you to use the list panels to select principals and policies, so it doesn’t affect how you use the SEAM Tool as the first option does. Also, even though caching doesn’t enable you to see the changes of other users, you can still see the latest list information based on your changes, because your changes update the lists both on the server and in the cache. And, if you want to update the cache to see other changes and get the lastest copy of the lists, you can use the Refresh menu whenever you want to refresh the cache from the server.

**How to Start the SEAM Tool**

1. **Start the SEAM Tool by using the gkadmin command.**
   
   $ /usr/sbin/gkadmin
   
   The SEAM Administration Login window is displayed.
If you don't want to use the default values, specify new default values.

The window automatically fills in with default values. The default principal name is determined by taking your current identity from the USER environment variable and appending `/admin` to it (`username/admin`). The default Realm and Master KDC fields are selected from the `/etc/krb5/krb5.conf` file. If you ever want to retrieve the default values, click Start Over.

**Note** – The administration operations that each Principal Name can perform are dictated by the Kerberos ACL file, `/etc/krb5/kadm5.acl`. For information about limited privileges, see “Using the SEAM Tool With Limited Kerberos Administration Privileges” on page 508.

3 Type a password for the specified principal name.

4 Click OK.

A window showing all of the principals is displayed.

### Administering Kerberos Principals

This section provides the step-by-step instructions used to administer principals with the SEAM Tool. This section also provides examples of command-line equivalents, when available.

#### Administering Kerberos Principals (Task Map)

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>View the list of principals.</td>
<td>View the list of principals by clicking the Principals tab.</td>
<td>“How to View the List of Kerberos Principals” on page 485</td>
</tr>
</tbody>
</table>
### Automating the Creation of New Kerberos Principals

Even though the SEAM Tool provides ease-of-use, it doesn’t provide a way to automate the creation of new principals. Automation is especially useful if you need to add 10 or even 100 new principals in a short time. However, by using the `kadmin.local` command in a Bourne shell script, you can do just that.

The following shell script line is an example of how to automate the creation of new principals:

```
awk '{ print "ank +needchange -pw", $2, $1 }' < /tmp/princnames |
    time /usr/sbin/kadmin.local> /dev/null
```

This example is split over two lines for readability. The script reads in a file called `princnames` that contains principal names and their passwords, and adds them to the Kerberos database. You would have to create the `princnames` file, which contains a principal name and its
password on each line, separated by one or more spaces. The +needchange option configures the principal so that the user is prompted for a new password during login with the principal for the first time. This practice helps to ensure that the passwords in the prinames file are not a security risk.

You can build more elaborate scripts. For example, your script could use the information in the name service to obtain the list of user names for the principal names. What you do and how you do it is determined by your site's needs and your scripting expertise.

▼ How to View the List of Kerberos Principals

An example of the command-line equivalent follows this procedure.

1 If necessary, start the SEAM Tool.
   See “How to Start the SEAM Tool” on page 482 for more information.
   $ /usr/sbin/gkadmin

2 Click the Principals tab.
   The list of principals is displayed.
3 **Display a specific principal or a sublist of principals.**

Type a filter string in the Filter field, and press Return. If the filter succeeds, the list of principals that match the filter is displayed.

The filter string must consist of one or more characters. Because the filter mechanism is case sensitive, you need to use the appropriate uppercase and lowercase letters for the filter. For example, if you type the filter string ge, the filter mechanism displays only the principals with the ge string in them (for example, george or edge).

If you want to display the entire list of principals, click Clear Filter.
Example 25–1  Viewing the List of Kerberos Principals (Command Line)

In the following example, the `list_principals` command of `kadmin` is used to list all the principals that match `kadmin*`. Wildcards can be used with the `list_principals` command.

```
  kadmin: list_principals kadmin*
  kadmin/changepw@EXAMPLE.COM
  kadmin/kdcl.example.com@EXAMPLE.COM
  kadmin/history@EXAMPLE.COM
  kadmin: quit
```

▼ How to View a Kerberos Principal's Attributes

An example of the command-line equivalent follows this procedure.

1  If necessary, start the SEAM Tool.
   See “How to Start the SEAM Tool” on page 482 for more information.
   $ /usr/sbin/gkadmin

2  Click the Principals tab.

3  Select the principal in the list that you want to view, then click Modify.
   The Principal Basics panel that contains some of the principal’s attributes is displayed.

4  Continue to click Next to view all the principal’s attributes.
   Three windows contain attribute information. Choose Context-Sensitive Help from the Help menu to get information about the various attributes in each window. Or, for all the principal attribute descriptions, go to “SEAM Tool Panel Descriptions” on page 505.

5  When you are finished viewing, click Cancel.

Example 25–2  Viewing a Kerberos Principal’s Attributes

The following example shows the first window when you are viewing the jdb/admin principal.
Example 25–3 Viewing a Kerberos Principal's Attributes (Command Line)

In the following example, the get_principal command of kadmin is used to view the attributes of the jdb/admin principal.

kadmin: getprinc jdb/admin
Principal: jdb/admin@EXAMPLE.COM
Expiration date: Fri Aug 25 17:19:05 PDT 2004
Last password change: [never]
Password expiration date: Wed Apr 14 11:53:10 PDT 2003
Maximum ticket life: 1 day 16:00:00
Maximum renewable life: 1 day 16:00:00
How to Create a New Kerberos Principal

An example of the command-line equivalent follows this procedure.

1. If necessary, start the SEAM Tool.
   See "How to Start the SEAM Tool" on page 482 for more information.

   Note – If you are creating a new principal that might need a new policy, you should create the new policy before you create the new principal. Go to "How to Create a New Kerberos Policy" on page 501.

   $ /usr/sbin/gkadmin

2. Click the Principals tab.

3. Click New.
   The Principal Basics panel that contains some attributes for a principal is displayed.

4. Specify a principal name and a password.
   Both the principal name and the password are mandatory.

5. Specify the encryption types for the principal.
   Click on the box to the right of the encryption key types field to open a new window that displays all of the encryption key types available. Click OK after selecting the required encryption types.
Specify the policy for the principal.

Specify values for the principal’s attributes, and continue to click Next to specify more attributes.

Three windows contain attribute information. Choose Context-Sensitive Help from the Help menu to get information about the various attributes in each window. Or, for all the principal attribute descriptions, go to “SEAM Tool Panel Descriptions” on page 505.

Click Save to save the principal, or click Done on the last panel.

If needed, set up Kerberos administration privileges for the new principal in the /etc/krb5/kadm5.acl file.

See “How to Modify the Kerberos Administration Privileges” on page 495 for more details.

Example 25–4 Creating a New Kerberos Principal

The following example shows the Principal Basics panel when a new principal called pak is created. The policy is set to testuser.
Administering Kerberos Principals

Chapter 25 • Administering Kerberos Principals and Policies (Tasks) 491

SEAM Administration Tool

File Edit Refresh

Principals Policies

Principal Basics

General

Principal Name: pak

Password: *********

Generate Random Password

Encryption Key Types: aes256-cts-hmac-sha1-96.no

Policy: testuser

Account Expires: Oct 8, 2010 10:49:40 AM

Admin History

Last Principal Change: Oct 8, 2009 11:35:10 AM

Last Changed By: kathys

Comments: 

Save Previous Next Cancel
Creating a New Kerberos Principal (Command Line)

In the following example, the add_principal command of kadmin is used to create a new principal called pak. The principal's policy is set to testuser.

```sh
kadmin: add_principal -policy testuser pak
Enter password for principal "pak@EXAMPLE.COM": <Type the password>
Re-enter password for principal "pak@EXAMPLE.COM": <Type the password again>
Principal "pak@EXAMPLE.COM" created.
kadmin: quit
```

How to Duplicate a Kerberos Principal

This procedure explains how to use all or some of the attributes of an existing principal to create a new principal. No command-line equivalent exists for this procedure.

1. **If necessary, start the SEAM Tool.**
   See "How to Start the SEAM Tool" on page 482 for more information.
   ```sh
   $ /usr/sbin/gkadmin
   ```

2. **Click the Principals tab.**

3. **Select the principal in the list that you want to duplicate, then click Duplicate.**
   The Principal Basics panel is displayed. All the attributes of the selected principal are duplicated, except for the Principal Name and Password fields, which are empty.

4. **Specify a principal name and a password.**
   Both the principal name and the password are mandatory. To make an exact duplicate of the principal you selected, click Save and skip to Step 7.

5. **Specify different values for the principal’s attributes, and continue to click Next to specify more attributes.**
   Three windows contain attribute information. Choose Context-Sensitive Help from the Help menu to get information about the various attributes in each window. Or, for all the principal attribute descriptions, go to “SEAM Tool Panel Descriptions” on page 505.

6. **Click Save to save the principal, or click Done on the last panel.**

7. **If needed, set up Kerberos administration privileges for the principal in /etc/krb5/kadm5.acl file.**
   See "How to Modify the Kerberos Administration Privileges" on page 495 for more details.
How to Modify a Kerberos Principal

An example of the command-line equivalent follows this procedure.

1 If necessary, start the SEAM Tool.
   See "How to Start the SEAM Tool" on page 482 for more information.
   
   $ /usr/sbin/gkadmin

2 Click the Principal tab.

3 Select the principal in the list that you want to modify, then click Modify.
   The Principal Basics panel that contains some of the attributes for the principal is displayed.

4 Modify the principal’s attributes, and continue to click Next to modify more attributes.
   Three windows contain attribute information. Choose Context-Sensitive Help from the Help menu to get information about the various attributes in each window. Or, for all the principal attribute descriptions, go to "SEAM Tool Panel Descriptions" on page 505.

   Note – You cannot modify a principal’s name. To rename a principal, you must duplicate the principal, specify a new name for it, save it, and then delete the old principal.

5 Click Save to save the principal, or click Done on the last panel.

6 Modify the Kerberos administration privileges for the principal in the /etc/krb5/kadm5.acl file.
   See "How to Modify the Kerberos Administration Privileges" on page 495 for more details.

Example 25–6 Modifying a Kerberos Principal’s Password (Command Line)

In the following example, the change_password command of kadmin is used to modify the password for the jdb principal. The change_password command does not let you change the password to a password that is in the principal’s password history.

   kadmin: change_password jdb
   Enter password for principal "jdb": <Type the new password>
   Re-enter password for principal "jdb": <Type the password again>
   Password for "jdb@EXAMPLE.COM" changed.
   kadmin: quit

To modify other attributes for a principal, you must use the modify_principal command of kadmin.
How to Delete a Kerberos Principal

An example of the command-line equivalent follows this procedure.

1. If necessary, start the SEAM Tool.
   See “How to Start the SEAM Tool” on page 482 for more information.
   $ /usr/sbin/gkadmin

2. Click the Principals tab.

3. Select the principal in the list that you want to delete, then click Delete.
   After you confirm the deletion, the principal is deleted.

4. Remove the principal from the Kerberos access control list (ACL) file, /etc/krb5/kadm5.acl.
   See “How to Modify the Kerberos Administration Privileges” on page 495 for more details.

Example 25–7 Deleting a Kerberos Principal (Command Line)

In the following example, the delete_principal command of kadmin is used to delete the jdb principal.

```
kadmin: delete_principal pak
Are you sure you want to delete the principal "pak@EXAMPLE.COM"? (yes/no): yes
Principal "pak@EXAMPLE.COM" deleted.
Make sure that you have removed this principal from all ACLs before reusing.
kadmin: quit
```

How to Set Up Defaults for Creating New Kerberos Principals

No command-line equivalent exists for this procedure.

1. If necessary, start the SEAM Tool.
   See “How to Start the SEAM Tool” on page 482 for more information.
   $ /usr/sbin/gkadmin

2. Choose Properties from the Edit Menu.
   The Properties window is displayed.
Select the defaults that you want to use when you create new principals. Choose Context-Sensitive Help from the Help menu for information about the various attributes in each window.

4 Click Save.

▼ How to Modify the Kerberos Administration Privileges

Even though your site probably has many user principals, you usually want only a few users to be able to administer the Kerberos database. Privileges to administer the Kerberos database are determined by the Kerberos access control list (ACL) file, kadm5.acl. The kadm5.acl file enables you to allow or disallow privileges for individual principals. Or, you can use the '*' wildcard in the principal name to specify privileges for groups of principals.

1 Become superuser on the master KDC.
2 **Edit the** /etc/krb5/kadm5.acl **file.**

An entry in the kadm5.acl file must have the following format:

```
principal privileges [principal-target]
```

**principal**  
Specifies the principal to which the privileges are granted. Any part of the principal name can include the ‘*’ wildcard, which is useful for providing the same privileges for a group of principals. For example, if you want to specify all principals with the admin instance, you would use */admin@realm.

Note that a common use of an admin instance is to grant separate privileges (such as administration access to the Kerberos database) to a separate Kerberos principal. For example, the user jdb might have a principal for his administrative use, called jdb/admin. This way, the user jdb obtains jdb/admin tickets only when he or she actually needs to use those privileges.

**privileges**  
Specifies which operations can or cannot be performed by the principal. This field consists of a string of one or more of the following list of characters or their uppercase counterparts. If the character is uppercase (or not specified), then the operation is disallowed. If the character is lowercase, then the operation is permitted.

```
a    [Dis]allows the addition of principals or policies.
d    [Dis]allows the deletion of principals or policies.
m    [Dis]allows the modification of principals or policies.
c    [Dis]allows the changing of passwords for principals.
i    [Dis]allows inquiriesto the Kerberos database.
l    [Dis]allows the listing of principals or policies in the Kerberos database.
```

**principal-target**  
When a principal is specified in this field, the privileges apply to the principal only when the principal operates on the principal-target. Any part of the principal name can include the ‘*’ wildcard, which is useful to group principals.

---

**Example 25–8**  
**Modifying the Kerberos Administration Privileges**

The following entry in the kadm5.acl file gives any principal in the EXAMPLE.COM realm with the admin instance all the privileges on the Kerberos database:

```
*/admin@EXAMPLE.COM *
```

The following entry in the kadm5.acl file gives the jdb@EXAMPLE.COM principal the privileges to add, list, and inquire about any principal that has the root instance.

```
jdb@EXAMPLE.COM ali */root@EXAMPLE.COM
```
Administering Kerberos Policies

This section provides step-by-step instructions used to administer policies with the SEAM Tool. This section also provides examples of command-line equivalents, when available.

Administering Kerberos Policies (Task Map)

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>View the list of policies.</td>
<td>View the list of policies by clicking the Policies tab.</td>
<td>“How to View the List of Kerberos Policies” on page 497</td>
</tr>
<tr>
<td>View a policy’s attributes.</td>
<td>View a policy’s attributes by selecting the policy in the Policy List, then clicking the Modify button.</td>
<td>“How to View a Kerberos Policy’s Attributes” on page 499</td>
</tr>
<tr>
<td>Create a new policy.</td>
<td>Create a new policy by clicking the Create New button in the Policy List panel.</td>
<td>“How to Create a New Kerberos Policy” on page 501</td>
</tr>
<tr>
<td>Duplicate a policy.</td>
<td>Duplicate a policy by selecting the policy to duplicate in the Policy List, then clicking the Duplicate button.</td>
<td>“How to Duplicate a Kerberos Policy” on page 503</td>
</tr>
<tr>
<td>Modify a policy.</td>
<td>Modify a policy by selecting the policy to modify in the Policy List, then clicking the Modify button. Note that you cannot modify a policy’s name. To rename a policy, you must duplicate the policy, specify a new name for it, save it, and then delete the old policy.</td>
<td>“How to Modify a Kerberos Policy” on page 503</td>
</tr>
<tr>
<td>Delete a policy.</td>
<td>Delete a policy by selecting the policy to delete in the Policy List, then clicking the Delete button.</td>
<td>“How to Delete a Kerberos Policy” on page 504</td>
</tr>
</tbody>
</table>

▼ How to View the List of Kerberos Policies

An example of the command-line equivalent follows this procedure.

1 If necessary, start the SEAM Tool.
See “How to Start the SEAM Tool” on page 482 for more information.

```bash
$ /usr/sbin/gkadmin
```

2 Click the Policies tab.
The list of policies is displayed.
3 Display a specific policy or a sublist of policies.

Type a filter string in the Filter field, and press Return. If the filter succeeds, the list of policies that match the filter is displayed.

The filter string must consist of one or more characters. Because the filter mechanism is case sensitive, you need to use the appropriate uppercase and lowercase letters for the filter. For example, if you type the filter string ge, the filter mechanism displays only the policies with the ge string in them (for example, george or edge).

If you want to display the entire list of policies, click Clear Filter.
Example 25–9  Viewing the List of Kerberos Policies (Command Line)

In the following example, the `list_policies` command of `kadmin` is used to list all the policies that match `*user*`. Wildcards can be used with the `list_policies` command.

```
  kadmin: list_policies *user*
testuser
genuser
kadmin: quit
```

▼ How to View a Kerberos Policy's Attributes

An example of the command-line equivalent follows this procedure.

1  If necessary, start the SEAM Tool.
   See “How to Start the SEAM Tool” on page 482 for more information.
   ```
   $ /usr/sbin/gkadmin
   ```

2  Click the Policies tab.

3  Select the policy in the list that you want to view, then click Modify.
   The Policy Details panel is displayed.

4  When you are finished viewing, click Cancel.

Example 25–10  Viewing a Kerberos Policy's Attributes

The following example shows the Policy Details panel when you are viewing the test policy.
In the following example, the `get_policy` command of `kadmin` is used to view the attributes of the `enguser` policy.

```
  kadmin: get_policy enguser
Policy: enguser
  Maximum password life: 2592000
  Minimum password life: 0
  Minimum password length: 8
  Minimum number of password character classes: 2
  Number of old keys kept: 3
```

![Policy Details](image)
Reference count: 0
kadmin: quit

The Reference count is the number of principals that use this policy.

▼ **How to Create a New Kerberos Policy**

An example of the command-line equivalent follows this procedure.

1. **If necessary, start the SEAM Tool.**
   
   See "How to Start the SEAM Tool" on page 482 for more information.
   
   ```
   $ /usr/sbin/gkadmin
   ```

2. **Click the Policies tab.**

3. **Click New.**

   The Policy Details panel is displayed.

4. **Specify a name for the policy in the Policy Name field.**

   The policy name is mandatory.

5. **Specify values for the policy’s attributes.**

   Choose Context-Sensitive Help from the Help menu for information about the various attributes in this window. Or, go to Table 25–5 for all the policy attribute descriptions.

6. **Click Save to save the policy, or click Done.**

**Example 25–12 Creating a New Kerberos Policy**

In the following example, a new policy called build11 is created. The Minimum Password Classes is set to 3.
In the following example, the `add_policy` command of `kadmin` is used to create the `build11` policy. This policy requires at least 3 character classes in a password.

```
$ kadmin
kadmin: add_policy -minclasses 3 build11
kadmin: quit
```
How to Duplicate a Kerberos Policy

This procedure explains how to use all or some of the attributes of an existing policy to create a new policy. No command-line equivalent exists for this procedure.

1. **If necessary, start the SEAM Tool.**
   See “How to Start the SEAM Tool” on page 482 for more information.
   
   ```
   $ /usr/sbin/gkadmin
   ```

2. **Click the Policies tab.**

3. **Select the policy in the list that you want to duplicate, then click Duplicate.**
   The Policy Details panel is displayed. All the attributes of the selected policy are duplicated, except for the Policy Name field, which is empty.

4. **Specify a name for the duplicated policy in the Policy Name field.**
   The policy name is mandatory. To make an exact duplicate of the policy you selected, skip to Step 6.

5. **Specify different values for the policy’s attributes.**
   Choose Context-Sensitive Help from the Help menu for information about the various attributes in this window. Or, go to Table 25–5 for all the policy attribute descriptions.

6. **Click Save to save the policy, or click Done.**

How to Modify a Kerberos Policy

An example of the command-line equivalent follows this procedure.

1. **If necessary, start the SEAM Tool.**
   See "How to Start the SEAM Tool" on page 482 for details.
   
   ```
   $ /usr/sbin/gkadmin
   ```

2. **Click the Policies tab.**

3. **Select the policy in the list that you want to modify, then click Modify.**
   The Policy Details panel is displayed.

4. **Modify the policy’s attributes.**
   Choose Context-Sensitive Help from the Help menu for information about the various attributes in this window. Or, go to Table 25–5 for all the policy attribute descriptions.
Note – You cannot modify a policy's name. To rename a policy, you must duplicate the policy, specify a new name for it, save it, and then delete the old policy.

5 Click Save to save the policy, or click Done.

Example 25–14 Modifying a Kerberos Policy (Command Line)
In the following example, the modify_policy command of kadmin is used to modify the minimum length of a password to five characters for the build11 policy.

```
$ kadmin
kadmin: modify_policy -minlength 5 build11
kadmin: quit
```

▼ How to Delete a Kerberos Policy
An example of the command-line equivalent follows this procedure.

Note – Before you delete a policy, you must cancel the policy from all principals that are currently using it. To do so, you need to modify the principals’ Policy attribute. The policy cannot be deleted if any principal is using it.

1 If necessary, start the SEAM Tool.
See “How to Start the SEAM Tool” on page 482 for more information.

```
$ /usr/sbin/gkadmin
```

2 Click the Policies tab.

3 Select the policy in the list that you want to delete, then click Delete.
After you confirm the deletion, the policy is deleted.

Example 25–15 Deleting a Kerberos Policy (Command Line)
In the following example, the delete_policy command of the kadmin command is used to delete the build11 policy.

```
    kadmin: delete_policy build11
    Are you sure you want to delete the policy "build11"? (yes/no): yes
    kadmin: quit
```

Example 25–14 Modifying a Kerberos Policy (Command Line)
In the following example, the modify_policy command of kadmin is used to modify the minimum length of a password to five characters for the build11 policy.

```
$ kadmin
kadmin: modify_policy -minlength 5 build11
kadmin: quit
```

▼ How to Delete a Kerberos Policy
An example of the command-line equivalent follows this procedure.

Note – Before you delete a policy, you must cancel the policy from all principals that are currently using it. To do so, you need to modify the principals’ Policy attribute. The policy cannot be deleted if any principal is using it.

1 If necessary, start the SEAM Tool.
See “How to Start the SEAM Tool” on page 482 for more information.

```
$ /usr/sbin/gkadmin
```

2 Click the Policies tab.

3 Select the policy in the list that you want to delete, then click Delete.
After you confirm the deletion, the policy is deleted.

Example 25–15 Deleting a Kerberos Policy (Command Line)
In the following example, the delete_policy command of the kadmin command is used to delete the build11 policy.

```
    kadmin: delete_policy build11
    Are you sure you want to delete the policy "build11"? (yes/no): yes
    kadmin: quit
```
Before you delete a policy, you must cancel the policy from all principals that are currently using it. To do so, you need to use the `modify_principal -policy` command of `kadmin` on the affected principals. The `delete_policy` command fails if the policy is in use by a principal.

SEAM Tool Reference

This section provides descriptions of each panel in the SEAM Tool. Also, information about using limited privileges with SEAM Tool are provided.

SEAM Tool Panel Descriptions

This section provides descriptions for each principal and policy attribute that you can either specify or view in the SEAM Tool. The attributes are organized by the panel in which they are displayed.

<table>
<thead>
<tr>
<th>TABLE 25–2 Attributes for the Principal Basics Panel of the SEAM Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Attribute</strong></td>
</tr>
<tr>
<td>Principal Name</td>
</tr>
<tr>
<td>Password</td>
</tr>
<tr>
<td>Policy</td>
</tr>
<tr>
<td>Account Expires</td>
</tr>
<tr>
<td>Last Principal Change</td>
</tr>
<tr>
<td>Last Changed By</td>
</tr>
<tr>
<td>Comments</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE 25–3 Attributes for the Principal Details Panel of the SEAM Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Attribute</strong></td>
</tr>
<tr>
<td>Last Success</td>
</tr>
<tr>
<td>Last Failure</td>
</tr>
<tr>
<td>Failure Count</td>
</tr>
<tr>
<td>Last Password Change</td>
</tr>
</tbody>
</table>
### TABLE 25–3  Attributes for the Principal Details Panel of the SEAM Tool  
(Continued)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Password Expires</td>
<td>The date and time when the principal’s current password expires.</td>
</tr>
<tr>
<td>Key Version</td>
<td>The key version number for the principal. This attribute is normally changed only when a password has been compromised.</td>
</tr>
<tr>
<td>Maximum Lifetime (seconds)</td>
<td>The maximum length of time for which a ticket can be granted for the principal (without renewal).</td>
</tr>
<tr>
<td>Maximum Renewal (seconds)</td>
<td>The maximum length of time for which an existing ticket can be renewed for the principal.</td>
</tr>
</tbody>
</table>

### TABLE 25–4  Attributes of the Principal Flags Panel of the SEAM Tool

<table>
<thead>
<tr>
<th>Attribute (Radio Buttons)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disable Account</td>
<td>When checked, prevents the principal from logging in. This attribute provides an easy way to temporarily freeze a principal account.</td>
</tr>
<tr>
<td>Require Password Change</td>
<td>When checked, expires the principal’s current password, which forces the user to use the kpasswd command to create a new password. This attribute is useful if a security breach occurs, and you need to make sure that old passwords are replaced.</td>
</tr>
<tr>
<td>Allow Postdated Tickets</td>
<td>When checked, allows the principal to obtain postdated tickets. For example, you might need to use postdated tickets for cron jobs that must run after hours, but you cannot obtain tickets in advance because of short ticket lifetimes.</td>
</tr>
<tr>
<td>Allow Forwardable Tickets</td>
<td>When checked, allows the principal to obtain forwardable tickets. Forwardable tickets are tickets that are forwarded to the remote host to provide a single-sign-on session. For example, if you are using forwardable tickets and you authenticate yourself through ftp or rsh, then other services, such as NFS services, are available without your being prompted for another password.</td>
</tr>
<tr>
<td>Allow Renewable Tickets</td>
<td>When checked, allows the principal to obtain renewable tickets. A principal can automatically extend the expiration date or time of a ticket that is renewable (rather than having to get a new ticket after the first ticket expires). Currently, the NFS service is the ticket service that can renew tickets.</td>
</tr>
<tr>
<td>Allow Proxiable Tickets</td>
<td>When checked, allows the principal to obtain proxiable tickets. A proxiable ticket is a ticket that can be used by a service on behalf of a client to perform an operation for the client. With a proxiable ticket, a service can take on the identity of a client and obtain a ticket for another service. However, the service cannot obtain a ticket-granting ticket (TGT).</td>
</tr>
<tr>
<td>Allow Service Tickets</td>
<td>When checked, allows service tickets to be issued for the principal. You should not allow service tickets to be issued for the kadmin/hostname and changepw/hostname principals. This practice ensures that only these principals can update the KDC database.</td>
</tr>
</tbody>
</table>
### TABLE 25–4 Attributes of the Principal Flags Panel of the SEAM Tool (Continued)

<table>
<thead>
<tr>
<th>Attribute (Radio Buttons)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allow TGT-Based Authentication</td>
<td>When checked, allows the service principal to provide services to another principal. More specifically, this attribute allows the KDC to issue a service ticket for the service principal. This attribute is valid only for service principals. When unchecked, service tickets cannot be issued for the service principal.</td>
</tr>
<tr>
<td>Allow Duplicate Authentication</td>
<td>When checked, allows the user principal to obtain service tickets for other user principals. This attribute is valid only for user principals. When unchecked, the user principal can still obtain service tickets for service principals, but not for other user principals.</td>
</tr>
<tr>
<td>Required Preauthentication</td>
<td>When checked, the KDC will not send a requested ticket-granting ticket (TGT) to the principal until the KDC can authenticate (through software) that the principal is really the principal that is requesting the TGT. This preauthentication is usually done through an extra password, for example, from a DES card. When unchecked, the KDC does not need to preauthenticate the principal before the KDC sends a requested TGT to the principal.</td>
</tr>
<tr>
<td>Required Hardware Authentication</td>
<td>When checked, the KDC will not send a requested ticket-granting ticket (TGT) to the principal until the KDC can authenticate (through hardware) that the principal is really the principal that is requesting the TGT. Hardware preauthentication can occur, for example, on a Java ring reader. When unchecked, the KDC does not need to preauthenticate the principal before the KDC sends a requested TGT to the principal.</td>
</tr>
</tbody>
</table>

### TABLE 25–5 Attributes for the Policy Basics Pane of the SEAM Tool

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy Name</td>
<td>The name of the policy. A policy is a set of rules that govern a principal’s password and tickets. If you are modifying a policy, you cannot edit its name.</td>
</tr>
<tr>
<td>Minimum Password Length</td>
<td>The minimum length for the principal’s password.</td>
</tr>
<tr>
<td>Minimum Password Classes</td>
<td>The minimum number of different character types that are required in the principal’s password. For example, a minimum classes value of 2 means that the password must have at least two different character types, such as letters and numbers (hi2mom). A value of 3 means that the password must have at least three different character types, such as letters, numbers, and punctuation (hi2mom!). And so on. A value of 1 sets no restriction on the number of password character types.</td>
</tr>
<tr>
<td>Saved Password History</td>
<td>The number of previous passwords that have been used by the principal, and a list of the previous passwords that cannot be reused.</td>
</tr>
<tr>
<td>Minimum Password Lifetime (seconds)</td>
<td>The minimum length of time that the password must be used before it can be changed.</td>
</tr>
</tbody>
</table>
### Using the SEAM Tool With Limited Kerberos Administration Privileges

All features of the SEAM Administration Tool are available if your admin principal has all the privileges to administer the Kerberos database. However, you might have limited privileges, such as only being allowed to view the list of principals or to change a principal's password. With limited Kerberos administration privileges, you can still use the SEAM Tool. However, various parts of the SEAM Tool change based on the Kerberos administration privileges that you do not have. Table 25–6 shows how the SEAM Tool changes based on your Kerberos administration privileges.

The most visual change to the SEAM Tool occurs when you don’t have the list privilege. Without the list privilege, the List panels do not display the list of principals and policies for you to manipulate. Instead, you must use the Name field in the List panels to specify a principal or a policy that you want to manipulate.

If you log into the SEAM Tool, and you do not have sufficient privileges to perform tasks with it, the following message displays and you are sent back to the SEAM Administration Login window:

**Insufficient privileges to use gkadmin: ADMCIL. Please try using another principal.**

To change the privileges for a principal so that it can administer the Kerberos database, go to “How to Modify the Kerberos Administration Privileges” on page 495.

### Table 25–6 Using the SEAM Tool With Limited Kerberos Administration Privileges

<table>
<thead>
<tr>
<th>Disallowed Privilege</th>
<th>How the SEAM Tool Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>a (add)</td>
<td>The Create New and Duplicate buttons are unavailable in the Principal List and Policy List panels. Without the add privilege, you cannot create new principals or policies, or duplicate them.</td>
</tr>
<tr>
<td>d (delete)</td>
<td>The Delete button is unavailable in the Principal List and Policy List panels. Without the delete privilege, you cannot delete principals or policies.</td>
</tr>
</tbody>
</table>
TABLE 25–6 Using the SEAM Tool With Limited Kerberos Administration Privileges  (Continued)

<table>
<thead>
<tr>
<th>Disallowed Privilege</th>
<th>How the SEAM Tool Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>m (modify)</td>
<td>The Modify button is unavailable in the Principal List and Policy List panels. Without the modify privilege, you cannot modify principals or policies. Also, with the Modify button unavailable, you cannot modify a principal’s password, even if you have the change password privilege.</td>
</tr>
<tr>
<td>c (change password)</td>
<td>The Password field in the Principal Basics panel is read only and cannot be changed. Without the change password privilege, you cannot modify a principal’s password. Note that even if you have the change password privilege, you must also have the modify privilege to change a principal’s password.</td>
</tr>
<tr>
<td>i (inquiry to database)</td>
<td>The Modify and Duplicate buttons are unavailable in the Principal List and Policy List panels. Without the inquiry privilege, you cannot modify or duplicate a principal or a policy. Also, with the Modify button unavailable, you cannot modify a principal’s password, even if you have the change password privilege.</td>
</tr>
<tr>
<td>l (list)</td>
<td>The list of principals and policies in the List panels are unavailable. Without the list privilege, you must use the Name field in the List panels to specify the principal or the policy that you want to manipulate.</td>
</tr>
</tbody>
</table>

Administering Keytab Files

Every host that provides a service must have a local file, called a keytab (short for “key table”). The keytab contains the principal for the appropriate service, called a service key. A service key is used by a service to authenticate itself to the KDC and is known only by Kerberos and the service itself. For example, if you have a Kerberized NFS server, that server must have a keytab file that contains its nfs service principal.

To add a service key to a keytab file, you add the appropriate service principal to a host’s keytab file by using the ktadd command of kadmin. Because you are adding a service principal to a keytab file, the principal must already exist in the Kerberos database so that kadmin can verify its existence. On application servers that provide Kerberized services, the keytab file is located at /etc/krb5/krb5.keytab, by default.

A keytab is analogous to a user’s password. Just as it is important for users to protect their passwords, it is equally important for application servers to protect their keytab files. You should always store keytab files on a local disk, and make them readable only by the root user. Also, you should never send a keytab file over an unsecured network.

There is also a special instance in which to add a root principal to a host’s keytab file. If you want a user on the Kerberos client to mount Kerberized NFS file systems that require
root-equivalent access, you must add the client’s root principal to the client’s keytab file. Otherwise, users must use the `kinit` command as root to obtain credentials for the client’s root principal whenever they want to mount a Kerberized NFS file system with root access, even when they are using the automounter.

Another command that you can use to administer keytab files is the `ktutil` command. This interactive command enables you to manage a local host’s keytab file without having Kerberos administration privileges, because `ktutil` doesn’t interact with the Kerberos database as `kadmin` does. So, after a principal is added to a keytab file, you can use `ktutil` to view the keylist in a keytab file or to temporarily disable authentication for a service.

**Note** – When you change a principal in a keytab file using the `ktadd` command in `kadmin`, a new key is generated and added to the keytab file.

## Administering Keytab Files (Task Map)

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add a service principal to a keytab file.</td>
<td>Use the <code>ktadd</code> command of <code>kadmin</code> to add a service principal to a keytab file.</td>
<td>&quot;How to Add a Kerberos Service Principal to a Keytab File&quot; on page 510</td>
</tr>
<tr>
<td>Remove a service principal from a keytab file.</td>
<td>Use the <code>ktremove</code> command of <code>kadmin</code> to remove a service from a keytab file.</td>
<td>&quot;How to Remove a Service Principal From a Keytab File&quot; on page 512</td>
</tr>
<tr>
<td>Display the keylist (list of principals) in a keytab file.</td>
<td>Use the <code>ktutil</code> command to display the keylist in a keytab file.</td>
<td>&quot;How to Display the Keylist (Principals) in a Keytab File&quot; on page 513</td>
</tr>
<tr>
<td>Temporarily disable authentication for a service on a host.</td>
<td>This procedure is a quick way to temporarily disable authentication for a service on a host without requiring <code>kadmin</code> privileges. Before you use <code>ktutil</code> to delete the service principal from the server’s keytab file, copy the original keytab file to a temporary location. When you want to enable the service again, copy the original keytab file back to its proper location.</td>
<td>&quot;How to Temporarily Disable Authentication for a Service on a Host&quot; on page 513</td>
</tr>
</tbody>
</table>

### How to Add a Kerberos Service Principal to a Keytab File

1. **Make sure that the principal already exists in the Kerberos database.**  
See "How to View the List of Kerberos Principals" on page 485 for more information.
2  Become superuser on the host that needs a principal added to its keytab file.

3  Start the kadmin command.
   
   # /usr/sbin/kadmin

4  Add a principal to a keytab file by using the ktadd command.
   
   kadmin: ktadd [-e enctype] [-k keytab] [-q] [principal | -glob principal-exp]
   
   -e enctype  Overrides the list of encryption types defined in the krb5.conf file.
   -k keytab  Specifies the keytab file. By default, /etc/krb5/krb5.keytab is used.
   -q  Displays less verbose information.
   principal  Specifies the principal to be added to the keytab file. You can add the following service principals: host, root, nfs, and ftp.
   -glob principal-exp  Specifies the principal expressions. All principals that match the principal-exp are added to the keytab file. The rules for principal expression are the same as for the list_principals command of kadmin.

5  Quit the kadmin command.
   
   kadmin: quit

Example 25–16  Adding a Service Principal to a Keytab File

In the following example, denver’s host principal is added to denver’s keytab file, so that the KDC can authenticate denver’s network services.

denver # /usr/sbin/kadmin
kadmin: ktadd host/denver.example.com
   Entry for principal host/denver.example.com with kvno 3, encryption type AES-256 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
   Entry for principal host/denver.example.com with kvno 3, encryption type AES-128 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
   Entry for principal host/denver.example.com with kvno 3, encryption type Triple DES cbc mode with HMAC/shal added to keytab WRFILE:/etc/krb5/krb5.keytab.
   Entry for principal host/denver.example.com with kvno 3, encryption type ArcFour with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
   Entry for principal host/denver.example.com with kvno 3, encryption type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.

kadmin: quit
How to Remove a Service Principal From a Keytab File

1. Become superuser on the host with a service principal that must be removed from its keytab file.

2. Start the kadmin command.
   
   ```
   # /usr/sbin/kadmin
   ```

3. (Optional) To display the current list of principals (keys) in the keytab file, use the ktutil command.
   
   See “How to Display the Keylist (Principals) in a Keytab File” on page 513 for detailed instructions.

4. Remove a principal from the keytab file by using the ktremove command.

   ```
   kadmin: ktremove [-k keytab] [-q] [principal [kvno | all | old ]
   ```

   - `k keytab` Specifies the keytab file. By default, `/etc/krb5/krb5.keytab` is used.
   - `-q` Displays less verbose information.
   - `principal` Specifies the principal to be removed from the keytab file.
   - `kvno` Removes all entries for the specified principal whose key version number matches `kvno`.
   - `all` Removes all entries for the specified principal.
   - `old` Removes all entries for the specified principal, except those principals with the highest key version number.

5. Quit the kadmin command.

   ```
   kadmin: quit
   ```

Example 25–17  Removing a Service Principal From a Keytab File

In the following example, denver’s host principal is removed from denver’s keytab file.

```
denver # /usr/sbin/kadmin
kadmin: ktremove host/denver.example.com@EXAMPLE.COM
kadmin: Entry for principal host/denver.example.com@EXAMPLE.COM with kvno 3
    removed from keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin: quit
```
How to Display the Keylist (Principals) in a Keytab File

1. Become superuser on the host with the keytab file.

Note – Although you can create keytab files that are owned by other users, using the default location for the keytab file requires root ownership.

2. Start the `ktutil` command.
   
   # /usr/bin/ktutil

3. Read the keytab file into the keylist buffer by using the `read_kt` command.
   
   `ktutil`: `read_kt` `keytab`

4. Display the keylist buffer by using the `list` command.
   
   `ktutil`: `list`
   
   The current keylist buffer is displayed.

5. Quit the `ktutil` command.
   
   `ktutil`: `quit`

Example 25–18  Displaying the Keylist (Principals) in a Keytab File

The following example displays the keylist in the `/etc/krb5/krb5.keytab` file on the `denver` host.

```
denver # /usr/bin/ktutil
   ktutil: `read_kt` `/etc/krb5/krb5.keytab`
   ktutil: `list`
   slot KVNO Principal
   ---- ---- ---------------------------------------
   1  5 host/denver@EXAMPLE.COM

   ktutil: `quit`
```

How to Temporarily Disable Authentication for a Service on a Host

At times, you might need to temporarily disable the authentication mechanism for a service, such as `rlogin` or `ftp`, on a network application server. For example, you might want to stop users from logging in to a system while you are performing maintenance procedures. The `ktutil` command enables you to accomplish this task by removing the service principal from
the server’s keytab file, without requiring kadmin privileges. To enable authentication again, you just need to copy the original keytab file that you saved back to its original location.

**Note** – By default, most services are set up to require authentication. If a service is not set up to require authentication, then the service still works, even if you disable authentication for the service.

1 **Become superuser on the host with the keytab file.**

   **Note** – Although you can create keytab files that are owned by other users, using the default location for the keytab file requires root ownership.

2 **Save the current keytab file to a temporary file.**

3 **Start the ktutil command.**

   ```
   # /usr/bin/ktutil
   ```

4 **Read the keytab file into the keylist buffer by using the read_kt command.**

   ```
   ktutil: read_kt keytab
   ```

5 **Display the keylist buffer by using the list command.**

   ```
   ktutil: list
   ```

   The current keylist buffer is displayed. Note the slot number for the service that you want to disable.

6 **To temporarily disable a host’s service, remove the specific service principal from the keylist buffer by using the delete_entry command.**

   ```
   ktutil: delete_entry slot-number
   ```

   Where slot-number specifies the slot number of the service principal to be deleted, which is displayed by the list command.

7 **Write the keylist buffer to a new keytab file by using the write_kt command.**

   ```
   ktutil: write_kt new-keytab
   ```

8 **Quit the ktutil command.**

   ```
   ktutil: quit
   ```

9 **Move the new keytab file.**

   ```
   # mv new-keytab keytab
   ```
When you want to re-enable the service, copy the temporary (original) keytab file back to its original location.

Example 25–19  Temporarily Disabling a Service on a Host

In the following example, the host service on the denver host is temporarily disabled. To re-enable the host service on denver, you would copy the krb5.keytab.temp file to the /etc/krb5/krb5.keytab file.

denver # cp /etc/krb5/krb5.keytab /etc/krb5/krb5.keytab.temp
denver # /usr/bin/ktutil
  ktutil:read_kt /etc/krb5/krb5.keytab
  ktutil:list
  slot KVNO Principal
     --------------------
     1     8  root/denver@EXAMPLE.COM
     2      5  host/denver@EXAMPLE.COM
  ktutil:delete_entry 2
  ktutil:list
  slot KVNO Principal
     --------------------
     1     8  root/denver@EXAMPLE.COM
  ktutil:write_kt /etc/krb5/new.krb5.keytab
  ktutil:quit
denver # cp /etc/krb5/new.krb5.keytab /etc/krb5/krb5.keytab
This chapter is intended for anyone on a system with the Kerberos service configured on it. This chapter explains how to use the "Kerberized" commands and services that are provided. You should already be familiar with these commands (in their non-Kerberized versions) before you read about them here.

Because this chapter is intended for the general reader, it includes information on tickets: obtaining, viewing, and destroying them. This chapter also includes information on choosing or changing a Kerberos password.

This is a list of the information in this chapter:

- “Kerberos Ticket Management” on page 517
- “Kerberos Password Management” on page 521
- “Kerberos User Commands” on page 526

For an overview of the Solaris Kerberos product, see Chapter 21, “Introduction to the Kerberos Service.”

**Kerberos Ticket Management**

This section explains how to obtain, view, and destroy tickets. For an introduction to tickets, see “How the Kerberos Service Works” on page 360.

**Do You Need to Worry About Tickets?**

With any of the SEAM releases or the Solaris 10 release installed, Kerberos is built into the `login` command, and you will obtain tickets automatically when you log in. The Kerberized commands `rsh`, `rcp`, `rdist`, `telnet`, and `rlogin` are usually set up to forward copies of your tickets to the other machines, so you don’t have to explicitly ask for tickets to get access to those machines. Your configuration might not include this automatic forwarding, but it is the default
Creating a Kerberos Ticket

Normally, if PAM is configured properly, a ticket is created automatically when you log in, and you need not do anything special to obtain a ticket. However, you might need to create a ticket if your ticket expires. Also, you might need to use a different principal besides your default principal, for example, if you use `rlogin -l` to log in to a machine as someone else.

To create a ticket, use the `kinit` command.

```
% /usr/bin/kinit
```

The `kinit` command prompts you for your password. For the full syntax of the `kinit` command, see the `kinit(1)` man page.

**EXAMPLE 26–1  Creating a Kerberos Ticket**

This example shows a user, `jennifer`, creating a ticket on her own system.

```
% kinit
Password for jennifer@ENG.EXAMPLE.COM: <Type password>
```

Here, the user `david` creates a ticket that is valid for three hours with the `-l` option.

```
% kinit -l 3h david@EXAMPLE.ORG
Password for david@EXAMPLE.ORG: <Type password>
```

This example shows the user `david` creating a forwardable ticket (with the `-f` option) for himself. With this forwardable ticket, he can, for example, log in to a second system, and then `telnet` to a third system.

```
% kinit -f david@EXAMPLE.ORG
Password for david@EXAMPLE.ORG: <Type password>
```

For more information on how forwarding tickets works, see “Forwarding Kerberos Tickets” on page 529 and “Types of Tickets” on page 538.
Viewing Kerberos Tickets

Not all tickets are alike. One ticket might, for example, be forwardable. Another ticket might be postdated. While a third ticket might be both forwardable and postdated. You can see which tickets you have, and what their attributes are, by using the klist command with the -f option:

```
% /usr/bin/klist -f
```

The following symbols indicate the attributes that are associated with each ticket, as displayed by klist:

A Preauthenticated
D Postdatable
d Postdated
F Forwardable
f Forwarded
I Initial
i Invalid
P Proxiable
p Proxy
R Renewable

“Types of Tickets” on page 538 describes the various attributes that a ticket can have.

**EXAMPLE 26–2  Viewing Kerberos Tickets**

This example shows that the user jennifer has an initial ticket, which is forwardable (F) and postdated (d), but not yet validated (i).

```
% /usr/bin/klist -f
Ticket cache: /tmp/krb5cc_74287
Default principal: jennifer@ENG.EXAMPLE.COM

Valid starting Expires Service principal
09 Mar 04 15:09:51 09 Mar 04 21:09:51 nfs/EXAMPLE.SUN.COM@EXAMPLE.SUN.COM
renew until 10 Mar 04 15:12:51, Flags: Fdi
```

The following example shows that the user david has two tickets that were forwarded (f) to his host from another host. The tickets are also forwardable (F).
EXAMPLE 26-2  Viewing Kerberos Tickets  (Continued)

% klist -f
Ticket cache: /tmp/krb5cc_74287
Default principal: david@EXAMPLE.SUN.COM

Valid starting  Expires  Service principal
07 Mar 04 06:09:51 09 Mar 04 23:33:51 host/EXAMPLE.COM@EXAMPLE.COM
  renew until 10 Mar 04 17:09:51, Flags: FF

Valid starting  Expires  Service principal
08 Mar 04 08:09:51 09 Mar 04 12:54:51 nfs/EXAMPLE.COM@EXAMPLE.COM
  renew until 10 Mar 04 15:22:51, Flags: FF

The following example shows how to display the encryption types of the session key and the
票 by using the -e option. The -a option is used to map the host address to a host name if the
name service can do the conversion.

% klist -fea
Ticket cache: /tmp/krb5cc_74287
Default principal: david@EXAMPLE.SUN.COM

Valid starting  Expires  Service principal
07 Mar 04 06:09:51 09 Mar 04 23:33:51 krbtgt/EXAMPLE.COM@EXAMPLE.COM
  renew until 10 Mar 04 17:09:51, Flags: FRIA
  Etype(skey, tkt): DES cbc mode with RSA-MD5, DES cbc mode with CRC-32
  Addresses: client.example.com

Destroying Kerberos Tickets

If you want to destroy all Kerberos tickets acquired during your current session, use the
kdestroy command. The command destroys you credential cache, which destroys all your
credentials and tickets. While this is not usually necessary, running kdestroy reduces the
chance of the credential cache being compromised during times that you are not logged in.

To destroy your tickets, use the kdestroy command.

% /usr/bin/kdestroy

The kdestroy command destroys all your tickets. You cannot use this command to selectively
destroy a particular ticket.

If you are going to be away from your system and are concerned about an intruder using your
permissions, you should use either kdestroy or a screen saver that locks the screen.
Kerberos Password Management

With the Kerberos service configured, you now have two passwords: your regular Solaris password and a Kerberos password. You can make both passwords the same, or they can be different.

Advice on Choosing a Password

Your password can include almost any character that you can type. The main exceptions are the Control keys and the Return key. A good password is a password that you can remember readily, but no one else can easily guess. Examples of bad passwords include the following:

- Words that can be found in a dictionary
- Any common or popular name
- The name of a famous person or character
- Your name or user name in any form (for example: your name spelled backward, repeated twice, and so forth)
- A spouse’s name, child’s name, or pet’s name
- Your birth date or a relative’s birth date
- Your social security number, driver’s license number, passport number, or other similar identifying number
- Any sample password that appears in this manual or any other manual

A good password is at least eight characters long. Moreover, a password should include a mix of characters, such as uppercase and lowercase letters, numbers, and punctuation marks. Examples of passwords that would be good if they didn’t appear in this manual include the following:

- Acronyms, such as “I2LMHinSF” (which is recalled as “I too left my heart in San Francisco”)
- Easy-to-pronounce nonsense words, such as “WumpaBun” or “WangDangdoodle!”
- Deliberately misspelled phrases, such as “6o’cluck” or “RriotGrrrIsRrrule!”

Caution – Don’t use these examples. Passwords that appear in manuals are the first passwords that an intruder will try.
### Changing Your Password

If PAM is properly configured, you can change your Kerberos password in two ways:

- With the usual UNIX `passwd` command. With the Kerberos service configured, the Solaris `passwd` command also automatically prompts for a new Kerberos password.

  The advantage of using `passwd` instead of `kpasswd` is that you can set both UNIX and Kerberos passwords at the same time. However, you generally do not have to change both passwords with `passwd`. Often, you can change only your UNIX password and leave the Kerberos password untouched, or vice versa.

  **Note** – The behavior of `passwd` depends on how the PAM module is configured. You might be required to change both passwords in some configurations. For some sites, the UNIX password must be changed, while other sites require the Kerberos password to change.

- With the `kpasswd` command. `kpasswd` is very similar to `passwd`. One difference is that `kpasswd` changes only Kerberos passwords. You must use `passwd` if you want to change your UNIX password.

  Another difference is that `kpasswd` can change a password for a Kerberos principal that is not a valid UNIX user. For example, `david/admin` is a Kerberos principal, but not an actual UNIX user, so you must use `kpasswd` instead of `passwd`.

After you change your password, it takes some time for the change to propagate through a system (especially over a large network). Depending on how your system is set up, this delay might take anywhere from a few minutes to an hour or more. If you need to get new Kerberos tickets shortly after you change your password, try the new password first. If the new password doesn’t work, try again using the old password.

Kerberos V5 protocol enables system administrators to set criteria about allowable passwords for each user. Such criteria is defined by the `policy` set for each user (or by a default policy). See “Administering Kerberos Policies” on page 497 for more on policies.

For example, suppose that user jennifer’s policy (call it `jenpol`) mandates that passwords be at least eight letters long and include a mix of at least two types of characters. `kpasswd` will therefore reject an attempt to use “sloth” as a password.

```bash
% kpasswd
kpasswd: Changing password for jennifer@ENG.EXAMPLE.COM.
Old password: <Jennifer types her existing password>
kpasswd: jennifer@ENG.EXAMPLE.COM's password is controlled by
the policy jenpol
which requires a minimum of 8 characters from at least 2 classes
(the five classes are lowercase, uppercase, numbers, punctuation,
...}
```
and all other characters).
New password:  <Jennifer types 'sloth'>
New password (again):  <Jennifer re-types 'sloth'>

kpasswd: New password is too short.
Please choose a password which is at least 4 characters long.

Here, Jennifer uses "slothrop49" as a password. "slothrop49" meets the criteria, because it is
over eight letters long and contains two different types of characters (numbers and lowercase
letters).

% kpasswd
kpasswd: Changing password for jennifer@ENG.EXAMPLE.COM.
Old password:  <Jennifer types her existing password>
kpasswd: jennifer@ENG.EXAMPLE.COM's password is controlled by
the policy jenpol
which requires a minimum of 8 characters from at least 2 classes
(the five classes are lowercase, uppercase, numbers, punctuation,
and all other characters).
New password:  <Jennifer types 'slothrop49'>
New password (again):  <Jennifer re-types 'slothrop49'>
Kerberos password changed.

EXAMPLE 26-3  Changing Your Password

In the following example, user david changes both his UNIX password and Kerberos password
with passwd.

% passwd
passwd: Changing password for david
Enter login (NIS+) password:  <Type the current UNIX password>
New password:  <Type the new UNIX password>
Re-enter password:  <Confirm the new UNIX password>
Old KRB5 password:  <Type the current Kerberos password>
New KRB5 password:  <Type the new Kerberos password>
Re-enter new KRB5 password:  <Confirm the new Kerberos password>

Note that passwd asks for both the UNIX password and the Kerberos password. This behavior is
established by the default configuration. In that case, user david must use kpasswd to set his
Kerberos password to something else, as shown next.

This example shows user david changing only his Kerberos password with kpasswd.

% kpasswd
kpasswd: Changing password for david@ENG.EXAMPLE.COM.
Old password:  <Type the current Kerberos password>
New password:  <Type the new Kerberos password>
New password (again):  <Confirm the new Kerberos password>
Kerberos password changed.

In this example, user david changes the password for the Kerberos principal david/admin (which is not a valid UNIX user). He must use kpasswd.

```
% kpasswd david/admin
kpasswd: Changing password for david/admin.
Old password:  <Type the current Kerberos password>
New password:  <Type the new Kerberos password>
New password (again):  <Type the new Kerberos password>
Kerberos password changed.
```

Granting Access to Your Account

If you need to give someone access to log in to your account (as you), you can do so through Kerberos, without revealing your password, by putting a .k5login file in your home directory. A .k5login file is a list of one or more Kerberos principals corresponding to each person for whom you want to grant access. Each principal must be on a separate line.

Suppose that the user david keeps a .k5login file in his home directory that looks like the following:

```
jennifer@ENG.EXAMPLE.COM
joe@EXAMPLE.ORG
```

This file allows the users jennifer and joe to assume david’s identity, provided that they already have Kerberos tickets in their respective realms. For example, jennifer can remotely log in to david’s machine (boston), as him, without having to give his password.
In the case where David's home directory is NFS-mounted, using Kerberos V5 protocols, from another (third) machine, Jennifer must have a forwardable ticket in order to access his home directory. See "Creating a Kerberos Ticket" on page 518 for an example of using a forwardable ticket.

If you will be logging into other machines across a network, you'll want to include your own Kerberos principal in `.k5login` files on those machines.

Using a `.k5login` file is much safer than giving out your password for these reasons:

- You can take access away any time by removing the principal from your `.k5login` file.
- Although users principals named in the `.k5login` file in your home directory have full access to your account on that machine (or sets of machines, if the `.k5login` file is shared, for example, over NFS). However, any Kerberized services will authorize access based on that user's identity, not yours. So Jennifer can log in to Joe's machine and perform tasks there. However, if she uses a Kerberized program such as `ftp` or `rlogin`, she does so as herself.
- Kerberos keeps a log of who obtains tickets, so a system administrator can find out, if necessary, who is capable of using your user identity at a particular time.

One common way to use the `.k5login` file is to put it in `root`'s home directory, giving `root` access for that machine to the Kerberos principals listed. This configuration allows system administrators to become `root` locally, or to log in remotely as `root`, without having to give out the `root` password, and without requiring anyone to type the `root` password over the network.

**EXAMPLE 26-4 Using the `.k5login` File to Grant Access to Your Account**

Suppose Jennifer decides to log in to the machine `boston.example.com` as `root`. Because she has an entry for her principal name in the `.k5login` file in `root`'s home directory on `boston.example.com`, she again does not have to type in her password.
Using the .k5login File to Grant Access to Your Account

(Continued)

% rlogin boston.example.com -l root -x
This rlogin session is using DES encryption for all data transmissions.
Last login: Thu Jun 20 16:20:50 from daffodil
SunOS Release 5.7 (GENERIC) #2: Tue Nov 14 18:09:31 EST 1998
boston[root]%

Kerberos User Commands

Kerberos V5 product is a single-sign-on system, which means that you only have to type your password once. The Kerberos V5 programs do the authenticating (and optional encrypting) for you, because Kerberos has been built into each of a suite of existing, familiar network programs. The Kerberos V5 applications are versions of existing UNIX network programs with Kerberos features added.

For example, when you use a Kerberized program to connect to a remote host, the program, the KDC, and the remote host perform a set of rapid negotiations. When these negotiations are completed, your program has proven your identity on your behalf to the remote host, and the remote host has granted you access.

Note that Kerberized commands try to authenticate with Kerberos first. If Kerberos authentication fails, an error occurs or UNIX authentication is attempted, depending on what options were used with the command. Refer to the Kerberos Security section in each Kerberos command man page for more detailed information.

Overview of Kerberized Commands

The Kerberized network services are programs that connect to another machine somewhere on the Internet. These programs are the following:

- ftp
- rcp
- rdist
- rlogin
- rsh
- ssh
- telnet

These programs have features that transparently use your Kerberos tickets for negotiating authentication and optional encryption with the remote host. In most cases, you'll notice only that you no longer have to type your password to use them, because Kerberos will provide proof of your identity for you.
The Kerberos V5 network programs include options that enable you to do the following:

- Forward your tickets to the another host (if you initially obtained forwardable tickets).
- Encrypt data transmitted between you and the remote host.

**Note** – This section assumes you are already familiar with the non-Kerberos versions of these programs, and highlights the Kerberos functionality added by the Kerberos V5 package. For detailed descriptions of the commands described here, see their respective man pages.

The following Kerberos options have been added to `ftp`, `rcp`, `rlogin`, `rsh`, and `telnet`:

- `-a` Attempts automatic login using your existing tickets. Uses the username as returned by `getlogin()`, unless the name is different from the current user ID. See the `telnet(1)` man page for details.

- `-f` Forwards a non-forwardable ticket to a remote host. This option is mutually exclusive with the `-F` option. They cannot be used together in the same command.

  You’ll want to forward a ticket if you have reason to believe you’ll need to authenticate yourself to other Kerberos-based services on a third host. For example, you might want to remotely log into another machine and then remotely log in from it to a third machine.

  You should definitely use a forwardable ticket if your home directory on the remote host is NFS-mounted using the Kerberos V5 mechanism. Otherwise, you won’t be able to access your home directory. That is, suppose you initially log in to System 1. From System 1, you remotely log in to your home machine, System 2, which mounts your home directory from System 3. Unless you’ve used the `-f` or `-F` option with `rlogin`, you won’t be able to get to your home directory because your ticket can’t be forwarded to System 3.

  By default, `kinit` obtains forwardable ticket-granting tickets (TGTs). However, your configuration might differ in this respect.

  For more information on forwarding tickets, see “Forwarding Kerberos Tickets” on page 529.

- `-F` Forwards a reforwardable copy of your TGT to a remote system. It is similar to `-f`, but it allows for access to a further (say, fourth or fifth) machine. The `-F` option can therefore be regarded as being a superset of the `-f` option. The `-F` option is mutually exclusive with the `-f` option. They cannot be used together in the same command.
For more information on forwarding tickets, see "Forwarding Kerberos Tickets" on page 529.

- \texttt{k realm} Requests tickets for the remote host in the specified \textit{realm}, instead of determining the realm itself using the \texttt{krb5.conf} file.

- \texttt{-K} Uses your tickets to authenticate to the remote host, but does not automatically log in.

- \texttt{-m mechanism} Specifies the GSS-API security mechanism to use, as listed in the \texttt{/etc/gss/mech} file. Defaults to \texttt{kerberos_v5}.

- \texttt{-x} Encrypts this session.

- \texttt{-X auth-type} Disables the \textit{auth-type} type of authentication.

The following table shows which commands have specific options. An "X" indicates that the command has that option.

\begin{table}[ht]
\centering
\begin{tabular}{l|cccc}
\hline
 & ftp & rcp & rlogin & rsh & telnet \\
\hline
-\texttt{a} & & & & & X \\
-\texttt{f} & & & & & X \\
-\texttt{-F} & & & & & X \\
-\texttt{-k} & & & & & X \\
-\texttt{-K} & & & & & X \\
-\texttt{-m} & & & & & X \\
-\texttt{-x} & & & & & X \\
-\texttt{-X} & & & & & X \\
\hline
\end{tabular}
\caption{Kerberos Options for Network Commands}
\end{table}

Additionally, \texttt{ftp} allows the protection level for a session to be set at its prompt:

\begin{itemize}
  \item \texttt{clear} Sets the protection level to "clear" (no protection). This protection level is the default.
  \item \texttt{private} Sets the protection level to "private." Data transmissions are confidentiality-protected and integrity-protected by encryption. The privacy service might not be available to all Kerberos users, however.
  \item \texttt{safe} Sets the protection level to "safe." Data transmissions are integrity-protected by cryptographic checksum.
\end{itemize}
You can also set the protection level at the ftp prompt by typing protect followed by any of the protection levels shown above (clear, private, or safe).

**Forwarding Kerberos Tickets**

As described in “Overview of Kerberized Commands” on page 526, some commands allow you to forward tickets with either the -f or -F option. Forwarding tickets allows you to “chain” your network transactions. You can, for example, remotely log in to one machine and then remotely log in from it to another machine. The -f option allows you to forward a ticket, while the -F option allows you to reforward a forwarded ticket.

In Figure 26–2, the user david obtains a non-forwardable ticket-granting ticket (TGT) with kinit. The ticket is non-forwardable because he did not specify the -f option. In scenario 1, he is able to remotely log in to machine B, but he can go no further. In scenario 2, the rlogin -f command fails because he is attempting to forward a ticket that is non-forwardable.

In actuality, Kerberos configuration files are set up so that kinit obtains forwardable tickets by default. However, your configuration might differ. For the sake of explanation, assume that kinit does not obtain forwardable TGTs unless it is invoked with kinit -f. Notice, by the way, that kinit does not have a -F option. TGTs are either forwardable or not.

In Figure 26–3, the user david obtains forwardable TGTs with kinit -f. In scenario 3, he is able to reach machine C because he uses a forwardable ticket with rlogin. In scenario 4, the second rlogin fails because the ticket is not reforwardable. By using the -F option instead, as in scenario 5, the second rlogin succeeds and the ticket can be reforwarded on to machine D.
Using Kerberized Commands (Examples)

The following examples show how the options to the Kerberized commands work.

**EXAMPLE 26–5 Using the -a, -f, and -x Options With telnet**

In this example, the user *david* has already logged in, and wants to `telnet` to the machine `denver.example.com`. He uses the `-f` option to forward his existing tickets, the `-x` option to encrypt the session, and the `-a` option to perform the login automatically. Because he does not plan to use the services of a third host, he can use `-f` instead of `-F`.

```bash
% telnet -a -f -x denver.example.com
Trying 128.0.0.5...
Connected to denver.example.com. Escape character is '^[].
[ Kerberos V5 accepts you as "david@eng.example.com" ]
[ Kerberos V5 accepted forwarded credentials ]
SunOS 5.9: Tue May 21 00:31:42 EDT 2004 Welcome to SunOS
%
```

Notice that *david's* machine used Kerberos to authenticate him to `denver.example.com`, and logged him in automatically as himself. He had an encrypted session, a copy of his tickets already waiting for him, and he never had to type his password. If he had used a non-Kerberos

**FIGURE 26–3 Using Forwardable Tickets**

Kerberos User Commands
EXAMPLE 26–5 Using the -a, -f, and -x Options With telnet  

(Continued)

version of telnet, he would have been prompted for his password, and it would have been sent over the network unencrypted. If an intruder had been watching network traffic at the time, the intruder would have known David's password.

If you forward your Kerberos tickets, telnet (as well as the other commands discussed here) destroys them when it exits.

EXAMPLE 26–6 Using rlogin With the -F Option

Here, the user Jennifer wants to log in to her own machine, boston.example.com. She forwards her existing tickets with the -F option, and encrypts the session with the -x option. She chooses -F rather than -f because after she is logged in to boston, she might want to perform other network transactions requiring tickets to be reforwarded. Also, because she is forwarding her existing tickets, she does not have to type her password.

```
% rlogin boston.example.com -F -x
This rlogin session is using encryption for all transmissions.
Last login Mon May 19 15:19:49 from daffodil
SunOS Release 5.9 (GENERIC) #2 Tue Nov 14 18:09:3 EST 2003
```

EXAMPLE 26–7 Setting the Protection Level in ftp

Suppose that Joe wants to use ftp to get his mail from the directory ~joe/MAIL from the machine denver.example.com, encrypting the session. The exchange would look like the following:

```
% ftp -f denver.example.com
Connected to denver.example.com
220 denver.example.org FTP server (Version 6.0) ready.
334 Using authentication type GSSAPI; ADAT must follow
GSSAPI accepted as authentication type
GSSAPI authentication succeeded Name (daffodil.example.org:joe)
232 GSSAPI user joe@MELPOMENE.EXAMPLE.COM is authorized as joe
230 User joe logged in.
Remote system type is UNIX.
Using BINARY mode to transfer files.
ftp> protect private
200 Protection level set to Private
ftp> cd ~joe/MAIL
250 CWD command successful.
ftp> get RMAIL
227 Entering Passive Mode (128,0,0,5,16,49)
```
To encrypt the session, joe sets the protection level to private.
This chapter lists many of the files, commands, and daemons that are part of the Kerberos product. In addition, this chapter provides detailed information about how Kerberos authentication works.

This is a list of the reference information in this chapter.

- “Kerberos Files” on page 533
- “Kerberos Commands” on page 535
- “Kerberos Daemons” on page 536
- “Kerberos Terminology” on page 536
- “How the Kerberos Authentication System Works” on page 542
- “Gaining Access to a Service Using Kerberos” on page 542
- “Using Kerberos Encryption Types” on page 545
- “Using the gsscred Table” on page 547
- “Notable Differences Between Solaris Kerberos and MIT Kerberos” on page 547

### Kerberos Files

**TABLE 27–1  Kerberos Files**

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>~/.gkadmin</td>
<td>Default values for creating new principals in the SEAM Administration Tool</td>
</tr>
<tr>
<td>~/.k5login</td>
<td>List of principals that grant access to a Kerberos account</td>
</tr>
<tr>
<td>/etc/krb5/kadm5.acl</td>
<td>Kerberos access control list file, which includes principal names of KDC administrators and their Kerberos administration privileges</td>
</tr>
<tr>
<td>File Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>/etc/krb5/kadm5.keytab</td>
<td>Obsolete: This file was removed in the Solaris Express Community Edition, build 102 release.</td>
</tr>
<tr>
<td>/etc/krb5/kdc.conf</td>
<td>KDC configuration file</td>
</tr>
<tr>
<td>/etc/krb5/kpropd.acl</td>
<td>Kerberos database propagation configuration file</td>
</tr>
<tr>
<td>/etc/krb5/krb5.conf</td>
<td>Kerberos realm configuration file</td>
</tr>
<tr>
<td>/etc/krb5/krb5.keytab</td>
<td>Keytab file for network application servers</td>
</tr>
<tr>
<td>/etc/krb5/warn.conf</td>
<td>Kerberos ticket expiration warning and automatic renewal configuration file</td>
</tr>
<tr>
<td>/etc/pam.conf</td>
<td>PAM configuration file</td>
</tr>
<tr>
<td>/tmp/krb5cc_uid</td>
<td>Default credentials cache, where uid is the decimal UID of the user</td>
</tr>
<tr>
<td>/tmp/ovsec_adm.xxxxxx</td>
<td>Temporary credentials cache for the lifetime of the password changing operation, where xxxxxx is a random string</td>
</tr>
<tr>
<td>/var/krb5/.k5.REALM</td>
<td>KDC stash file, which contains a copy of the KDC master key</td>
</tr>
<tr>
<td>/var/krb5/kadmin.log</td>
<td>Log file for kadmin</td>
</tr>
<tr>
<td>/var/krb5/kdc.log</td>
<td>Log file for the KDC</td>
</tr>
<tr>
<td>/var/krb5/principal</td>
<td>Kerberos principal database</td>
</tr>
<tr>
<td>/var/krb5/principal.kadm5</td>
<td>Kerberos administrative database, which contains policy information</td>
</tr>
<tr>
<td>/var/krb5/principal.kadm5.lock</td>
<td>Kerberos administrative database lock file</td>
</tr>
<tr>
<td>/var/krb5/principal.ok</td>
<td>Kerberos principal database initialization file that is created when the Kerberos database is initialized successfully</td>
</tr>
<tr>
<td>/var/krb5/principal. ulog</td>
<td>Kerberos update log, which contains updates for incremental propagation</td>
</tr>
<tr>
<td>/var/krb5/slave_datatrans</td>
<td>Backup file of the KDC that the kprop_script script uses for propagation</td>
</tr>
<tr>
<td>/var/krb5/slave_datatrans_slave</td>
<td>Temporary dump file that is created when full updates are made to the specified slave</td>
</tr>
</tbody>
</table>
Kerberos Commands

This section lists some commands that are included in the Kerberos product.

**TABLE 27–2  Kerberos Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/usr/bin/ftp</td>
<td>File Transfer Protocol program</td>
</tr>
<tr>
<td>/usr/bin/kdestroy</td>
<td>Destroys Kerberos tickets</td>
</tr>
<tr>
<td>/usr/bin/kinit</td>
<td>Obtains and caches Kerberos ticket-granting tickets</td>
</tr>
<tr>
<td>/usr/bin/klist</td>
<td>Displays current Kerberos tickets</td>
</tr>
<tr>
<td>/usr/bin/kpasswd</td>
<td>Changes a Kerberos password</td>
</tr>
<tr>
<td>/usr/bin/ktutil</td>
<td>Manages Kerberos keytab files</td>
</tr>
<tr>
<td>/usr/bin/rcp</td>
<td>Remote file copy program</td>
</tr>
<tr>
<td>/usr/bin/rdist</td>
<td>Remote file distribution program</td>
</tr>
<tr>
<td>/usr/bin/rlogin</td>
<td>Remote login program</td>
</tr>
<tr>
<td>/usr/bin/rsh</td>
<td>Remote shell program</td>
</tr>
<tr>
<td>/usr/bin/telnet</td>
<td>Kerberized telnet program</td>
</tr>
<tr>
<td>/usr/lib/krb5/kprop</td>
<td>Kerberos database propagation program</td>
</tr>
<tr>
<td>/usr/sbin/gkadmin</td>
<td>Kerberos database administration GUI program, which is used to manage principals and policies</td>
</tr>
<tr>
<td>/usr/sbin/gsscred</td>
<td>Manage gsscred table entries</td>
</tr>
<tr>
<td>/usr/sbin/kadmin</td>
<td>Remote Kerberos database administration program (run with Kerberos authentication), which is used to manage principals, policies, and keytab files</td>
</tr>
<tr>
<td>/usr/sbin/kadmin.local</td>
<td>Local Kerberos database administration program (run without Kerberos authentication and must be run on master KDC), which is used to manage principals, policies, and keytab files</td>
</tr>
<tr>
<td>/usr/sbin/kclient</td>
<td>Kerberos client installation script which is used with or without a installation profile</td>
</tr>
<tr>
<td>/usr/sbin/kdb5_ldap_util</td>
<td>Creates LDAP containers for Kerberos databases</td>
</tr>
<tr>
<td>/usr/sbin/kdb5_util</td>
<td>Creates Kerberos databases and stash files</td>
</tr>
<tr>
<td>/usr/sbin/kgcmgr</td>
<td>Configures Kerberos master and slave KDCs</td>
</tr>
<tr>
<td>/usr/sbin/kproplog</td>
<td>Lists a summary of update entries in the update log</td>
</tr>
</tbody>
</table>
Kerberos Daemons

The following table lists the daemons that the Kerberos product uses.

<table>
<thead>
<tr>
<th>Daemon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/usr/sbin/in.ftpd</td>
<td>File Transfer Protocol daemon</td>
</tr>
<tr>
<td>/usr/lib/krb5/kadmind</td>
<td>Kerberos database administration daemon</td>
</tr>
<tr>
<td>/usr/lib/krb5/kpropd</td>
<td>Kerberos database propagation daemon</td>
</tr>
<tr>
<td>/usr/lib/krb5/krb5kdc</td>
<td>Kerberos ticket processing daemon</td>
</tr>
<tr>
<td>/usr/lib/krb5/ktkt Warnd</td>
<td>Kerberos ticket expiration warning and automatic renewal daemon</td>
</tr>
<tr>
<td>/usr/sbin/in.rlogind</td>
<td>Remote login daemon</td>
</tr>
<tr>
<td>/usr/sbin/in.rshd</td>
<td>Remote shell daemon</td>
</tr>
<tr>
<td>/usr/sbin/in.telnetd</td>
<td>telnet daemon</td>
</tr>
</tbody>
</table>

Kerberos Terminology

The following section presents Kerberos terms and their definitions. These terms are used throughout the Kerberos documentation. To grasp Kerberos concepts, an understanding of these terms is essential.

Kerberos-Specific Terminology

You need to understand the terms in this section in order to administer KDCs.

The Key Distribution Center or KDC is the component of Kerberos that is responsible for issuing credentials. These credentials are created by using information that is stored in the KDC database. Each realm needs at least two KDCs, a master and at least one slave. All KDCs generate credentials, but only the master KDC handles any changes to the KDC database.

A stash file contains the master key for the KDC. This key is used when a server is rebooted to automatically authenticate the KDC before starting the kadmind and krb5kdc commands. Because this file includes the master key, the file and any backups of the file should be kept secure. The file is created with read-only permissions for root. To keep the file secure, do not change the permissions. If the file is compromised, then the key could be used to access or modify the KDC database.
Authentication-Specific Terminology

You need to know the terms in this section to understand the authentication process. Programmers and system administrators should be familiar with these terms.

A client is the software that runs on a user’s workstation. The Kerberos software that runs on the client makes many requests during this process. So, differentiating the actions of this software from the user is important.

The terms server and service are often used interchangeably. To clarify, the term server is used to define the physical system that Kerberos software is running on. The term service corresponds to a particular function that is being supported on a server (for example, ftp or nfs). Documentation often mentions servers as part of a service, but this definition clouds the meaning of the terms. Therefore, the term server refers to the physical system. The term service refers to the software.

The Kerberos product uses two types of keys. One type of key is a password derived key. The password derived key is given to each user principal and is known only to the user and to the KDC. The other type of key used by the Kerberos product is a random key that is not associated with a password and so is not suitable for use by user principals. Random keys are typically used for service principals that have entries in a keytab and session keys generated by the KDC. Service principals can use random keys since the service can access the key in the keytab which allows it to run non-interactively. Session keys are generated by the KDC (and shared between the client and service) to provide secure transactions between a client and a service.

A ticket is an information packet that is used to securely pass the identity of a user to a server or service. A ticket is valid for only a single client and a particular service on a specific server. A ticket contains:

- Principal name of the service
- Principal name of the user
- IP address of the user’s host
- Timestamp
- Value which defines the lifetime of the ticket
- Copy of the session key

All of this data is encrypted in the server’s service key. Note, the KDC issues the ticket embedded in a credential described below. After a ticket has been issued, it can be reused until the ticket expires.

A credential is a packet of information that includes a ticket and a matching session key. The credential is encrypted with the requesting principal’s key. Typically, the KDC generates a credential in response to a ticket request from a client.

An authenticator is information used by the server to authenticate the client user principal. An authenticator includes the principal name of the user, a timestamp, and other data. Unlike a ticket, an authenticator can be used once only, usually when access to a service is requested.
authenticator is encrypted by using the session key shared by the client and server. Typically, the client creates the authenticator and sends it with the server’s or service’s ticket in order to authenticate to the server or service.

### Types of Tickets

Tickets have properties that govern how they can be used. These properties are assigned to the ticket when it is created, although you can modify a ticket’s properties later. For example, a ticket can change from being forwardable to being forwarded. You can view ticket properties with the `klist` command. See “Viewing Kerberos Tickets” on page 519.

Tickets can be described by one or more of the following terms:

- **Forwardable/forwarded**: A forwardable ticket can be sent from one host to another host, obviating the need for a client to reauthenticate itself. For example, if the user david obtains a forwardable ticket while on user jennifer’s machine, he can log in to his own machine without having to get a new ticket (and thus authenticate himself again). See Example 26–1 for an example of a forwardable ticket.

- **Initial**: An initial ticket is a ticket that is issued directly, not based on a ticket-granting ticket. Some services, such as applications that change passwords, can require tickets to be marked initial in order to assure themselves that the client can demonstrate a knowledge of its secret key. An initial ticket indicates that the client has recently authenticated itself, instead of relying on a ticket-granting ticket, which might have been around for a long time.

- **Invalid**: An invalid ticket is a postdated ticket that has not yet become usable. An invalid ticket will be rejected by an application server until it becomes validated. To be validated, a ticket must be presented to the KDC by the client in a ticket-granting service request, with the `VALIDATE` flag set, after its start time has passed.

- **Postdatable/postdated**: A postdated ticket is a ticket that does not become valid until some specified time after its creation. Such a ticket is useful, for example, for batch jobs that are intended to be run late at night, because the ticket, if stolen, cannot be used until the batch job is to be run. When a postdated ticket is issued, it is issued as invalid and remains that way until its start time has passed, and the client requests validation by the KDC. A postdated ticket is normally valid until the expiration time of the ticket-granting ticket. However, if the ticket is marked renewable, its lifetime is normally set to be equal to the duration of the full life of the ticket-granting ticket.
Proxiable/proxy

At times, it is necessary for a principal to allow a service to perform an operation on its behalf. The principal name of the proxy must be specified when the ticket is created. The Solaris release does not support proxiable or proxy tickets.

A proxiable ticket is similar to a forwardable ticket, except that it is valid only for a single service, whereas a forwardable ticket grants the service the complete use of the client’s identity. A forwardable ticket can therefore be thought of as a sort of super-proxy.

Renewable

Because it is a security risk to have tickets with very long lives, tickets can be designated as renewable. A renewable ticket has two expiration times: the time at which the current instance of the ticket expires, and the maximum lifetime for any ticket, which is one week. If a client wants to continue to use a ticket, the client renews it before the first expiration occurs. For example, a ticket can be valid for one hour, with all tickets having a maximum lifetime of 10 hours. If the client that is holding the ticket wants to keep it for more than an hour, the client must renew it within that hour. When a ticket reaches the maximum ticket lifetime (10 hours), it automatically expires and cannot be renewed.

For information on how to view the attributes of tickets, see “Viewing Kerberos Tickets” on page 519.

Ticket Lifetimes

Any time a principal obtains a ticket, including a ticket–granting ticket (TGT), the ticket’s lifetime is set as the smallest of the following lifetime values:

- The lifetime value that is specified by the `-l` option of `kinit`, if `kinit` is used to get the ticket. By default, `kinit` used the maximum lifetime value.
- The maximum lifetime value (`max_life`) that is specified in the `kdc.conf` file.
- The maximum lifetime value that is specified in the Kerberos database for the service principal that provides the ticket. In the case of `kinit`, the service principal is `krbtgt/realm`.
- The maximum lifetime value that is specified in the Kerberos database for the user principal that requests the ticket.

Figure 27–1 shows how a TGT’s lifetime is determined and where the four lifetime values come from. Even though this figure shows how a TGT’s lifetime is determined, basically the same thing happens when any principal obtains a ticket. The only differences are that `kinit` doesn’t provide a lifetime value, and the service principal that provides the ticket provides a maximum lifetime value (instead of the `krbtgt/realm` principal).
The renewable ticket lifetime is also determined from the minimum of four values, but renewable lifetime values are used instead, as follows:

- The renewable lifetime value that is specified by the `-r` option of `kinit`, if `kinit` is used to obtain or renew the ticket.
- The maximum renewable lifetime value (`max_renewable_life`) that is specified in the `kdc.conf` file.
- The maximum lifetime renewable value that is specified in the Kerberos database for the service principal that provides the ticket. In the case of `kinit`, the service principal is `krbtgt/realm`.
- The maximum lifetime renewable value that is specified in the Kerberos database for the user principal that requests the ticket.

**Kerberos Principal Names**

Each ticket is identified by a principal name. The principal name can identify a user or a service. Here are examples of several principal names.
<table>
<thead>
<tr>
<th>Principal Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>changepw/kdc1.example.com@EXAMPLE.COM</td>
<td>A principal for the master KDC server that allows access to the KDC when you are changing passwords.</td>
</tr>
<tr>
<td>clntconfig/admin@EXAMPLE.COM</td>
<td>A principal that is used by the kclient installation utility.</td>
</tr>
<tr>
<td>ftp/boston.example.com@EXAMPLE.COM</td>
<td>A principal used by the ftp service. This principal can be used instead of a host principal.</td>
</tr>
<tr>
<td>host/boston.example.com@EXAMPLE.COM</td>
<td>A principal that is used by the Kerberized applications (klist and kprop, for example) and services (such as ftp and telnet). This principal is called a host or service principal. The principal is used to authenticate NFS mounts. This principal is also used by a client to verify that the TGT that is issued to the client is from the correct KDC.</td>
</tr>
<tr>
<td>K/M@EXAMPLE.COM</td>
<td>The master key name principal. One master key name principal is associated with each master KDC.</td>
</tr>
<tr>
<td>kadmin/history@EXAMPLE.COM</td>
<td>A principal that includes a key used to keep password histories for other principals. Each master KDC has one of these principals.</td>
</tr>
<tr>
<td>kadmin/kdc1.example.com@EXAMPLE.COM</td>
<td>A principal for the master KDC server that allows access to the KDC by using kadmin.</td>
</tr>
<tr>
<td>kadmin/changepw.example.com@EXAMPLE.COM</td>
<td>A principal that is used to accept password change requests from clients that are not running a Solaris release.</td>
</tr>
<tr>
<td><a href="mailto:krbtgt@EXAMPLE.COM">krbtgt@EXAMPLE.COM</a>@EXAMPLE.COM</td>
<td>This principal is used when you generate a ticket-granting ticket.</td>
</tr>
<tr>
<td>krbtgt/EAST.EXAMPLE.COM@WEST.EXAMPLE.COM</td>
<td>This principal is an example of a cross-realm ticket-granting ticket.</td>
</tr>
<tr>
<td>nfs/boston.example.com@EXAMPLE.COM</td>
<td>A principal that is used by the NFS service. This principal can be used instead of a host principal.</td>
</tr>
<tr>
<td>root/boston.example.com@EXAMPLE.COM</td>
<td>A principal that is associated with the root account on a client. This principal is called a root principal and provides root access to NFS mounted file systems.</td>
</tr>
<tr>
<td><a href="mailto:username@EXAMPLE.COM">username@EXAMPLE.COM</a></td>
<td>A principal for a user.</td>
</tr>
<tr>
<td>username/admin@EXAMPLE.COM</td>
<td>An admin principal that can be used to administer the KDC database.</td>
</tr>
</tbody>
</table>
How the Kerberos Authentication System Works

Applications allow you to log in to a remote system if you can provide a ticket that proves your identity, and a matching session key. The session key contains information that is specific to the user and the service that is being accessed. A ticket and session key are created by the KDC for all users when they first log in. The ticket and the matching session key form a credential. While using multiple networking services, a user can gather many credentials. The user needs to have a credential for each service that runs on a particular server. For example, access to the `ftp` service on a server named `boston` requires one credential. Access to the `ftp` service on another server requires its own credential.

The process of creating and storing the credentials is transparent. Credentials are created by the KDC that sends the credential to the requester. When received, the credential is stored in a credential cache.

Gaining Access to a Service Using Kerberos

To access a specific service on a specific server, the user must obtain two credentials. The first credential is for the ticket-granting ticket (known as the TGT). Once the ticket-granting service has decrypted this credential, the service creates a second credential for the server that the user is requesting access to. This second credential can then be used to request access to the service on the server. After the server has successfully decrypted the second credential, then the user is given access. The following sections describe this process in more detail.

Obtaining a Credential for the Ticket-Granting Service

1. To start the authentication process, the client sends a request to the authentication server for a specific user principal. This request is sent without encryption. No secure information is included in the request, so it is not necessary to use encryption.

2. When the request is received by the authentication service, the principal name of the user is looked up in the KDC database. If a principal matches the entry in the database, the authentication service obtains the private key for that principal. The authentication service then generates a session key to be used by the client and the ticket-granting service (call it Session key 1) and a ticket for the ticket-granting service (Ticket 1). This ticket is also known as the ticket-granting ticket (TGT). Both the session key and the ticket are encrypted by using the user’s private key, and the information is sent back to the client.

3. The client uses this information to decrypt Session Key 1 and Ticket 1, by using the private key for the user principal. Because the private key should only be known by the user and the KDC database, the information in the packet should be safe. The client stores the information in the credentials cache.
During this process, a user is normally prompted for a password. If the password the user specifies is the same as the password that was used to build the private key stored in the KDC database, then the client can successfully decrypt the information that is sent by the authentication service. Now the client has a credential to be used with the ticket-granting service. The client is ready to request a credential for a server.

1. Client requests credential for server.

2. Authentication service sends back credential, which includes Session Key 1 and TGT.

3. Client decrypts credential with entered password.

TGT = Ticket-granting ticket
KDC = Key Distribution Center

FIGURE 27–2 Obtaining a Credential for the Ticket-Granting Service

Obtaining a Credential for a Server

1. To request access to a specific server, a client must first have obtained a credential for that server from the authentication service. See “Obtaining a Credential for the Ticket-Granting Service” on page 542. The client then sends a request to the ticket-granting service, which includes the service principal name, Ticket 1, and an authenticator that was encrypted with Session Key 1. Ticket 1 was originally encrypted by the authentication service by using the service key of the ticket-granting service.

2. Because the service key of the ticket-granting service is known to the ticket-granting service, Ticket 1 can be decrypted. The information in Ticket 1 includes Session Key 1, so the ticket-granting service can decrypt the authenticator. At this point, the user principal is authenticated with the ticket-granting service.
3. Once the authentication is successful, the ticket-granting service generates a session key for the user principal and the server (Session Key 2), and a ticket for the server (Ticket 2). Session Key 2 and Ticket 2 are then encrypted by using Session Key 1. Because Session Key 1 is known only to the client and the ticket-granting service, this information is secure and can be safely sent over the network.

4. When the client receives this information packet, the client decrypts the information by using Session Key 1, which it had stored in the credential cache. The client has obtained a credential to be used with the server. Now the client is ready to request access to a particular service on that server.

**Obtaining Access to a Specific Service**

1. To request access to a specific service, the client must first have obtained a credential for the ticket-granting service from the authentication server, and a server credential from the ticket-granting service. See "Obtaining a Credential for the Ticket-Granting Service" on page 542 and "Obtaining a Credential for a Server" on page 543. The client can then send a request to the server including Ticket 2 and another authenticator. The authenticator is encrypted by using Session Key 2.

2. Ticket 2 was encrypted by the ticket-granting service with the service key for the service. Because the service key is known by the service principal, the service can decrypt Ticket 2 and get Session Key 2. Session Key 2 can then be used to decrypt the authenticator. If the authenticator is successfully decrypted, the client is given access to the service.
Using Kerberos Encryption Types

Encryption types identify which cryptographic algorithms and mode to use when cryptographic operations are performed. The aes, des3-cbc-sha1 and rc4-hmac encryption types enable the creation of keys that can be used for higher strength cryptographic operations. These higher strength operations enhance the overall security of the Kerberos service.

**Note** – In releases prior to Solaris 10 8/07 release, the aes256-cts-hmac-sha1-96 encryption type can be used with the Kerberos service if the unbundled Strong Cryptographic packages are installed.

When a client requests a ticket from the KDC, the KDC must use keys whose encryption type is compatible with both the client and the server. While the Kerberos protocol allows the client to request that the KDC use particular encryption types for the client’s part of the ticket reply, the protocol does not allow the server to specify encryption types to the KDC.

**Note** – If you have a master KDC installed that is not running the Solaris 10 release, the slave KDCs must be upgraded to the Solaris 10 release before you upgrade the master KDC. A Solaris 10 master KDC will use the new encryption types, which an older slave will not be able to handle.

The following lists some of the issues that must be considered before you change the encryption types.

- The KDC assumes that the first key/encrypt type associated with the server principal entry in the principal database is supported by the server.
On the KDC, you should make sure that the keys generated for the principal are compatible with the systems on which the principal will be authenticated. By default, the kadmin command creates keys for all supported encryption types. If the systems that the principal is used on do not support this default set of encryption types, then you should restrict the encryption types when creating a principal. You can restrict the encryption types through use of the -e flag in kadmin addprinc or by setting the supported_enctypes parameter in the kdc.conf file to this subset. The supported_enctypes parameter should be used when most of the systems in a Kerberos realm support a subset of the default set of encryption types. Setting supported_enctypes specifies the default set of encryption types kadmin addprinc uses when it creates a principal for a particular realm. As a general rule, it is best to control the encryption types used by Kerberos using one of these two methods.

When determining the encryption types a system supports, consider both the version of Kerberos running on the system as well as the cryptographic algorithms supported by the server application for which a server principal is being created. For example, when creating an nfs/hostname service principal, you should restrict the encryption types to the types supported by the NFS server on that host. Note that in the Solaris 10 release, all supported Kerberos encryption types are also supported by the NFS server.

The master_key_enctype parameter in the kdc.conf file can be used to control the encryption type of the master key that encrypts the entries in the principal database. Do not use this parameter if the KDC principal database has already been created. The master_key_enctype parameter can be used at database creation time to change the default master key encryption type from des - cbc - cbc to a stronger encryption type. Make sure that all slave KDCs support the chosen encryption type and that they have an identical master_key_enctype entry in their kdc.conf when configuring the slave KDCs. Also, make sure that the master_key_enctype is set to one of the encryption types in supported_enctypes, if supported_enctypes is set in kdc.conf. If either of these issues are not handled properly, then the master KDC might not be able to work with the slave KDCs.

On the client, you can control which encryption types the client requests when getting tickets from the KDC through a couple of parameters in krb5.conf. The default_tkt_enctypes parameter specifies the encryption types the client is willing to use when the client requests a ticket-granting ticket (TGT) from the KDC. The TGT is used by the client to acquire other server tickets in a more efficient manner. The effect of setting default_tkt_enctypes is to give the client some control over the encryption types used to protect the communication between the client and KDC when the client requests a server ticket using the TGT (this is called a TGS request). Note, that the encryption types specified in default_tkt_enctypes must match at least one of the principal key encryption types in the principal database stored on the KDC. Otherwise, the TGT request will fail. In most situations, it is best not to set default_tkt_enctypes because this parameter can be a source of interoperability problems. By default, the client code requests that all supported encryption types and the KDC choose the encryption types based on the keys the KDC finds in the principal database.
The `default_tgs_enctypes` parameter restricts the encryption types the client requests in its TGS requests, which are used to acquire server tickets. This parameter also restricts the encryption types the KDC uses when creating the session key that the client and server share. For example, if a client wants to only use 3DES encryption when doing secure NFS, you should set `default_tgs_enctypes = des3-cbc-sha1`. Make sure that the client and server principals have a `des-3-cbc-sha1` key in the principal database. As with `default_tkt_enctype`, it is probably best in most cases not to set this because it can cause interoperability problems if the credentials are not setup properly both on the KDC and the server.

On the server, you can control the encryption types accepted by the server with the `permitted_enctypes` in `kdc.conf`. In addition, you can specify the encryption types used when creating `keytab` entries. Again, it is generally best not to use either of these methods to control encryption types and instead let the KDC determine the encryption types to use because the KDC does not communicate with the server application to determine which key or encryption type to use.

### Using the `gsscred` Table

The `gsscred` table is used by an NFS server when the server is trying to identify a Kerberos user, if the default mappings are not sufficient. The NFS service uses UNIX IDs to identify users. These IDs are not part of a user principal or a credential. The `gsscred` table provides additional mapping from GSS credentials to UNIX UIDs (from the password file). The table must be created and administered after the KDC database is populated. See “Mapping GSS Credentials to UNIX Credentials” on page 381 for more information.

When a client request comes in, the NFS service tries to map the credential name to a UNIX ID. If the mapping fails, the `gsscred` table is checked.

### Notable Differences Between Solaris Kerberos and MIT Kerberos

The Solaris 10 version of the Kerberos service is based on MIT Kerberos version 1.2.1. The following lists the enhancements included in the Solaris 10 release that are not included in the MIT 1.2.1 version:

- Kerberos support of Solaris remote applications
- Incremental propagation for the KDC database
- Client configuration script
- Localized error messages
- BSM audit record support
- Thread safe use of Kerberos using GSS-API
- Use of the Encryption Framework for cryptography
Notable Differences Between Solaris Kerberos and MIT Kerberos

This version also includes some post MIT 1.2.1 bug fixes. In particular, 1.2.5 btree bug fixes and 1.3 TCP support have been added.
This section provides information on the configuration, management, and use of the Solaris auditing subsystem.
Solaris Auditing (Overview)

Solaris auditing keeps a record of how the system is being used. The audit service includes tools to assist with the analysis of the auditing data.

This chapter introduces how auditing works in the Solaris Operating System. The following is a list of the information in this chapter.

- “What Is Auditing?” on page 551
- “How Does Auditing Work?” on page 553
- “How Is Auditing Related to Security?” on page 554
- “Audit Terminology and Concepts” on page 554
- “Auditing on a System With Zones” on page 560

For planning suggestions, see Chapter 29, “Planning for Solaris Auditing.” For procedures to configure auditing at your site, see Chapter 30, “Managing Solaris Auditing (Tasks).” For reference information, see Chapter 31, “Solaris Auditing (Reference).”

What Is Auditing?

Auditing is the collecting of data about the use of system resources. The audit data provides a record of security-related system events. This data can then be used to assign responsibility for actions that take place on a host. Successful auditing starts with two security features: identification and authentication. At each login, after a user supplies a user name and password, a unique audit session ID is generated and associated with the user’s process. The audit session ID is inherited by every process that is started during the login session. Even if a user changes identity within a single session, all user actions are tracked with the same audit session ID. For more details about changing identity, see the su(1M) man page.

The audit service makes the following possible:

- Monitoring security-relevant events that take place on the host
- Recording the events in a network-wide audit trail
Detecting misuse or unauthorized activity
- Reviewing patterns of access and the access histories of individuals and objects
- Discovering attempts to bypass the protection mechanisms
- Discovering extended use of privilege that occurs when a user changes identity

During system configuration, you preselect which classes of audit records to monitor. You can also fine-tune the degree of auditing that is done for individual users. The following figure shows details of the flow of Solaris auditing.

After audit data is collected in the kernel, plugins distribute the data to the appropriate locations. Then, postselection tools enable you to reduce and examine interesting parts of the audit trail. For example, you can choose to review audit records for individual users or specific groups. You can examine all records for a certain type of event on a specific day. Or, you can select records that were generated at a certain time of day.
Systems that install non-global zones can audit all zones identically from the global zone. These systems can also be configured to collect different records in the non-global zones. For more information, see “Auditing and Solaris Zones” on page 637.

How Does Auditing Work?

Auditing generates audit records when specified events occur. Most commonly, events that generate audit records include the following:

- System startup and system shutdown
- Login and logout
- Process creation or process destruction, or thread creation or thread destruction
- Opening, closing, creating, destroying, or renaming of objects
- Use of privilege capabilities or role-based access control (RBAC)
- Identification actions and authentication actions
- Permission changes by a process or user
- Administrative actions, such as installing a package
- Site-specific applications

Audit records are generated from three sources:

- By an application
- As a result of an asynchronous audit event
- As a result of a process system call

Once the relevant event information has been captured, the information is formatted into an audit record. The record is then written to audit files. Complete audit records are stored in binary format. With the Solaris 10 release, audit records can also be logged by the syslog utility.

Audit files in binary format can be stored in a local file system. The files can also be stored on NFS-mounted file servers. The location can include multiple partitions on the same system, partitions on different systems, or partitions on systems on different but linked networks. The collection of audit files that are linked together is considered an audit trail. Audit records accumulate in audit files chronologically. Contained in each audit record is information that identifies the event, what caused the event, the time of the event, and other relevant information.

Audit records can also be monitored by using the syslog utility. These audit logs can be stored locally. Or, the logs can be sent to a remote system over the UDP protocol. For more information, see “Audit Logs” on page 558.
How Is Auditing Related to Security?

Solaris auditing helps to detect potential security breaches by revealing suspicious or abnormal patterns of system usage. Solaris auditing also provides a means to trace suspect actions back to a particular user, thus serving as a deterrent. Users who know that their activities are being audited are less likely to attempt malicious activities.

To protect a computer system, especially a system on a network, requires mechanisms that control activities before system processes or user processes begin. Security requires tools that monitor activities as the activities occur. Security also requires reports of activities after the activities have happened. Initial configuration of Solaris auditing requires that parameters be set before users log in or system processes begin. Most auditing activities involve monitoring current events and reporting those events that meet the specified parameters. How Solaris auditing monitors and reports these events is discussed in detail in Chapter 29, "Planning for Solaris Auditing," and Chapter 30, "Managing Solaris Auditing (Tasks)."

Auditing cannot prevent hackers from unauthorized entry. However, the audit service can report, for example, that a specific user performed specific actions at a specific time and date. The audit report can identify the user by entry path and username. Such information can be reported immediately to your terminal and to a file for later analysis. Thus, the audit service provides data that helps you determine the following:

- How system security was compromised
- What loopholes need to be closed to ensure the desired level of security

Audit Terminology and Concepts

The following terms are used to describe the audit service. Some definitions include pointers to more complete descriptions.

TABLE 28–1  Solaris Auditing Terms

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audit class</td>
<td>A grouping of audit events. Audit classes provide a way to select a group of events to be audited. For more information, see &quot;Audit Classes and Preselection&quot; on page 556.</td>
</tr>
<tr>
<td>Audit directory</td>
<td>A repository of audit files in binary format. For a description of the types of audit directories, see &quot;Audit Logs&quot; on page 558.</td>
</tr>
<tr>
<td>Audit event</td>
<td>A security-related system action that is audited. For ease of selection, events are grouped into audit classes. For a description of the system actions that can be audited, see &quot;Audit Events&quot; on page 555.</td>
</tr>
</tbody>
</table>
Table 28–1  Solaris Auditing Terms  (Continued)

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audit policy</td>
<td>A set of auditing options that you can enable or disable at your site. These options include whether to record certain kinds of audit data. The options also include whether to suspend auditable actions when the audit trail is full. For more information, see “Determining Audit Policy” on page 568.</td>
</tr>
<tr>
<td>Audit record</td>
<td>Audit data that is stored in audit files. An audit record describes a single audit event. Each audit record is composed of audit tokens. For more information about audit records, see “Audit Records and Audit Tokens” on page 557.</td>
</tr>
<tr>
<td>Audit token</td>
<td>A field of an audit record or event. Each audit token describes an attribute of an audit event, such as a user, a program, or other object. For descriptions of all the audit tokens, see “Audit Token Formats” on page 644.</td>
</tr>
<tr>
<td>Audit trail</td>
<td>A collection of one or more audit files that store the audit data from all systems that run the audit service. For more information, see “Audit Trail” on page 642.</td>
</tr>
<tr>
<td>Preselection</td>
<td>Preselection is the choice of which audit classes to monitor before you enable the audit service. The audit events of preselected audit classes appear in the audit trail. Audit classes that are not preselected are not audited, so their events do not appear in the audit trail. A postselection tool, the audit reduce command, selects records from the audit trail. For more information, see “Audit Classes and Preselection” on page 556.</td>
</tr>
<tr>
<td>Public objects</td>
<td>A public object is a file that is owned by the root user and readable by the world. For example, files in the /etc directory and the /usr/bin directory are public objects. Public objects are not audited for read-only events. For example, even if the file_read (fr) audit class is preselected, the reading of public objects is not audited. You can override the default by changing the public audit policy option.</td>
</tr>
<tr>
<td>Audit plugins</td>
<td>Modules that transfer the audit records in the kernel queue to a specified location. The audit_binfile.so plugin creates binary audit files (the audit trail). The audit_syslog.so plugin filters selected audit records to the syslog logs. For more information, see “Audit Plugin Modules” on page 558.</td>
</tr>
</tbody>
</table>

Audit Events

Security-relevant system actions can be audited. These auditable actions are defined as audit events. Audit events are listed in the /etc/security/audit_event file. Each audit event is defined in the file by an event number, a symbolic name, a short description, and the set of audit classes to which the event belongs. For more information on the audit_event file, see the audit_event(4) man page.

For example, the following entry defines the audit event for the exec() system call:

7: AUE_EXEC: exec(2): ps, ex

When you preselect for auditing either the audit class ps or the audit class ex, then exec() system calls are recorded in the audit trail.
Solaris auditing handles *attributable* and *nonattributable* events. Audit policy divides events into *synchronous* and *asynchronous* events. as follows:

- **Attributable events** – Events that can be attributed to a user. The `exec()` system call can be attributed to a user, so the call is considered an attributable event. All attributable events are synchronous events.

- **Nonattributable events** – Events that occur at the kernel-interrupt level or before a user is authenticated. The `na` audit class handles audit events that are nonattributable. For example, booting the system is a nonattributable event. Most nonattributable events are asynchronous events. However, nonattributable events that have associated processes, such as failed login, are synchronous events.

- **Synchronous events** – Events that are associated with a process in the system. Synchronous events are the majority of system events.

- **Asynchronous events** – Events that are not associated with any process, so no process is available to be blocked and later woken up. Initial system boot and PROM enter and exit events are examples of asynchronous events.

When the class to which an audit event belongs is preselected for auditing, the event is recorded in the audit trail. For example, when you preselect the `ps` and `na` audit classes for auditing, the `exec()` system calls and system boot actions, among other events, are recorded in the audit trail.

In addition to the audit events that are defined by the Solaris audit service, third-party applications can generate audit events. Audit event numbers from 32768 to 65535 are available for third-party applications.

**Audit Classes and Preselection**

Each audit event belongs to an *audit class* or classes. Audit classes are convenient containers for large numbers of audit events. When you *preselect* a class to be audited, you specify that all the events in that class should be recorded in the audit trail. You can preselect for events on a system and for events initiated by a particular user. After the audit service is running, you can dynamically add or remove audit classes from the preselected classes.

- **System-wide preselection** – Specify system-wide defaults for auditing in the `flags`, `naflags`, and plugin lines in the `audit_control` file. The `audit_control` file is described in “audit_control File” on page 632. See also the `audit_control(4)` man page.

- **User-specific preselection** – Specify additions to the system-wide auditing defaults for individual users in the `audit_user` database.

  The audit preselection mask determines which classes of events are audited for a user. The user’s audit preselection mask is a combination of the system-wide defaults and the audit classes that are specified for the user. For a more detailed discussion, see “Process Audit Characteristics” on page 641.
The audit_user database can be administered locally or by a naming service. The Solaris Management Console provides the graphical user interface (GUI) to administer the database. For details, see the audit_user(4) man page.

- **Dynamic preselection** – Specify audit classes as arguments to the auditconfig command to add or remove those audit classes from a process or session. For more information, see the auditconfig(1M) man page.

A postselection command, auditreduce, enables you to select records from the preselected audit records. For more information, see “Examining the Audit Trail” on page 560 and the auditreduce(1M) man page.

Audit classes are defined in the /etc/security/audit_class file. Each entry contains the audit mask for the class, the name for the class, and a descriptive name for the class. For example, the ps and na class definitions appear in the audit_class file as follows:

```
0x00100000:ps:process start/stop
0x00000400:na:non-attribute
```

There are 32 possible audit classes. The classes include the two global classes: all and no. The audit classes are described in the audit_class(4) man page.

The mapping of audit events to classes is configurable. You can remove events from a class, add events to a class, and create a new class to contain selected events. For the procedure, see “How to Change an Audit Event’s Class Membership” on page 584.

### Audit Records and Audit Tokens

Each audit record records the occurrence of a single audited event. The record includes information such as who did the action, which files were affected, what action was attempted, and where and when the action occurred. The following example shows a login audit record:

```
header,81,2,login - local,,2003-10-13 11:23:31.050 -07:00
subject,root,root,other,root,other,378,378,0 0 example_system
text,successful login
return,success,0
```

The type of information that is saved for each audit event is defined by a set of audit tokens. Each time an audit record is created for an event, the record contains some or all of the tokens that are defined for the event. The nature of the event determines which tokens are recorded. In the preceding example, each line begins with the name of the audit token. The content of the audit token follows the name. Together, the four audit tokens comprise the login audit record.

For a detailed description of the structure of each audit token with an example of praudit output, see “Audit Token Formats” on page 644. For a description of the binary stream of audit tokens, see the audit.log(4) man page.
Audit Plugin Modules

You can specify audit plugin modules to handle the records that your preselection has placed in the audit queue. The plugins are entries in the audit_control file.

- **audit_binfile.so** plugin – Handles delivery of the audit queue to the binary audit files. In the audit_control file, if no plugin is specified and the dir entry has a value, then the audit daemon uses this plugin.

- **audit_remote.so** plugin – Handles secure delivery of binary audit records from the audit queue to a configured remote server. The audit_remote.so plugin uses the libgss() library to authenticate the server. The transmission is protected for privacy and integrity.

- **audit_syslog.so** plugin – Handles delivery of selected records from the audit queue to the syslog logs.

For the syntax of the audit_control file, see the audit_control(4) man page. For examples, see the tasks in “Configuring Audit Files (Task Map)” on page 576.

For information about the plugins, see the audit_binfile(5), audit_syslog(5), audit_remote(5), and audit_control(4) man pages.

Audit Logs

Audit records are collected in audit logs. Solaris auditing provides two output modes for audit logs. Logs that are called audit files store audit records in binary format. The set of audit files from a system or site provide a complete audit record. The complete audit record is called the audit trail.

The syslog utility collects and stores text version summaries of the audit record. A syslog record is not complete. The following example shows a syslog entry for a login audit record:

```
Oct 13 11:24:11 example_system auditd: [ID 6472 audit.notice] login - login ok session 378 by root as root:other
```

A site can store audit records in both formats. You can configure the systems at your site to use binary mode, to use syslog mode, or to use both modes. The following table compares binary audit records with syslog audit records.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Binary Records</th>
<th>syslog Records</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>Writes to the file system</td>
<td>Uses UDP for remote logging</td>
</tr>
<tr>
<td>Data type</td>
<td>Binary</td>
<td>Text</td>
</tr>
<tr>
<td>Record length</td>
<td>No limit</td>
<td>Up to 1024 characters per audit record</td>
</tr>
</tbody>
</table>
TABLE 28–2  Comparison of Binary Audit Records With **syslog** Audit Records  

<table>
<thead>
<tr>
<th>Feature</th>
<th>Binary Records</th>
<th><strong>syslog</strong> Records</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Location</strong></td>
<td>Stored on local disk, and in directories that are mounted by using NFS</td>
<td>Stored in a location that is specified in the <code>syslog.conf</code> file</td>
</tr>
<tr>
<td><strong>How to configure</strong></td>
<td>Edit <code>audit_control</code> file, and protect and NFS-mount audit directories</td>
<td>Edit <code>audit_control</code> file, and edit <code>syslog.conf</code> file</td>
</tr>
<tr>
<td><strong>How to read</strong></td>
<td>Typically, in batch mode, browser output in XML</td>
<td>In real time, or searched by scripts that you have created for <code>syslog</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Plain text output</td>
</tr>
<tr>
<td><strong>Completeness</strong></td>
<td>Guaranteed to be complete, and to appear in the correct order</td>
<td>Are not guaranteed to be complete</td>
</tr>
<tr>
<td><strong>Timestamp</strong></td>
<td>Greenwich Mean Time (GMT)</td>
<td>Time on the system that is being audited</td>
</tr>
</tbody>
</table>

Binary records provide the greatest security and coverage. Binary output meets the requirements of security certifications, such as the Common Criteria Controlled Access Protection Profile (CAPP). The records are written to a file system that you protect from snooping. On a single system, all binary records are collected and are displayed in order. The GMT timestamp on binary logs enables accurate comparison when systems on one audit trail are distributed across time zones. The `praudit -x` command enables you to view the records in a browser in XML. You can also use scripts to parse the XML output.

In contrast, the **syslog** records provide greater convenience and flexibility. For example, you can collect the **syslog** data from a variety of sources. Also, when you monitor `audit notice` events in the `syslog.conf` file, the **syslog** utility logs an audit record summary with the current timestamp. You can use the same management and analysis tools that you have developed for **syslog** messages from a variety of sources, including workstations, servers, firewalls, and routers. The records can be viewed in real time, and can be stored on a remote system.

By using `syslog.conf` to store audit records remotely, you protect log data from alteration or deletion by an attacker. On the other hand, when audit records are stored remotely, the records are susceptible to network attacks such as denial of service and spoofed source addresses. Also, UDP can drop packets or can deliver packets out of order. The limit on **syslog** entries is 1024 characters, so some audit records could be truncated in the log. On a single system, not all audit records are collected. The records might not display in order. Because each audit record is stamped with the local system’s date and time, you cannot rely on the timestamp to construct an audit trail for several systems.

For more information on audit logs, refer to the following:

- `audit_syslog(5)` man page
- `audit.log(4)` man page
- “How to Configure **syslog** Audit Logs” on page 579
Storing the Audit Trail

An audit directory holds audit files in binary format. A typical installation uses many audit directories. The contents of all audit directories comprise the audit trail. Audit records are stored in audit directories in the following order:

- **Primary audit directory** – A directory where the audit files for a system are placed under normal conditions
- **Secondary audit directory** – A directory where the audit files for a system are placed if the primary audit directory is full or not available
- **Directory of last resort** – A local audit directory that is used if the primary audit directory and all secondary audit directories are not available

The directories are specified in the audit_control file. A directory is not used until a directory that is earlier in the list is full. For an annotated audit_control file with a list of directory entries, see Example 30–3.

Placing the audit files in the default audit root directory assists the audit reviewer when reviewing the audit trail. The auditreduce command uses the audit root directory to find all files in the audit trail. The default audit root directory is /etc/security/audit. This directory is symbolically linked to /var/audit. Audit files in directories that are named /var/audit/hostname/files are easily found by the auditreduce command. For more information, see “auditreduce Command” on page 628.

Examining the Audit Trail

The audit service provides commands to combine and reduce files from the audit trail. The auditreduce command can merge audit files from the audit trail. The command can also filter files to locate particular events. The praudit command reads the binary files. Options to the praudit command provide output that is suitable for scripting and for browser display.

Auditing on a System With Zones

A zone is a virtualized operating system environment that is created within a single instance of the Solaris OS. The audit service audits the entire system, including activities in zones. A system that has installed non-global zones can run a single audit service to audit all zones identically. Or, it can configure one audit service per zone, including the global zone.
Sites that satisfy the following conditions can run a single audit service:

- The site requires a single-image audit trail.
- The non-global zones are used as application containers. The zones are part of one administrative domain. That is, no non-global zone has customized naming service files.
  
  If all the zones on a system are within one administrative domain, the zone implies audit policy can be used to distinguish audit events that execute in different zones.
- Administrators want low audit overhead. The global zone administrator audits all zones identically. Also, the global zone's audit daemon serves all zones on the system.

Sites that satisfy the following conditions can run one audit service per zone:

- The site does not require a single-image audit trail.
- The non-global zones have customized naming service files. These separate administrative domains typically function as servers.
- Individual zone administrators want to control auditing in the zones that they administer. In per-zone auditing, zone administrators can decide to enable or to disable auditing for the zone that they administer.

The advantages of per-zone auditing are a customized audit trail for each zone, and the ability to disable auditing on a zone by zone basis. These advantages can be offset by the administrative overhead. The zone administrator customizes every audit configuration file. Each zone runs its own audit daemon, and has its own audit queue and audit logs. The zone's audit log files must be managed.
This chapter describes how to set up the audit service for your Solaris installation. In particular, the chapter covers issues that you need to consider before you enable the audit service. The following is a list of the planning information in this chapter:

- “Planning Solaris Auditing (Task Map)” on page 563
- “Determining Audit Policy” on page 568
- “Controlling Auditing Costs” on page 571
- “Auditing Efficiently” on page 573

For an overview of auditing, see Chapter 28, “Solaris Auditing (Overview).” For procedures to configure auditing at your site, see Chapter 30, “Managing Solaris Auditing (Tasks).” For reference information, see Chapter 31, “Solaris Auditing (Reference).”

### Planning Solaris Auditing (Task Map)

The following task map points to the major tasks that are required for planning disk space and what events to record.

<table>
<thead>
<tr>
<th>Task</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determine auditing strategy for non-global zones</td>
<td>“How to Plan Auditing in Zones” on page 564</td>
</tr>
<tr>
<td>Plan storage space for the audit trail</td>
<td>“How to Plan Storage for Audit Records” on page 565</td>
</tr>
<tr>
<td>Determine who and what to audit</td>
<td>“How to Plan Who and What to Audit” on page 566</td>
</tr>
</tbody>
</table>
Planning Solaris Auditing (Tasks)

You want to be selective about what kinds of activities are audited. At the same time, you want to collect useful audit information. Audit files can quickly grow to fill the available space, so you should allocate enough disk space. You also need to carefully plan who to audit and what to audit.

▼ How to Plan Auditing in Zones

If your system has implemented zones, you have two audit configuration possibilities:

■ You can configure a single audit service in the global zone for all zones.
■ You can configure one audit service per zone.

For a discussion of the trade-offs, see “Auditing on a System With Zones” on page 560.

● Choose one of the following methods.

■ OPTION 1 - Configure a single audit service for all zones.

Auditing all zones identically can create a single-image audit trail. A single-image audit trail occurs when all zones on a system are part of one administrative domain. The audit records can then be easily compared, because the records in every zone are preselected with identical settings.

This configuration treats all zones as part of one system. The global zone runs the only audit daemon on a system, and collects audit logs for every zone. You customize audit configuration files only in the global zone, then copy the audit configuration files to every non-global zone.

a. Copy the audit_control file from the global zone to every non-global zone.

b. Use the same audit_user database for every zone.

The audit_user database might be a local file, or you might get it from a shared naming service.

c. Enable the audit records to be selected by zone.

To put the zone name as part of the audit record, set the zonename policy in the global zone. The audit_reduce command can then select audit events by zone from the audit trail. For an example, see the auditreduce(1M) man page.

To plan a single-image audit trail, refer to “How to Plan Who and What to Audit” on page 566. Start with the first step. The global zone administrator must also set aside storage, as described in “How to Plan Storage for Audit Records” on page 565.
- **OPTION 2 - Configure one audit service per zone.**
  Choose to configure per-zone auditing if different zones have different naming service files, or if zone administrators want to control auditing in their zones.

  - When you configure per-zone auditing, you must configure the global zone for auditing. You set the per zone audit policy in the global zone. To set audit policy, see “How to Configure Per-Zone Auditing” on page 600.

  **Note** – If naming service files are customized in non-global zones, and per zone policy is not set, then careful use of the audit tools is required to select usable records. A user ID in one zone can refer to a different user from the same ID in a different zone.

  - To generate records that can be traced to their originating zone, set the zonename audit policy in the global zone. In the global zone, run the audit reduce command with the zonename option. Then, in the zonename zone, run the praudit command on the audit reduce output.
  - Each zone administrator configures the audit files for the zone.
    - A non-global zone administrator can set all policy options except per zone and ahlt.
  - Each zone administrator can enable or disable auditing in the zone.

  If you customize audit configuration files in every zone, use "How to Plan Who and What to Audit" on page 566 to plan for every zone. You can skip the first step. Each zone administrator must also set aside storage for every zone, as described in "How to Plan Storage for Audit Records" on page 565.

### How to Plan Storage for Audit Records

The audit trail requires dedicated file space. The dedicated file space for audit files must be available and secure. Each system should have several audit directories that are configured for audit files. You should decide how to configure the audit directories as one of the first tasks before you enable auditing on any systems. The following procedure covers the issues to be resolved when you plan for audit trail storage.

**Before You Begin**
If you are implementing non-global zones, complete “How to Plan Auditing in Zones” on page 564 before using this procedure.

1. **Determine how much auditing your site needs.**
   Balance your site’s security needs against the availability of disk space for the audit trail.

   For guidance on how to reduce space requirements while still maintaining site security, as well as how to design audit storage, see "Controlling Auditing Costs" on page 571 and "Auditing Efficiently" on page 573.
2 **Determine which systems are to be audited.**
On those systems, allocate space for at least one local audit directory. To specify the audit directories, see Example 30–3.

3 **Determine which systems are to store audit files.**
Decide which servers are to hold the primary and secondary audit directories. For examples of configuring disks for audit directories, see “How to Create Partitions for Audit Files” on page 587.

4 **Name the audit directories.**
Create a list of all the audit directories that you plan to use. For naming guidelines, see “Storing the Audit Trail” on page 560 and “auditreduce Command” on page 628.

5 **Determine which systems are to use which audit directories.**
Create a map that shows which system should use which audit directory. The map helps you to balance the auditing activity. For an illustration, see Figure 31–1 and Figure 31–2.

**How to Plan Who and What to Audit**

**Before You Begin** If you are implementing non-global zones, complete “How to Plan Auditing in Zones” on page 564 before using this procedure.

1 **Determine if you want a single-system image audit trail.**
Systems within a single administrative domain can create a single-system image audit trail. If your systems use different naming services, start with the next step. You should complete the rest of the planning steps for every system.

A single-system image audit trail treats the systems that are being audited as one machine. To create a single-system image audit trail for a site, every system in the installation should be configured as follows:

- Use the same naming service.
  
  To interpret the audit records, two commands are used, auditreduce and praudit. For correct interpretation of the audit records, the passwd, hosts, and audit_user files must be consistent.

- Use the same audit service settings for all systems. For information about displaying and modifying the service settings, see the auditconfig(1M) man page.

- Use the same audit_warn, audit_event, and audit_class files for all systems.

- Use the same audit_user database. The database can be in a naming service such as NIS or LDAP.

- Have identical flags, naflags, and plugin entries in the audit_control file.
2 Determine the audit policy.
   Use the auditconfig -ls policy command to see a short description of available policy options. By default, only the cnt policy is turned on. For a fuller discussion, see Step 8.

   For the effects of the policy options, see “Determining Audit Policy” on page 568. To set audit policy, see “How to Configure Audit Policy” on page 590.

3 Determine if you want to modify event-to-class mappings.
   In many situations, the default mapping is sufficient. However, if you add new classes, change class definitions, or determine that a record of a specific system call is not useful, you might also need to move an event to a different class.

   For an example, see “How to Change an Audit Event’s Class Membership” on page 584.

4 Determine which audit classes to preselect.
   The best time to add audit classes or to change the default classes is before you start the audit service.

   The audit class values of the flags, nflags, and plugin entries in the audit_control file apply to all users and processes. The preselected classes determine whether an audit class is audited for success, for failure, or for both.

   To preselect audit classes, see “How to Modify the audit_control File” on page 577.

5 Determine user exceptions to the system-wide preselected audit classes.
   If you decide that some users should be audited differently from the system-wide preselected audit classes, modify the individual users’ entries in the audit_user database.

   For an example, see “How to Change a User’s Audit Characteristics” on page 581.

6 Determine the minimum free disk space.
   When disk space on an audit file system drops below the minfree percentage, the auditd daemon switches to the next available audit directory. The daemon then sends a warning that the soft limit has been exceeded.

   To set the minimum free disk space, see Example 30–4.

7 Decide how to manage the audit_warn email alias.
   The audit_warn script is run whenever the audit system needs to notify you of a situation that requires administrative attention. By default, the audit_warn script sends email to an audit_warn alias and sends a message to the console.

   To set up the alias, see “How to Configure the audit_warn Email Alias” on page 589.
8 Decide what action to take when all the audit directories are full.

By default, when the audit trail overflows, the system continues to work. The system counts the audit records that are dropped, but does not record the events. For greater security, you can disable the cnt policy, and enable the ahlt policy. The ahlt policy stops the system when an asynchronous event cannot be placed in the audit queue.

For a discussion of these policy options, see “Audit Policies for Asynchronous and Synchronous Events” on page 570. To configure these policy options, see Example 30–13.

9 Decide whether to collect audit records in binary format, in syslog format, or in both formats.

For overview information, see “Audit Logs” on page 558.

For an example, see “How to Configure syslog Audit Logs” on page 579.

Determining Audit Policy

Audit policy determines the characteristics of the audit records for the local system. You use the auditconfig command to set these policies. For more information, see the auditconfig(1M) man page.

Most audit policy options are disabled by default to minimize storage requirements and system processing demands. These options are stored as properties of the audit service and determine the policy options that are in effect at system boot or when the service is restarted. For more information about the audit service, see “Solaris Audit Service” on page 625.

Use the following table to determine if the needs of your site justify the additional overhead that results from enabling one or more audit policy options.

<table>
<thead>
<tr>
<th>Policy Name</th>
<th>Description</th>
<th>Why Change the Policy Option?</th>
</tr>
</thead>
<tbody>
<tr>
<td>ahlt</td>
<td>This policy applies to asynchronous events only. When disabled, this policy allows the event to complete without an audit record being generated. When enabled, this policy stops the system when the audit file systems are full. Administrative intervention is required to clean up the audit queue, make space available for audit records, and reboot. This policy can only be enabled in the global zone. The policy affects all zones.</td>
<td>The disabled option makes sense when system availability is more important than security. The enabled option makes sense in an environment where security is paramount.</td>
</tr>
</tbody>
</table>
### TABLE 29–1  Effects of Audit Policy Options  (Continued)

<table>
<thead>
<tr>
<th>Policy Name</th>
<th>Description</th>
<th>Why Change the Policy Option?</th>
</tr>
</thead>
<tbody>
<tr>
<td>arge</td>
<td>When disabled, this policy omits environment variables of an executed program from the exec audit record. When enabled, this policy adds the environment variables of an executed program to the exec audit record. The resulting audit records contain much more detail than when this policy is disabled.</td>
<td>The disabled option collects much less information than the enabled option. The enabled option makes sense when you are auditing a few users. The option is also useful when you have suspicions about the environment variables that are being used in exec programs.</td>
</tr>
<tr>
<td>argv</td>
<td>When disabled, this policy omits the arguments of an executed program from the exec audit record. When enabled, this policy adds the arguments of an executed program to the exec audit record. The resulting audit records contain much more detail than when this policy is disabled.</td>
<td>The disabled option collects much less information than the enabled option. The enabled option makes sense when you are auditing a few users. The option is also useful when you have reason to believe that unusual exec programs are being run.</td>
</tr>
<tr>
<td>cnt</td>
<td>When disabled, this policy blocks a user or application from running. The blocking happens when audit records cannot be added to the audit trail because no disk space is available. When enabled, this policy allows the event to complete without an audit record being generated. The policy maintains a count of audit records that are dropped.</td>
<td>The disabled option makes sense in an environment where security is paramount. The enabled option makes sense when system availability is more important than security.</td>
</tr>
<tr>
<td>group</td>
<td>When disabled, this policy does not add a groups list to audit records. When enabled, this policy adds a groups list to every audit record as a special token.</td>
<td>The disabled option usually satisfies requirements for site security. The enabled option makes sense when you need to audit which groups are generating audit events.</td>
</tr>
<tr>
<td>path</td>
<td>When disabled, this policy records in an audit record at most one path that is used during a system call. When enabled, this policy records every path that is used in conjunction with an audit event to every audit record.</td>
<td>The disabled option places at most one path in an audit record. The enabled option enters each file name or path that is used during a system call in the audit record as a path token.</td>
</tr>
<tr>
<td>perzone</td>
<td>When disabled, this policy maintains a single audit configuration for a system. One audit daemon runs in the global zone. Audit events in non-global zones can be located in the audit record by preselecting the zonename audit token. When enabled, this policy maintains separate audit configuration, audit queue, and audit logs for each zone. A separate version of the audit daemon runs in each zone. This policy can be enabled in the global zone only.</td>
<td>The disabled option is useful when you have no special reason to maintain a separate audit log, queue, and daemon for each zone. The enabled option is useful when you cannot monitor your system effectively by simply preselecting the zonename audit token.</td>
</tr>
</tbody>
</table>
### TABLE 29–1  Effects of Audit Policy Options  
(Continued)

<table>
<thead>
<tr>
<th>Policy Name</th>
<th>Description</th>
<th>Why Change the Policy Option?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>public</strong></td>
<td>When disabled, this policy does not add read-only events of public objects to the audit trail when the reading of files is preselected. Audit classes that contain read-only events include fr, fa, and ct. When enabled, this policy records every read-only audit event of public objects if an appropriate audit class is preselected.</td>
<td>The disabled option usually satisfies requirements for site security. The enabled option is rarely useful.</td>
</tr>
<tr>
<td><strong>seq</strong></td>
<td>When disabled, this policy does not add a sequence number to every audit record. When enabled, this policy adds a sequence number to every audit record. The sequence token holds the sequence number.</td>
<td>The disabled option is sufficient when auditing is running smoothly. The enabled option makes sense when the cnt policy is enabled. The seq policy enables you to determine when data was discarded.</td>
</tr>
<tr>
<td><strong>trail</strong></td>
<td>When disabled, this policy does not add a trailer token to audit records. When enabled, this policy adds a trailer token to every audit record.</td>
<td>The disabled option creates a smaller audit record. The enabled option clearly marks the end of each audit record with a trailer token. The trailer token is often used in conjunction with the sequence token. The trailer token provides easier and more accurate resynchronization of audit records.</td>
</tr>
<tr>
<td><strong>zonename</strong></td>
<td>When disabled, this policy does not include a zonename token in audit records. When enabled, this policy includes a zonename token in every audit record from a non-global zone.</td>
<td>The disabled option is useful when you do not need to compare audit behavior across zones. The enabled option is useful when you want to isolate and compare audit behavior across zones.</td>
</tr>
</tbody>
</table>

### Audit Policies for Asynchronous and Synchronous Events

Together, the `ahlt` policy and the `cnt` policy govern what happens when the audit queue is full and cannot accept more events. The policies are independent and related. The combinations of the policies have the following effects:

- `-ahlt +cnt` is the default policy that is shipped. This default lets an audited event be processed even if the event cannot be logged. The `-ahlt` policy states that if an audit record of an asynchronous event cannot be placed in the kernel audit queue, the system will count the events and continue processing. In the global zone, the `as_dropped` counter records the count.
  - The `+cnt` policy states that if a synchronous event arrives and the event cannot be placed in the kernel audit queue, the system will count the event and continue processing. The zone’s `as_dropped` counter records the count.
The -ahlt +cnt configuration is generally used at sites where processing must continue, even if continued processing could result in a loss of audit records. The auditstat drop field shows the number of audit records that are dropped in a zone.

- The +ahlt -cnt policy states that processing halts when an event cannot be added to the kernel audit queue.

The +ahlt policy states that if an audit record of an asynchronous event cannot be placed in the kernel audit queue, all processing is stopped. The system will panic. The asynchronous event will not be in the audit queue and must be recovered from pointers on the call stack.

The -cnt policy states that if a synchronous event cannot be placed in the kernel audit queue, the thread that is attempting to deliver the event will be blocked. The thread is placed in a sleep queue until audit space becomes available. No count is kept. Programs might appear to hang until audit space becomes available.

The -ahlt -cnt configuration is generally used in sites where a record of every audit event takes precedence over system availability. Programs will appear to hang until audit space becomes available. The auditstat wblk field shows the number of times that threads were blocked.

However, if an asynchronous event occurs, the system will panic, leading to an outage. The kernel queue of audit events can be manually recovered from a saved crash dump. The asynchronous event will not be in the audit queue and must be recovered from pointers on the call stack.

- The -ahlt -cnt policy states that if an asynchronous event cannot be placed in the kernel audit queue, the event will be counted and processing will continue. When a synchronous event cannot be placed in the kernel audit queue, the thread that is attempting to deliver the event will be blocked. The thread is placed in a sleep queue until audit space becomes available. No count is kept. Programs might appear to hang until audit space becomes available.

The -ahlt -cnt configuration is generally used in sites where the recording of all synchronous audit events takes precedence over some potential loss of asynchronous audit records. The auditstat wblk field shows the number of times that threads were blocked.

- The +ahlt +cnt policy states that if an asynchronous event cannot be placed in the kernel audit queue, the system will panic. If a synchronous event cannot be placed in the kernel audit queue, the system will count the event and continue processing.

**Controlling Auditing Costs**

Because auditing consumes system resources, you must control the degree of detail that is recorded. When you decide what to audit, consider the following costs of auditing:

- Cost of increased processing time
- Cost of analysis of audit data
- Cost of storage of audit data
Cost of Increased Processing Time of Audit Data

The cost of increased processing time is the least significant of the costs of auditing. The first reason is that auditing generally does not occur during computation-intensive tasks, such as image processing, complex calculations, and so forth. The other reason is that the cost for single-user systems is usually small enough to ignore.

Cost of Analysis of Audit Data

The cost of analysis is roughly proportional to the amount of audit data that is collected. The cost of analysis includes the time that is required to merge and review audit records. Cost also includes the time that is required to archive the records and keep the records in a safe place.

The fewer records that you generate, the less time that is required to analyze the audit trail. Upcoming sections, “Cost of Storage of Audit Data” on page 572 and “Auditing Efficiently” on page 573, describe ways to audit efficiently. Efficient auditing reduces the amount of audit data, while still providing enough coverage to achieve your site’s security goals.

Cost of Storage of Audit Data

Storage cost is the most significant cost of auditing. The amount of audit data depends on the following:

- Number of users
- Number of systems
- Amount of use
- Degree of traceability and accountability that is required

Because these factors vary from site to site, no formula can predetermine the amount of disk space to set aside for audit data storage. Use the following information as a guide:

- Preselect audit classes judiciously to reduce the volume of records that are generated.
  
  Full auditing, that is, with the all class, fills disks quickly. Even a simple task such as compiling a program could generate a large audit file. A program of modest size could generate thousands of audit records in less than a minute.
  
  For example, by omitting the file_read audit class, fr, you can significantly reduce audit volume. By choosing to audit for failed operations only, you can at times reduce audit volume. For example, by auditing for failed file_read operations, -fr, you can generate far fewer records than by auditing for all file_read events.

- Efficient audit file management is also important. After the audit records are created, file management reduces the amount of storage that is required.

- Understand the audit classes
Before you configure auditing, you should understand the types of events that the classes contain. You can change the audit event-class mappings to optimize audit record collection.

- Develop a philosophy of auditing for your site.
  Base your philosophy on sensible measures. Such measures include the amount of traceability that your site requires, and the types of users that you administer.

**Auditing Efficiently**

The following techniques can help you achieve your organization's security goals while auditing more efficiently.

- Randomly audit only a certain percentage of users at any one time.
- Reduce the disk-storage requirements for audit files by combining, reducing, and compressing the files. Develop procedures for archiving the files, for transferring the files to removable media, and for storing the files offline.
- Monitor the audit data in real time for unusual behaviors. You can extend management and analysis tools that you have already developed to handle audit records in syslog files.

You can also set up procedures to monitor the audit trail for certain activities. You can write a script to trigger an automatic increase in the auditing of certain users or certain systems in response to detection of unusual events.

For example, you could write a script that does the following:

1. Monitors the creation of audit files on all the audit file servers.
2. Processes the audit files with the `tail` command.

   The piping of the output from the `tail -0f` command through the `praudit` command can yield a stream of audit records as the records are generated. For more information, see the `tail(1)` man page.
3. Analyzes this stream for unusual message types or other indicators, and delivers the analysis to the auditor.

   Or, the script can be used to trigger automatic responses.
4. Constantly monitors the audit directories for the appearance of new not_ terminated audit files.
5. Terminates outstanding `tail` processes when their files are no longer being written to.
Managing Solaris Auditing (Tasks)

This chapter presents procedures to help you set up and manage a Solaris system that is audited. This chapter also includes instructions for administering the audit trail. The following is a list of the information in this chapter.

- “Solaris Auditing (Task Map)” on page 575
- “Configuring Audit Files (Task Map)” on page 576
- “Configuring and Enabling the Audit Service (Task Map)” on page 586
- “Managing Audit Records (Task Map)” on page 601
- “Troubleshooting Solaris Auditing (Task Map)” on page 612

For an overview of the audit service, see Chapter 28, “Solaris Auditing (Overview).” For planning suggestions, see Chapter 29, “Planning for Solaris Auditing.” For reference information, see Chapter 31, “Solaris Auditing (Reference).”

Solaris Auditing (Task Map)

The following task map points to the major tasks that are required to manage auditing. The tasks are ordered.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Plan for auditing</td>
<td>Contains configuration issues to decide before you configure the audit service.</td>
<td>“Planning Solaris Auditing (Task Map)” on page 563</td>
</tr>
<tr>
<td>2. Configure audit files</td>
<td>Defines which events, classes, and users require auditing.</td>
<td>“Configuring Audit Files (Task Map)” on page 576</td>
</tr>
</tbody>
</table>
## Configuring Audit Files (Task Map)

The following task map points to the procedures for configuring files to customize auditing at your site. Most of the tasks are optional.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
</table>
| Select audit classes, and customize audit_control settings | Involves:  
■ Preselecting system-wide audit classes  
■ Specifying the audit directories for each system  
■ Setting disk space limits on audit file systems | "How to Modify the audit_control File" on page 577 |
| (Optional) Log audit events in two modes | Enables you to monitor audit events in real time, in addition to storing audit records in binary format. | "How to Configure syslog Audit Logs" on page 579 |
| (Optional) Change audit characteristics for users | Sets user-specific exceptions to the system-wide preselected audit classes. | "How to Change a User’s Audit Characteristics" on page 581 |
| (Optional) Add audit classes | Reduces the number of audit records by creating a new audit class to hold events. | "How to Add an Audit Class" on page 583 |
| (Optional) Change event-to-class mappings | Reduces the number of audit records by changing the event-class mapping. | "How to Change an Audit Event’s Class Membership" on page 584 |

## Configuring Audit Files (Tasks)

Before you enable auditing on your network, you can customize the audit configuration files for your site auditing requirements. You can also restart the audit service or reboot the local system to read changed configuration files after the audit service has been enabled. However, the recommended practice is to customize your audit configuration as much as possible before you start the audit service.

If you have implemented zones, you can choose to audit all zones from the global zone. To differentiate between zones in the audit output, you can set the zonename policy option. Alternatively, to audit non-global zones individually, you can set the per zone policy in the

---

[30x580]Task Description ForInstructions
[30x552]3. Configure and enable auditing Configure each host for disk space and other audit service requirements. Then, starts the audit service.

On a host that has installed non-global zones, configure one audit service for the system, or one audit service per zone.

"Configuring and Enabling the Audit Service (Task Map)” on page 586

"Configuring the Audit Service in Zones (Tasks)” on page 598

4. Manage audit records Collects and analyzes the audit data.

"Managing Audit Records (Task Map)” on page 601
global zone and customize the audit configuration files in the non-global zones. For an overview, see "Auditing and Solaris Zones" on page 637. For planning, see "How to Plan Auditing in Zones" on page 564. For procedures, see "Configuring the Audit Service in Zones (Tasks)" on page 598.

▼ How to Modify the audit_control File

The /etc/security/audit_control file configures system-wide auditing. The file determines which events are audited, when audit warnings are issued, and the location of the audit files.

1 Assume the Primary Administrator role, or become superuser.
The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2 (Optional) Save a backup copy of the audit_control file.
   # cp /etc/security/audit_control /etc/security/audit_control.orig

3 Modify the audit_control file for your site.
   Each entry has the following format:
   keyword: value
   
   keyword      Defines the type of line. The types are flags, naflags, and plugin.
   
   For explanations of the keywords, see the following examples.
   
   value        Specifies data that is associated with the line type.

   Note – To specify the locations of audit directories, use the p_dir attribute to the audit_binfile.so plugin. To specify the minimum free space, use the p_minfree attribute.

4 (Optional) Verify the syntax of the file.
   # audit -v /etc/security/audit_control
   syntax ok

Example 30–1 Preselecting Audit Classes for All Users

The flags line in the audit_control file defines which classes of attributable events are audited for all users on the system. The classes are separated by commas. White space is allowed. In this example, the events in the lo and ap classes are audited for all users.
To see which events are assigned to a class, read the audit_event file. You can also use the auditrecord command, as shown in Example 30–24.

### Example 30–2 Preselecting Nonattributable Events

In this example, all events in the na class, and all login events that are not attributable, are audited.

```plaintext
## audit_control file
flags:lo,ap
naflags:lo
plugin:name=...
```

### Example 30–3 Specifying the Location of Binary Audit Data

The p_dir flag to the audit_binfile.so plugin lists which audit file systems to use for binary audit data. In this example, three locations for binary audit data are defined. The directories are listed in order from the primary directory to the directory of last resort. The plugin line does not contain a line break.

```plaintext
## audit_control file
##
flags:lo
naflags:lo,na
plugin:name=audit_binfile.so; p_dir=/var/audit/egret.1/files, /var/audit/egret.2/files,/var/audit
```

To set up file systems to hold audit binary audit data, see “How to Create Partitions for Audit Files” on page 587.

### Example 30–4 Changing the Soft Limit for Warnings

In this example, the minimum free-space level for all audit file systems is set so that a warning is issued when only 10 percent of the file system is available.

The plugin line does not contain a line break.

```plaintext
## audit_control file
#
flags:lo
```
The audit_warn alias receives the warning. To set up the alias, see "How to Configure the audit_warn Email Alias" on page 589.

▼ How to Configure syslog Audit Logs

You can instruct the audit service to copy some or all of the collected audit records in the audit queue to syslog. In the following procedure, you save binary audit data and text audit data. The collected text audit data is a subset of the binary data.

Before You Begin

You must preselect audit classes. Preselected audit classes are specified in the flags line and the naflags line of the audit_control file. You can also preselect classes for individual users in the audit_user file and dynamically add audit classes with the auditconfig command.

1 Assume the Primary Administrator role, or become superuser.

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in System Administration Guide: Basic Administration.

2 (Optional) Save a backup copy of the audit_control file.

```
# cp /etc/security/audit_control /etc/security/audit_control.save
```

3 Add an audit_syslog.so plugin entry.

```
## audit_control file
flags:lo,ss
naflags:lo,na
plugin:name=audit_binfile.so;p_dir=/var/audit; p_minfree=20;
plugin:name=audit_syslog.so;p_flags=lo,-ss
```

A plugin entry has the following format:

```
plugin:name=name; qsize=max-queued-records;p.*=value
```

- **name** – Lists the name of the plugin. The valid values are audit_binfile.so and audit_syslog.so.
- **qsize** – Specifies the maximum number of records to queue for audit data that is being sent to the plugin. This attribute is optional.
- **p.*=** – Specifies plugin-specific attributes. The audit_syslog.so plugin accepts `p_flags`. The audit_binfile.so plugin accepts `p_dir`, `p_minfree` and `p_fsize`.

Before You Begin

1 Assume the Primary Administrator role, or become superuser.

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in System Administration Guide: Basic Administration.

2 (Optional) Save a backup copy of the audit_control file.

```
# cp /etc/security/audit_control /etc/security/audit_control.save
```

3 Add an audit_syslog.so plugin entry.

```
## audit_control file
flags:lo,ss
naflags:lo,na
plugin:name=audit_binfile.so;p_dir=/var/audit; p_minfree=20;
plugin:name=audit_syslog.so;p_flags=lo,-ss
```

A plugin entry has the following format:

```
plugin:name=name; qsize=max-queued-records;p.*=value
```

- **name** – Lists the name of the plugin. The valid values are audit_binfile.so and audit_syslog.so.
- **qsize** – Specifies the maximum number of records to queue for audit data that is being sent to the plugin. This attribute is optional.
- **p.*=** – Specifies plugin-specific attributes. The audit_syslog.so plugin accepts `p_flags`. The audit_binfile.so plugin accepts `p_dir`, `p_minfree` and `p_fsize`.
The audit_remote.so plugin accepts `p_hosts`, `p_retries` and `p_timeout`

For more information about the plugin-specific attributes, see the `OBJECT ATTRIBUTES` section of the `audit_binfile(5)` and `audit_syslog(5)` man pages. For the audit_remote.so plugin, see the `audit_remote(5)` man page.

4 Add an `audit.notice` entry to the `syslog.conf` file.

The entry includes the location of the log file.

```bash
# cat /etc/syslog.conf
...audit.notice /var/adm/auditlog
```

Do not store text logs where the binary audit files are stored. The `auditreduce` command, which reads binary audit files, assumes that all files in an audit partition are binary audit files.

5 Create the log file.

```bash
# touch /var/adm/auditlog
```

6 Refresh the configuration information for the syslog service.

```bash
# svcadm refresh system/system-log
```

7 Regularly archive the syslog log files.

The audit service can generate extensive output. To manage the logs, see the `logadm(1M)` man page.

**Example 30–5** Specifying Audit Classes for syslog Output

In the following example, the `syslog` utility collects a subset of the preselected audit classes.

```bash
## audit_user file
jdoe:pf

## audit_control file
flags:lo,ss
naflags:lo,na
plugin:name=audit_binfile.so; p_dir=/var/audit/host.1/files, /var/audit/host.2/files,/var/audit/localhost/files; p_minfree=10
plugin:name=audit_syslog.so; p_flags=-lo,-na,-ss,+pf
```

The `flags` and `naflags` entries instruct the system to collect all login/logout, nonattributable, and change of system state audit records in binary format. The `audit_syslog.so` plugin entry instructs the `syslog` utility to collect only failed logins, failed nonattributable events, and failed
changes of system state. For the jdoe user, the binary audit record includes all uses of a profile-aware shell. The syslog utility collects successful profile-aware commands. The pf class is created in Example 30–10.

**Example 30–6**  Putting syslog Audit Records on a Remote System

You can change the audit.notice entry in the syslog.conf file to point to a remote system. In this example, the name of the local system is example1. The remote system is remote1.

```bash
example1 # cat /etc/syslog.conf
... audit.notice @remote1
```

The audit.notice entry in the syslog.conf file on the remote1 system points to the log file.

```bash
remote1 # cat /etc/syslog.conf
... audit.notice /var/adm/auditlog
```

**Example 30–7**  Using Plugins in the audit_control File

The preferred method for specifying non-flags information in the audit_control file is to use the plugin entry. In this example, the audit flags are selected, then the plugin information is listed.

```bash
## audit_control file
flags:lo,ss
naflags:lo,na
plugin:name=audit_binfile.so; p_minfree=10; p_dir=/var/audit
plugin:name=audit_syslog.so; p_flags=+lo
```

**How to Change a User's Audit Characteristics**

Definitions for each user are stored in the audit_user database. These definitions modify, for the specified user, the preselected classes in the audit_control file. The nsswitch.conf file determines if a local file or if a naming service database is used. To calculate the user’s final audit preselection mask, see “Process Audit Characteristics” on page 641.

1. Assume the Primary Administrator role, or become superuser.

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in *System Administration Guide: Basic Administration.*
2 (Optional) Save a backup copy of the audit_user database.
   # cp /etc/security/audit_user /etc/security/audit_user.orig

3 Add new entries to the audit_user database.
   In the local database, each entry has the following format:
   username:always-audit:never-audit
   
   username      Selects the name of the user to be audited.
   always-audit  Selects the list of audit classes that should always be audited for
                 the specified user.
   never-audit   Selects the list of audit classes that should never be audited
                 for the specified user.

   You can specify multiple classes by separating the audit classes with commas.
   The audit_user entries are in effect at the user’s next login.

Example 30–8 Changing Which Events Are Audited for One User

In this example, the audit_control file contains the preselected audit classes for the system:

    ## audit_control file
    ...
    flags:lo,ss
    naflags:lo,na

   The audit_user file shows an exception. When the user jdoe uses a profile shell, that use is audited:

    ## audit_user file
    jdoe:pf

   The audit preselection mask for jdoe is a combination of the audit_user settings with the audit_control settings. The auditconfig -getaudit command shows the preselection mask for jdoe:

    # auditconfig -getaudit
    audit id = jdoe(1234567)
    process preselection mask = ss,pf,lo(0x13000,0x13000)
    terminal id (maj,min,host) = 242,511,example1(192.168.160.171)
    audit session id = 2138517656
Example 30–9  Auditing Users Only, Not the System

In this example, the login and role activities of four users only are audited on this system. The audit_control file does not preselect audit classes for the system.

```bash
## audit_control file
...
flags:
naflags:

The audit_user file preselects two audit classes for four users, as follows:

```bash
## audit_user file
jdoe:lo,pf
kdoe:lo,pf
pdoe:lo,pf
sdoe:lo,pf
```

The following audit_control file records unwarranted intrusion. In combination with the audit_user file, this file protects the system more than the first audit_control file in this example.

```bash
## audit_control file
...
flags:
naflags:lo
plugin:name=...
```

▼ How to Add an Audit Class

When you create your own audit class, you can place into it just those audit events that you want to audit for your site. When you add the class on one system, you should copy the change to all systems that are being audited.

1  **Assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2  **(Optional) Save a backup copy of the audit_class file.**

   # cp /etc/security/audit_class /etc/security/audit_class.orig
3  **Add new entries to the audit_class file.**

Each entry has the following format:

0x<number>:<name>:<description>

- **0x** Identifies *number* as hexadecimal.
- **number** Defines the unique audit class mask.
- **name** Defines the letter name of the audit class.
- **description** Defines the descriptive name of the audit class.

The entry must be unique in the file. Do not use existing audit class masks.

### Example 30–10  Creating a New Audit Class

This example creates a class to hold a small set of audit events. The added entry to the audit_class file is as follows:

0x10000000:pf:profile command

The entry creates a new audit class that is called *pf*. **Example 30–11** populates the new audit class.

#### Troubleshooting

If you have customized the audit_class file, make sure that any modifications to audit_user are consistent with the new audit classes. Errors occur when the audit classes in audit_user are not a subset of the audit_class database.

### How to Change an Audit Event's Class Membership

You might want to change an audit event's class membership to reduce the size of an existing audit class, or to place the event in a class of its own. When you reconfigure audit event-class mappings on one system, you should copy the change to all systems that are being audited.

1  **Assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2  **(Optional) Save a backup copy of the audit_event file.**

   # cp /etc/security/audit_event /etc/security/audit_event.orig
Change the class to which particular events belong by changing the `class-list` of the events.

Each entry has the following format:

```
number: name: description: class-list
```

- `number` is the audit event ID.
- `name` is the name of the audit event.
- `description` typically, the system call or executable that triggers the creation of an audit record.
- `class-list` is a comma-separated list of audit classes.

### Example 30–11 Mapping Existing Audit Events to a New Class

This example maps an existing audit event to the new class that was created in Example 30–10.

In the `audit_control` file, the binary audit record captures successes and failures of events in the `pf` class. The `syslog` audit log contains only failures of events in the `pf` class.

```
# grep pf | /etc/security/audit_class
0x10000000:pf:profile command
# vi /etc/security/audit_event
6180:AUE_prof_cmd:profile command:ua,as,pf
# vi audit_control
... flags:lo,pf
plugin:name=audit_binfile.so; p_dir=/var/audit; p_minfree=10
plugin:name=audit_syslog.so; p_flags=-lo,-pf
```

### Example 30–12 Auditing the Use of setuid Programs

This example creates a class to hold events that monitor calls to the `setuid` and `setgid` programs. The binary audit record captures successes and failures of events in the `lo` and `na` classes, and the successes of events in the `st` class. The `syslog` audit log contains only successes of events in the `st` class.

```
# vi /etc/security/audit_class
0x00000800:st:setuid class
# vi /etc/security/audit_event
26:AUE_SETGROUPS:setgroups(2):st
27:AUE_SETGRP:setgrp(2):st
40:AUE_SETREUID:setreuid(2):st
41:AUE_SETREGID:setregid(2):st
214:AUE_SETEGID:setegid(2):st
215:AUE_SETEUID:seteuid(2):st
```
Configuring and Enabling the Audit Service (Task Map)

The following task map points to procedures for configuring and enabling the audit service. The tasks are ordered.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. (Optional) Change the audit configuration files</td>
<td>Selects which events, classes, and users require auditing.</td>
<td>“Configuring Audit Files (Task Map)” on page 576</td>
</tr>
<tr>
<td>2. Create audit partitions</td>
<td>Creates disk space for the audit files, and protects them with file permissions.</td>
<td>“How to Create Partitions for Audit Files” on page 587</td>
</tr>
<tr>
<td>3. Create the audit_warn alias</td>
<td>Defines who should get email warnings when the audit service needs attention.</td>
<td>“How to Configure the audit_warn Email Alias” on page 589</td>
</tr>
<tr>
<td>4. (Optional) Change audit policy</td>
<td>Defines additional audit data that your site requires.</td>
<td>“How to Configure Audit Policy” on page 590</td>
</tr>
<tr>
<td>5. (Optional) Change audit queue parameters</td>
<td>Modifies the default queue parameters.</td>
<td>“How to Configure Audit Queue Parameters” on page 592</td>
</tr>
<tr>
<td>6. Configure auditing in non-global zones</td>
<td>Enable non-global zones to collect auditing records.</td>
<td>“Configuring the Audit Service in Zones (Tasks)” on page 598</td>
</tr>
<tr>
<td>7. Enable auditing</td>
<td>Turns on the audit service.</td>
<td>“How to Enable the Audit Service” on page 594</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example 30–17</td>
</tr>
<tr>
<td>8. (Optional) Disable auditing</td>
<td>Turns off the audit service.</td>
<td>“How to Disable the Audit Service” on page 595</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example 30–22</td>
</tr>
<tr>
<td>9. (Optional) Reread auditing configuration changes</td>
<td>Reads audit configuration changes into the kernel while the auditd daemon is running.</td>
<td>“How to Update the Audit Service” on page 596</td>
</tr>
</tbody>
</table>
Configuring and Enabling the Audit Service (Tasks)

After the configuration files have been set up for your site, you need to set up disk space for your audit files. You also need to set up other attributes of the audit service, and then enable the service. This section also contains procedures to refresh the audit service when you change configuration settings.

When a non-global zone is installed, you can choose to audit the zone exactly as the global zone is being audited. Alternatively, to audit the non-global zone individually, you can modify the audit configuration files in the non-global zone. To customize audit configuration files, see “Configuring Audit Files (Task Map)” on page 576.

How to Create Partitions for Audit Files

The following procedure shows how to create partitions for audit files, as well as the corresponding file systems and directories. Skip steps as necessary, depending on if you already have an empty partition, or if you have already mounted an empty file system.

1. **Assume the Primary Administrator role, or become superuser.**
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

2. **Determine the amount of disk space that is required.**
   Assign at least 200 Mbytes of disk space per host. However, how much auditing you require dictates the disk space requirements. So, your disk space requirements might be far greater than this figure. Remember to include a local partition for a directory of last resort.

3. **Create a storage pool and a mirror.**
   For more information, see “What Is ZFS?” in Solaris ZFS Administration Guide.
   ```bash
   # zpool create audit-pool mirror slice1 slice2
   ```
   For example, create the audit pool from two slices, and mirror them:
   ```bash
   # zpool create auditf mirror c0t4d0 c0t5d0
   ```
   If the local host is to be audited, also create an audit directory of last resort for the local host.

4. **Create a mount point for the audit files.**
   ```bash
   # zpool create -o mountpoint=/mountpoint /audit-pool/mountpoint
   ```
   For example, create the /audit mount point:
   ```bash
   # zfs create -o mountpoint=/audit auditf/audit
   ```
Create audit directories for every set of audit files that is going to be mounted.

# zfs create auditf/audit/machine1
# zfs create auditf/audit/machine1/files
# zfs create auditf/audit/machine2
# zfs create auditf/audit/machine2/files

For example, create directories for the noddy and blinken systems:

# zfs create auditf/audit/noddy
# zfs create auditf/audit/noddy/files
# zfs create auditf/audit/blinken
# zfs create auditf/audit/blinken/files

Protect the mount points.

The following ZFS properties are set to off:

# zfs set devices=off auditf/audit
# zfs set exec=off auditf/audit
# zfs set setuid=off auditf/audit

On a file server, define the file systems to be made available to other hosts.

# zfs set sharenfs=on audit-pool/mountpoint

For example, share the audit pool directory:

# zfs set sharenfs=on auditf/audit

As a result of sharing the auditf/audit directory, the following directories are shared:

/audit/noddy
/audit/noddy/files
/audit/blinken
/audit/blinken/files

(Optional) Remove the minimum free space threshold on the audit pool.

If you use the default configuration, a warning is generated when the directory is 80 percent full. The warning removes the reason to reserve free space on the audit pool.

# tunefs -m 0 /auditf/audit

For example, tune the audit pool directory:

# tunefs -m 0 auditf/audit
9 On a file server, restart the NFS service.
   If this command is the first share command or set of share commands that you have initiated, the NFS daemons might not be running.

   ■ If the NFS service is offline, enable the service.
     
     \[
     \%
     \text{svcs \"nfs\"}
     \]
     
     disabled Nov 02 svc:/network/nfs/rquota:default
     offline Nov 02 svc:/network/nfs/server:default
     
     # svcadm enable network/nfs/server

   ■ If the NFS service is running, restart the service.
     
     \[
     \%
     \text{svcs \"nfs\"}
     \]
     
     online Nov 02 svc:/network/nfs/client:default
     online Nov 02 svc:/network/nfs/server:default
     
     # svcadm restart network/nfs/server

For more information about the NFS service, refer to “Setting Up NFS Services” in System Administration Guide: Network Services. For information on managing persistent services, see Chapter 16, “Managing Services (Overview),” in System Administration Guide: Basic Administration and the \textit{smf}(5) man page.

\begin{itemize}
\item \textbf{How to Configure the audit\_warn Email Alias}
\item The \texttt{audit\_warn} script generates mail to an email alias that is called \texttt{audit\_warn}. To send this mail to a valid email address, you can follow one of the options that are described in \textbf{Step 2}:
\item \textbf{1 Assume the Primary Administrator role, or become superuser.}
   
   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in System Administration Guide: Basic Administration.

\item \textbf{2 Configure the audit\_warn email alias.}
   
   Choose one of the following options:
   
   \begin{itemize}
   \item \textbf{OPTION 1} – Replace the \texttt{audit\_warn} email alias with another email account in the \texttt{audit\_warn} script.
     
     Change the email alias in the following line of the script:
     
     ADDRESS=\texttt{audit\_warn} # standard alias for audit alerts

   \item \textbf{OPTION 2} – Redirect the \texttt{audit\_warn} email to another mail account.
     
     In this case, you would add the \texttt{audit\_warn} email alias to the appropriate mail aliases file. You could add the alias to the local /etc/mail/aliases file or to the mail\_aliases database in the name space. The new entry would resemble the following if the root mail account was made a member of the \texttt{audit\_warn} email alias:
# How to Configure Audit Policy

Audit policy determines the characteristics of the audit records for the local host. When auditing is enabled, the policies that you set by using the `auditconfig -setpolicy` command determine the audit policy.

You can inspect and change the current audit policy options with the `auditconfig` command. This command can set a temporary, or active policy. An active policy is a policy that is currently used by the kernel. This command also sets persistent policy, or configured policy. Configured policy is the policy that is restored when you restart the audit service.

1. **Assume a role that includes the Audit Control profile, or become superuser.**
   To create a role that includes the Audit Control profile and to assign the role to a user, see “Configuring RBAC (Task Map)” on page 174.

2. **Review the audit policy.**
   To view the settings, use the `auditconfig -getpolicy` command:
   ```
   $ auditconfig -getpolicy
   ```

3. **View the available policy options.**
   ```
   $ auditconfig -lspolicy
   ```

   **Note** – The perzone and ahlh policy options can be set only in the global zone.

4. **Enable or disable selected audit policy options.**
   ```
   # auditconfig [ -t ] -setpolicy prefix policy
   
   -t Optional. Creates a temporary, or active policy. The policy setting is not restored when you restart the audit service.
   
   prefix A prefix value of + enables the policy option. A prefix value of - disables the policy option.
   
   policy Selects the policy to be enabled or to be disabled.
   ```

   A temporary (-t) policy is in effect until the the audit service is restarted, or until the policy is modified by the `auditconfig -setpolicy` command. Without the -t option, the policy setting persists across restarts of the audit service.

   For a description of each policy option, see “Determining Audit Policy” on page 568.
Example 30–13  Setting the cnt and ahlt Audit Policy Options

In this example, the cnt policy is disabled, and the ahlt policy is enabled. With these settings, system use is halted when the audit partitions are full and an asynchronous event occurs. When a synchronous event occurs, the process that created the thread hangs. These settings are appropriate when security is more important than availability.

The following auditconfig policy commands disable the cnt policy option and enable the ahlt policy option:

# auditconfig -setpolicy -cnt
# auditconfig -setpolicy +ahlt

These settings persist until you change them.

Example 30–14  Setting an Audit Policy Temporarily

In this example, the audit service is running and the ahlt audit policy is configured. The administrator adds the seq audit policy to the active policy, but does not configure the audit service to use the seq audit policy permanently. The seq policy is useful for debugging the audit service when audit records are corrupted, or when records are being dropped.

The + prefix adds the seq option to the audit policy, rather than replaces the current audit policy with seq. The -t option makes the policy active in the kernel.

$ auditconfig -setpolicy none
$ auditconfig -getpolicy
  configured audit policies = none
  active audit policies = none
$ auditconfig -setpolicy ahlt
$ auditconfig -getpolicy
  configured audit policies = ahlt
  active audit policies = ahlt
$ auditconfig -t -setpolicy +seq
  configured audit policies = ahlt,seq
  active audit policies = ahlt,seq

The administrator unsetsthe seq policy when the debugging is completed:

$ auditconfig -setpolicy -seq
$ auditconfig -getpolicy
  configured audit policies = ahlt
  active audit policies = ahlt
**Example 30–15 Setting the per zone Audit Policy**

In this example, the per zone audit policy is set in the global zone. When a zone boots, the non-global zone collects audit records according to the audit configuration settings in its zone. The per zone policy setting is stored as a property of the audit service, so it is in effect during the session and when the audit service is restarted.

```
$ auditconfig -setpolicy none
$ auditconfig -getpolicy
  configured audit policies = none
  active audit policies = none
$ auditconfig -setpolicy +perzone
$ auditconfig -getpolicy
  configured audit policies = perzone
  active audit policies = perzone
```

**How to Configure Audit Queue Parameters**

The audit service provides default values for audit queue parameters. You can inspect and set these values with the `auditconfig` command.

The `auditconfig` command can set a temporary, or *active*, value. This value is used by the kernel, but it is not set as a property of the audit service. This command also sets persistent, or *configured*, values. Configured values are the values that are restored when you restart the audit service.

1. **Assume a role that includes the Audit Control profile, or become superuser.**
   
   To create a role that includes the Audit Control profile and to assign the role to a user, see “Configuring RBAC (Task Map)” on page 174.

2. **Review the audit queue parameter values.**
   
   To view the values, use the `auditconfig -getqctrl` command.
   
   ```bash
   $ auditconfig -getqctrl
   no configured audit queue lowater mark
   no configured ...
   ```

   The string `no configured` indicates that the system is using the default settings. For a description of the audit queue parameters, see the `auditconfig(1M)` man page.

3. **Modify selected audit queue parameters.**

   - To modify all audit queue parameters, use the `-setqctrl` option.

   ```bash
   # auditconfig [ -t ] -setqctrl hiwater lowater bufsz interval
   ```
To modify a specific audit queue parameter, use the specific option, one of -setqbufsz, -setqdelay, -setqlowater, and -setqhiwater.

```
# auditconfig [-t] -setq* value
```

- **t**  Optional. Sets a temporary, or *active*, value. This value is not set as a property value of the audit service.

**hiwater...**  Are audit queue parameters.

**value**  Is a value for the audit queue parameter.

A temporary (-t) or active value is in effect until the audit service is restarted, or until the value is modified by the `auditconfig [-t] -setq*` command. Without the -t option, the queue parameter value is stored as a property value of the audit service.

### Example 30–16 Configuring Audit Queue Parameters

In this example, the administrator configures the audit queue’s write buffer size and wakeup interval. The administrator tunes these parameters differently in the active configuration by using the -t option.

```
# auditconfig -getqctrl
no configured audit queue hiwater mark
no configured audit queue lowater mark
no configured audit queue buffer size
no configured audit queue delay
active audit queue hiwater mark (records) = 110
active audit queue lowater mark (records) = 30
active audit queue buffer size (bytes) = 8192
active audit queue delay (ticks) = 20
# auditconfig -setqbufsz 8192
# auditconfig -setqdelay 20
# auditconfig -t -setqbufsz 12288
# auditconfig -t -setqdelay 25
# auditconfig -getqctrl
no configured audit queue lowater mark
no configured audit queue hiwater mark
configured audit queue buffer size (bytes) = 8192
configured audit queue delay (ticks) = 20
active audit queue hiwater mark (records) = 100
active audit queue lowater mark (records) = 10
active audit queue buffer size (bytes) = 12288
active audit queue delay (ticks) = 25
```

The administrator resets the parameters to their default values by setting them to zero.
How to Enable the Audit Service

This procedure enables the audit service for all zones. To start the audit daemon in a non-global zone, see Example 30–17.

When auditing is configured securely, the system is in single-user mode until auditing is enabled. You can also enable auditing in multiuser mode.

Before You Begin

You should perform this procedure as superuser after completing the following tasks:

- Planning – “Planning Solaris Auditing (Task Map)” on page 563
- Customizing audit files – “Configuring Audit Files (Task Map)” on page 576
- Setting up audit partitions – “How to Create Partitions for Audit Files” on page 587
- Setting up audit warning messages – “How to Configure the audit_warn Email Alias” on page 589
- Setting audit policy – “How to Configure Audit Policy” on page 590

1 Run the script that enables the audit service.

Go to the /etc/security directory, and execute the bsmconv script there.

```
# cd /etc/security
#. /bsmconv
```

This script is used to enable the Basic Security Module (BSM).

```
Shall we continue with the conversion now? [y/n] y
bsmconv: INFO: turning on audit module.
bsmconv: INFO: initializing device allocation.
```

The Basic Security Module is ready.
If there were any errors, please fix them now. Configure BSM by editing files located in /etc/security. Reboot this system now to come up with BSM enabled.

For the effects of the script, see the `bsmconv(1M)` man page.

2 **Reboot the system.**

```bash
# reboot
```

The `auditd` daemon starts the audit service when the system enters multiuser mode. The FMRI for the audit service is `svc:/system/auditd:default`.

Another effect of the script is to turn on device allocation. To configure device allocation, see “Managing Device Allocation (Task Map)” on page 87.

**Example 30–17 Enabling Auditing in a Non-Global Zone**

In the following example, the global zone administrator turned on per zone policy after auditing was enabled in the global zone and after the non-global zone had booted. The zone administrator of the non-global zone has configured the audit files for the zone, and then starts the audit daemon in the zone.

```bash
zone1# svcadm enable svc:/system/auditd
```

**How to Disable the Audit Service**

If the audit service is no longer required at some point, this procedure returns the system to the system state before auditing was enabled. If non-global zones are being audited, their audit service is also disabled.

**Caution** – This command also disables device allocation. Do not run this command if you want to be able to allocate devices. To disable auditing and retain device allocation, see Example 30–18.

1 **Become superuser and bring the system into single-user mode.**

```bash
% su
Password: <Type root password>
# init 5
```

For more information, see the `init(1M)` man page.
2 Run the script to disable auditing.
   Change to the /etc/security directory, and execute the bsmunconv script.
   
   # cd /etc/security
   # ./bsmunconv
   
   Another effect of the script is to disable device allocation.
   For information on the full effect of the bsmunconv script, see the bsmconv(1M) man page.

3 Bring the system into multiuser mode.
   
   # init 6

Example 30–18 Disabling Auditing and Keeping Device Allocation

   In this example, the audit service stops collecting records, but device allocation continues to work. All values from the flags, naflags, and plugin entries in the audit_control file are removed, as are all user entries in the audit_user file.

   ## audit_control file
   flags:
   naflags:

   ## audit_user file
   
   The auditsd daemon runs, but no audit records are kept.

Example 30–19 Disabling Auditing on a Per-Zone Basis

   In this example, the audit service stops running in zone1 where the audit service is disabled. Device allocation continues to work. When this command is run in the global zone, and the perzone audit policy is not set, auditing is disabled for all zones, not just the global zone.

   zone1 # audit -t

▼ How to Update the Audit Service

   This procedure restarts the auditsd daemon when you have made changes to audit configuration files after the daemon has been running.

1 Assume a role that includes the Audit Control rights profile, or become superuser.
   To create a role that includes the Audit Control rights profile and assign the role to a user, see “Configuring RBAC (Task Map)” on page 174.
2 Choose the appropriate command.

- If you modify the naflags line in the audit_control file, change the kernel mask for nonattributable events.
  
  $ /usr/sbin/auditconfig -aconf
  
  You can also reboot.

- If you modify other lines in the audit_control file, reread the audit_control file.
  The audit daemon stores information from the audit_control file internally. To use the new information, either reboot the system or instruct the audit daemon to read the modified file.
  
  $ /usr/sbin/audit -s

  **Note** – Audit records are generated based on the audit preselection mask that is associated with each process. Executing audit -s does *not* change the masks in existing processes. To change the preselection mask for an existing process, you must restart the process. You can also reboot.

  The audit -s command causes the audit daemon to re-read the directory and minfree values from the audit_control file. The command changes the generation of the preselection mask for processes spawned by subsequent logins.

- If you modify the audit_event file or the audit_class file while the audit daemon is running, refresh the audit service.
  
  Read the modified event-class mappings into the system, and ensure that each user who uses the machine is correctly audited.
  
  $ auditconfig -conf
  
  $ auditconfig -setumask auid classes

  *auid*    Is the user ID.

  *classes*  Are the preselected audit classes.

  For an example, see “How to Modify a User’s Preselection Mask” on page 619.

### Example 30–20 Restarting the Audit Daemon

In this example, the system is brought down to single-user mode, then back up to multiuser mode. When the system is brought into multiuser mode, modified audit configuration files are read into the system.

```bash
# init 5
# init 6
```
Configuring the Audit Service in Zones (Tasks)

The audit service audits the entire system, including audit events in zones. A system that has installed non-global zones can audit all zones identically, or can control auditing per zone. For background, see “Auditing on a System With Zones” on page 560. To plan, see “How to Plan Auditing in Zones” on page 564.

▼ How to Configure All Zones Identically for Auditing

This procedure enables audits every zone identically. This method requires the least computer overhead and administrative resources.

1 Configure the global zone for auditing.
   a. Complete the tasks in “Configuring Audit Files (Task Map)” on page 576.
   b. Complete the tasks in “Configuring and Enabling the Audit Service (Task Map)” on page 586, with the following exceptions.
      ■ Do not enable per zone audit policy.
      ■ Do not enable the audit service. You enable the audit service after you have configured the non-global zones for auditing.

2 Copy the audit configuration files from the global zone to every non-global zone.
   Copy any of the following files that you have edited: audit_class, audit_control, audit_event, audit_user. Do not copy audit_warn. You do not have to copy files that you have not edited.
   You have two options. As superuser, you can copy the files, or loopback mount the files. The non-global zone must be running.

   ■ Copy the files.
      a. From the global zone, list the /etc/security directory in the non-global zone.
         # ls /zone/zonename/etc/security/
      b. Copy the audit configuration files to the zone’s /etc/security directory.
         # cp /etc/security/audit-file /zone/zonename/etc/security/audit-file
         Later, if you modify an audit configuration file in the global zone, you re-copy the file to the non-global zones.
- Loopback mount the configuration files.

  a. From the global zone, halt the non-global zone.
     
     ```
     # zoneadm -z non-global-zone halt
     ```

  b. Create a read-only loopback mount for every audit configuration file that you modified in the global zone.
     
     ```
     # zonecfg -z non-global-zone
     add fs
     set special=/etc/security/audit-file
     set dir=/etc/security/audit-file
     set type=lofs
     add options [ro,nodevices,nosetuid]
     end
     ```

  c. To make the changes effective, boot the non-global zone.
     
     ```
     # zoneadm -z non-global-zone boot
     ```

     You can also reboot the system.

     Later, if you modify an audit configuration file in the global zone, you reboot the system to refresh the loopback-mounted files in the non-global zones.

**Example 30–21 Loopback Mounting Audit Configuration Files**

In this example, the system administrator has modified the `audit_class`, `audit_event`, `audit_control`, `audit_user`, and `audit_warn` files.

The `audit_warn` file is read in the global zone only, so does not have to be loopback mounted into the non-global zones.

On this system, `machine1`, the administrator has created two non-global zones, `machine1-webserver` and `machine1-appserver`. The administrator has finished customizing the audit configuration files. If the administrator later modifies the files, the system will be rebooted to make the changes effective.

```bash
# zoneadm -z machine1-webserver halt
# zoneadm -z machine1-appserver halt
# zonecfg -z machine1-webserver
add fs
set special=/etc/security/audit_class
set dir=/etc/security/audit_class
set type=lofs
add options [ro,nodevices,nosetuid]
end
```
When the zones are rebooted, the audit configuration files are read-only in the zones.

### How to Configure Per-Zone Auditing

This procedure enables separate zone administrators to control the audit service in their zone. For the complete list of policy options, see the `auditconfig(1M)` man page.

1. **In the global zone, configure auditing, but do not enable the audit service.**

   a. Complete the tasks in “Configuring Audit Files (Task Map)” on page 576.

   b. Complete the tasks in “Configuring and Enabling the Audit Service (Task Map)” on page 586, with the following exceptions.

      - Add the `per zone` audit policy. For an example, see Example 30–15.
Do not enable the audit service. You enable the audit service after the non-global zones are configured for auditing.

2 In each non-global zone, configure the audit files.

   Note – If you are planning to disable auditing in the non-global zone, you can skip this step. To disable auditing, see Example 30–22.

   a. Complete the tasks in “Configuring Audit Files (Task Map)” on page 576.
   b. Follow the procedures that are described in “Configuring and Enabling the Audit Service (Task Map)” on page 586.
   c. Do not configure system-wide audit settings.
      Specifically, do not add the per zone or auth policy to the non-global zone. And do not run the bsmconv command from the non-global zone.
   d. Enable auditing in your zone.
      When the global zone reboots after auditing is configured, auditing is automatically enabled in your zone.
      If the global zone administrator activates the per zone audit policy after the system is booted, individual zone administrators must enable auditing. For details, see Example 30–17.

3 In the global zone, enable the audit service.
For the procedure, see “How to Enable the Audit Service” on page 594.

Example 30–22 Disabling Auditing in a Non-Global Zone

This example works if the global zone has set the per zone audit policy. The zone administrator of the noaudit zone disables auditing for that zone. Because the administrator planned to disable auditing, she did not edit the audit configuration files.

`noauditzone # svcadm disable svc:/system/auditd`

Managing Audit Records (Task Map)

The following task map points to procedures for selecting, analyzing, and managing audit records.
Managing Audit Records

By managing the audit trail, you can monitor the actions of users on your network. Auditing can generate large amounts of data. The following tasks show you how to work with all this data.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display the formats of audit records</td>
<td>Shows the kind of information that is collected for an audit event, and the order in which the information is presented.</td>
<td>“How to Display Audit Record Formats” on page 602</td>
</tr>
<tr>
<td>Merge audit records</td>
<td>Combines audit files from several machines into one audit trail.</td>
<td>“How to Merge Audit Files From the Audit Trail” on page 604</td>
</tr>
<tr>
<td>Select records to examine</td>
<td>Selects particular events for study.</td>
<td>“How to Select Audit Events From the Audit Trail” on page 606</td>
</tr>
<tr>
<td>Display audit records</td>
<td>Enables you to view binary audit records.</td>
<td>“How to View the Contents of Binary Audit Files” on page 608</td>
</tr>
<tr>
<td>Clean up incorrectly named audit files</td>
<td>Provides an end timestamp to audit files that were inadvertently left open by the audit service.</td>
<td>“How to Clean Up a not_terminated Audit File” on page 610</td>
</tr>
<tr>
<td>Prevent audit trail overflow</td>
<td>Prevents the audit file systems from becoming full.</td>
<td>“How to Prevent Audit Trail Overflow” on page 611</td>
</tr>
</tbody>
</table>

Managing Audit Records

To write scripts that can find the audit data that you want, you need to know the order of tokens in an audit event. The `auditrecord` command displays the audit event number, audit class, selection mask, and record format of an audit event.

- **Put the format of all audit event records in an HTML file.**

  The -a option lists all audit event record formats. The -h option puts the list in HTML format that can be displayed in a browser.

  ```
  % auditrecord -a -h > audit.events.html
  ```

  When you display the *html file in a browser, use the browser's Find tool to find specific records.

  For more information, see the `auditrecord(1M)` man page.
Example 30–23  Displaying the Audit Record Formats of a Program

In this example, the format of all audit records that are generated by the `login` program are displayed. The login programs include `rlogin`, `telnet`, `newgrp`, `role login` to the Solaris Management Console, and Solaris Secure Shell.

```bash
% auditrecord -p login
terminal login program /usr/sbin/login See login(1)
   /usr/dt/bin/dtlogin See dtlogin event ID 6152 AUE_login
class lo (0x00001000) header subject
   text error message or "successful login"
return
login: logout
   program various See login(1)
event ID 6153 AUE_logout
...

newgrp
   program newgrp See newgrp login
event ID 6212 AUE_newgrp_login
...

rlogin
   program /usr/sbin/login See login(1) - rlogin
event ID 6155 AUE_rlogin
...

SMC: role login
   program SMC server See role login
event ID 6173 AUE_role_login
...

/usr/lib/ssh/sshd
   program `/usr/lib/ssh/sshd` See login - ssh
event ID 6172 AUE_ssh
...

telnet login
   program `/usr/sbin/login` See login(1) - telnet
event ID 6154 AUE_telnet
...
```

Example 30–24  Displaying the Audit Record Formats of an Audit Class

In this example, the format of all audit records in the `fd` class are displayed.

```bash
% auditrecord -c fd
```
How to Merge Audit Files From the Audit Trail

By merging all audit files in all the audit directories, you can analyze the contents of the entire audit trail. The `auditreduce` command merges all the records from its input files into a single output file. The input files can then be deleted. When the output file is placed in a directory that is named `/etc/security/auditserver-name/files`, the `auditreduce` command can find the output file without your specifying the full path.

Note – This procedure applies only to binary audit records.

1 Assume a role that includes the Audit Review profile, or become superuser.

The System Administrator role includes the Audit Review profile. You can also create a separate role that includes the Audit Review profile. To create a role and assign the role to a user, see “Configuring RBAC (Task Map)” on page 174.

2 Create a directory for storing merged audit files.

   # mkdir audit-trail-directory

3 Limit access to the directory.

   # chmod 700 audit-trail-directory
   # ls -la audit-trail-directory
   drwx------ 3 root  sys  512 May 12 11:47 .
   drwxr-xr-x 4 root  sys 1024 May 12 12:47 ..

```bash
rmdir
system call rmdir See rmdir(2)
event ID 48 AUE_RMDIR
class fd (0x00000020)
  header
  path
  [attribute]
  subject
  [use_of_privilege]
  return
```

```bash
unlink
system call unlink See unlink(2)
event ID 6 AUE_UNLINK
...
```

```bash
unlinkat
system call unlinkat See openat(2)
event ID 286 AUE_UNLINKAT
...
```
4 Merge the audit records in the audit trail.

Change directories to the `audit-trail-directory` and merge the audit records into a file with a named suffix. All directories that are listed in the `dir` lines of the `audit_control` file on the local system are merged.

```bash
# cd audit-trail-directory
# auditreduce -Uppercase-option -O suffix
```

The uppercase options to the `auditreduce` command manipulate files in the audit trail. The uppercase options include the following:

- `-A` Selects all of the files in the audit trail.
- `-C` Selects complete files only. This option ignores files with the suffix `not_terminated`.
- `-M` Selects files with a particular suffix. The suffix can be a machine name, or it can be a suffix that you have specified for a summary file.
- `-O` Creates an audit file with 14-character timestamps for both the start time and the end time, with the suffix `suffix` in the current directory.

### Example 30–25 Copying Audit Files to a Summary File

In the following example, the System Administrator role, `sysadmin`, copies all files from the audit trail into a merged file.

```
$ whoami
sysadmin
$ mkdir /var/audit/audit_summary.dir
$ chmod 700 /var/audit/audit_summary.dir
$ cd /var/audit/audit_summary.dir
$ auditreduce -A -O All
$ ls *All
20030827183214.20030827215318.All
```

In the following example, only complete files are copied from the audit trail into a merged file.

```
$ cd /var/audit/audit_summary.dir
$ auditreduce -C -O Complete
$ ls *Complete
20030827183214.20030827214217.Complete
```

In the following example, only complete files are copied from the `example1` machine into a merged file.

```
$ cd /var/audit/audit_summary.dir
$ auditreduce -M example1 -O example1summ
```
Example 30–26  Moving Audit Files to a Summary File

The -D option to the auditreduce command deletes an audit file when you copy it to another location. In the following example, the complete audit files from one system are copied to the summary directory for later examination.

$ cd /var/audit/audit_summary.dir
$ auditreduce -C -O daily_example1 -D example1
$ ls *example1
20030827183214.20030827214217.daily_example1

The audit files from the example1 system that were the input to the *daily_example1 file are removed when this command successfully completes.

▼ How to Select Audit Events From the Audit Trail

You can filter audit records for examination. For the complete list of filtering options, see the auditreduce(1M) man page.

1 Assume a role that includes the Audit Review profile, or become superuser.

The System Administrator role includes the Audit Review profile. You can also create a separate role that includes the Audit Review profile. To create a role and assign the role to a user, see “Configuring RBAC (Task Map)” on page 174.

2 Select the kinds of records that you want from the audit trail, or from a specified audit file.

   auditreduce  -lowercase-option argument  [optional-file]

     argument  Specific argument that a lowercase option requires. For example, the -c option requires an argument of an audit class, such as ua.

     -d  Selects all of the events on a particular date. The date format for argument is yyyymmdd. Other date options, -b and -a, select events before and after a particular date.

     -u  Selects all of the events attributable to a particular user. The argument is a user name. Another user option, -e, selects all of the events attributable to an effective user ID.

     -c  Selects all of the events in a preselected audit class. The argument is an audit class name.
Select all of the instances of a particular audit event. The argument is an audit event.

optional-file
Is the name of an audit file.

Example 30–27 Combining and Reducing Audit Files
The auditreduce command can eliminate the less interesting records as it combines the input files. For example, you might use the auditreduce command to retain only the login and logout records in audit files that are over a month old. If you need to retrieve the complete audit trail, you could recover the trail from backup media.

# cd /var/audit/audit_summary.dir
# auditreduce -O lo.summary -b 20030827 -c lo; compress *lo.summary

Example 30–28 Copying NSA Audit Records to a Summary File
In this example, all the records of nonattributable audit events in the audit trail are collected into one file.

$ whoami
sysadmin
$ cd /var/audit/audit_summary.dir
$ auditreduce -c na -O nasumm
$ ls *nasumm
20030827183214.20030827215318.nasumm

The merged nasumm audit file is time stamped with the beginning and ending date of the na records.

Example 30–29 Finding Audit Events in a Specified Audit File
You can select audit files manually to search just the named set of files. For example, you can further process the *nasumm file in the previous example to find system boot events. To do so, you would specify the file name as the final argument to the auditreduce command.

$ auditreduce -m 113 -O systemboot 20030827183214.20030827215318.nasumm
20030827183214.20030827215318.systemboot

The 20030827183214.20030827183214.systemboot file contains only system boot audit events.
Example 30–30  Copying One User’s Audit Records to a Summary File

In this example, the records in the audit trail that contain the name of a particular user are merged. The -e option finds the effective user. The -u option finds the audit user.

```
$ cd /var/audit/audit_summary.dir
$ auditreduce -e tamiko -O tamiko
```

You can look for specific events in this file. In the following example, what time the user logged in and out on Sept 7, 2003, your time, is checked. Only those files with the user’s name as the file suffix are checked. The short form of the date is yyyyymmdd.

```
# auditreduce -M tamiko -O tamikolo -d 20030907 -u tamiko -c lo
```

Example 30–31  Copying Selected Records to a Single File

In this example, login and logout messages for a particular day are selected from the audit trail. The messages are merged into a target file. The target file is written in a directory other than the normal audit root directory.

```
# auditreduce -c lo -d 20030827 -O /var/audit/audit_summary.dir/logins
# ls /var/audit/audit_summary.dir/*logins
/var/audit/audit_summary.dir/20030827183936.20030827232326.logins
```

How to View the Contents of Binary Audit Files

The praudit command enables you to view the contents of binary audit files. You can pipe the output from the auditreduce command, or you can read a particular audit file. The -x option is useful for further processing.

1 Assume a role that includes the Audit Review profile, or become superuser.
   The System Administrator role includes the Audit Review profile. You can also create a separate role that includes the Audit Review profile. To create a role and assign the role to a user, see “Configuring RBAC (Task Map)” on page 174.

2 Use one of the following praudit commands to produce the output that is best for your purposes.
   The following examples show praudit output from the same audit event. Audit policy has been set to include the sequence and trailer tokens.
   - The praudit -s command displays audit records in a short format, one token per line. Use the -l option to place each record on one line.
The `praudit -r` command displays audit records in their raw format, one token per line. Use the `-l` option to place each record on one line.

```
$ auditreduce -c lo | praudit -r
21,101,2,6155,0x0000,192.168.60.83,1062021202,64408258
36,2026700,2026700,10,2026700,10,749,749,195 1234 192.168.60.17
40,successful login
39,0,0
47,1298
```

The `praudit -x` command displays audit records in XML format, one token per line. Use the `-l` option to place the XML output for one record on one line.

```
$ auditreduce -c lo | praudit -x
<record version="2" event="login - rlogin" host="example1" time="Wed Aug 27 14:53:22 PDT 2003" msec="64">
  <subject audit-uid="jdoe" uid="jdoe" gid="staff" ruid="jdoe" rgid="staff" pid="749" sid="749" tid="195 1234 server1"/>
  <text>successful login</text>
  <return errval="success" retval="0"/>
  <sequence seq-num="1298"/>
</record>
```

**Example 30–32** Printing the Entire Audit Trail

With a pipe to the `lp` command, the output for the entire audit trail goes to the printer. The printer should have limited access.

```
# auditreduce | praudit | lp -d example.protected.printer
```

**Example 30–33** Viewing a Specific Audit File

In this example, a summary login file is examined in a terminal window.

```
# cd /var/audit/audit_summary.dir/logins
# praudit 20030827183936.20030827232326.logins | more
```
Example 30–34  Putting Audit Records in XML Format

In this example, the audit records are converted to XML format.

```
# praudit -x 20030827183214.20030827215318.logins > 20030827.logins.xml
```

The *xml file can be displayed in a browser. The contents of the file can be operated on by a script to extract the relevant information.

Troubleshooting

A message similar to the following indicates that you do not have enough privilege to use the `praudit` command:

```
praudit: Can't assign 20090408164827.20090408171614.example1 to stdin.
```

▼ How to Clean Up a not_terminated Audit File

Occasionally, an audit daemon exits while its audit file is still open. Or, a server becomes inaccessible and forces the machine to switch to a new server. In such instances, an audit file remains with the string not_terminated as the end timestamp, even though the file is no longer used for audit records. Use the `auditreduce -O` command to give the file the correct timestamp.

1. **List the files with the not_terminated string on your audit file system in order of creation.**
   
   ```
   # ls -Rlt audit-directory*/files/* | grep not_terminated
   -R   Lists files in subdirectories.
   -t   Lists files from most recent to oldest.
   -l   Lists the files in one column.
   ```

2. **Clean up the old not_terminated file.**
   
   Specify the name of the old file to the `auditreduce -O` command.
   
   ```
   # auditreduce -O system-name old-not-terminated-file
   ```

3. **Remove the old not_terminated file.**
   
   ```
   # rm system-name old-not-terminated-file
   ```

Example 30–35  Cleaning Up Closed not_terminated Audit Files

In the following example, not_terminated files are found, renamed, then the originals are removed.
ls -Rlt */files/* | grep not_terminated
.../egret.1/20030908162220.not_terminated.egret
.../egret.1/20030827215359.not_terminated.egret
# cd */files/egret.1
# auditreduce -O egret 20030908162220.not_terminated.egret
# ls -1t
20030908162220.not_terminated.egret Current audit file
20030827230920.20030830000909.egret Input (old) audit file
20030827215359.not_terminated.egret
# rm 20030827215359.not_terminated.egret
# ls -1t
20030908162220.not_terminated.egret Current audit file
20030827230920.20030830000909.egret Cleaned up audit file

The start timestamp on the new file reflects the time of the first audit event in the not_terminated file. The end timestamp reflects the time of the last audit event in the file.

▼ How to Prevent Audit Trail Overflow

If your security policy requires that all audit data be saved, do the following:

1 Set up a schedule to regularly archive audit files.
   Archive audit files by backing up the files to offline media. You can also move the files to an archive file system.
   If you are collecting text audit logs with the syslog utility, archive the text logs. For more information, see the logadm(1M) man page.

2 Set up a schedule to delete the archived audit files from the audit file system.

3 Save and store auxiliary information.
   Archive information that is necessary to interpret audit records along with the audit trail.

4 Keep records of which audit files have been archived.

5 Store the archived media appropriately.

6 Reduce the volume of audit data that you store by creating summary files.
   You can extract summary files from the audit trail by using options to the auditreduce command. The summary files contain only records for specified types of audit events. To extract summary files, see Example 30–27 and Example 30–31.
Troubleshooting Solaris Auditing (Tasks)

This section covers various Solaris auditing error messages, preferences, and the auditing that is provided by other tools. These procedures can help you record the audit events that you require at your site.

Troubleshooting Solaris Auditing (Task Map)

The following task map points to procedures for troubleshooting Solaris auditing.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Solution</th>
<th>For Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Why are audit files not being created when I have configured auditing?</td>
<td>Troubleshoot the audit daemon and audit configuration files.</td>
<td>“How to Determine That Solaris Auditing Is Running” on page 612</td>
</tr>
<tr>
<td>How can I reduce the amount of audit information that is being collected?</td>
<td>Audit just the events that you want to audit.</td>
<td>“How to Lessen the Volume of Audit Records That Are Produced” on page 615</td>
</tr>
<tr>
<td>How can I audit everything that a user does on the system?</td>
<td>Audit one or more users for every command.</td>
<td>“How to Audit All Commands by Users” on page 617</td>
</tr>
<tr>
<td>How can I change the audit events that are being recorded and have the change affect existing sessions?</td>
<td>Update a user’s preselection mask</td>
<td>“How to Modify a User’s Preselection Mask” on page 619</td>
</tr>
<tr>
<td>How can I locate modifications to particular files?</td>
<td>Audit file modifications, then use the audit reduce command to find particular files.</td>
<td>“How to Find Audit Records of Changes to Specific Files” on page 619</td>
</tr>
<tr>
<td>How can I reduce the size of my audit files?</td>
<td>Limit the size of the binary audit file.</td>
<td>“How to Limit the Size of Binary Audit Files” on page 622</td>
</tr>
<tr>
<td>How can I remove audit events from the audit_event file?</td>
<td>Update the audit_event file.</td>
<td>“How to Prevent the Auditing of Certain Events” on page 621</td>
</tr>
<tr>
<td>How can I audit all logins to a Solaris system?</td>
<td>Audit logins from any system.</td>
<td>“How to Audit Logins From Other OSES” on page 622</td>
</tr>
<tr>
<td>Why are auditing records not being kept for my FTP transfers?</td>
<td>Use the appropriate auditing tool for utilities that generate their own logs.</td>
<td>“How to Audit FTP and SFTP File Transfers” on page 622</td>
</tr>
</tbody>
</table>

▼ How to Determine That Solaris Auditing Is Running

If you believe that auditing has been activated, but no audit records are in your primary audit directory, try the following.
1 Determine that auditing is running.

- **Verify that the c2audit kernel module is loaded.**
  
  ```shell
  # modinfo | grep c2audit
  ```
  
  No listing indicates that auditing is not running. The following listing indicates that auditing is running:
  
  ```shell
  40 132ce90 14230 186 1 c2audit (C2 system call)
  ```

- **Verify that the audit daemon is running.**
  
  Verify the status of the auditd service. The following listing indicates that auditing is not running:
  
  ```shell
  # svcs -x auditd
  svc:/system/auditd:default (Solaris audit daemon)
  State: disabled since Fri Aug 14 19:02:35 2009
  Reason: Disabled by an administrator.
  See: http://sun.com/msg/SMF-8000-05
  See: auditd(1M)
  See: audit(1M)
  Impact: This service is not running.
  ```
  
  The following listing indicates that the audit service is running:
  
  ```shell
  # svcs auditd
  STATE STIME FMRI
  online 10:10:10 svc:/system/auditd:default
  ```

- **Verify the current audit condition.**
  
  The following listing indicates that auditing is not running:
  
  ```shell
  # auditconfig -getcond
  auditconfig: auditon(2) failed.
  auditconfig: error = Operation not supported(48)
  ```
  
  The following listing indicates that auditing is running:
  
  ```shell
  # auditconfig -getcond
  audit condition = auditing
  ```
  
  If the audit service is not running, enable it. For the procedure, see “How to Enable the Audit Service” on page 594.

2 **Verify the syntax of the audit_control file.**

  ```shell
  # audit -v /etc/security/audit_control
  audit: audit_control must have either a valid "dir:" entry
  or a valid "plugin:" entry with "p.dir:" specified.
  ```
Correct the errors. The message syntax ok indicates that the file is syntactically correct.

3 **Verify that the audit_control file has valid values for the flags and naflags keywords.**

```
# grep flags /etc/security/audit_control
flags:lo
naflags:na,lp
```

Supply valid values if the audit_control file has invalid values. In the preceding example, lp is an invalid class.

4 **Verify that the audit_user file has valid values for every user.**

```
# tail audit_user
...
# User Level Audit User File
#
# File Format
#
#   username:always:never
#
root:lo:no
admin:lp:no
```

Supply valid values if the audit_user file contains invalid values. In the preceding example, lp is an invalid class.

5 **If you created a customized audit class, verify that you assigned events to the class.**

For example, the following audit_control file contains a class that Sun did not deliver:

```
# grep flags /etc/security/audit_control
flags:lo, pf
naflags:na,lo
```

For a description of creating the pf class, see “How to Add an Audit Class” on page 583.

a. **Verify that the class is defined in the audit_class file.**

The audit class mask must be unique.

```
# grep pf /etc/security/audit_class
0x10000000:pf:profile command
```

If the class is not defined, define it. Otherwise, remove the class from the audit_control and audit_user files.

b. **Verify that events have been assigned to the class.**

```
# grep pf /etc/security/audit_event
6180:AUE_prof_cmd:profile command:ua,as,pf
```

If events are not assigned to the class, assign the appropriate events to this class.
6 If the previous steps did not indicate a problem, review the system log files, 
/var/adm/messages and /var/log/syslog.

   a. Locate and fix the problems.

   b. Then, if the audit service is running, restart it.
      
      # audit -s

   c. If the audit service is not running, enable it.
      
      For the procedure, see “How to Enable the Audit Service” on page 594.

▼ How to Lessen the Volume of Audit Records That Are Produced

After you have determined which events must be audited at your site, use the following 
suggestions to create manageable audit files.

1 Use the default audit policy.

Specifically, avoid adding events and audit tokens to the audit trail. The following policies affect 
the size of the audit trail.

   ▪ large policy – Adds environment variables to exec audit events.
   ▪ argv policy – Adds command parameters to exec audit events.
   ▪ public policy – If file events are being audited, adds an event to the audit trail every time an 
auditable event happens to a public file. File classes include fa, fc, fd, fm, fr, fw, and cl. For 
the definition of a public file, see “Audit Terminology and Concepts” on page 554.
   ▪ path policy – Adds a path token to audit events that include an optional path token.
   ▪ group policy – Adds a group token to audit events that include an optional new groups 
token.
   ▪ seq policy – Adds a sequence token to every audit event.
   ▪ trail policy – Adds a trailer token to every audit event.
   ▪ windata_down policy – On a system that is configured with Trusted Extensions, adds events 
when information in a labeled window is downgraded.
   ▪ windata_up policy – On a system that is configured with Trusted Extensions, adds events 
when information in a labeled window is upgraded.
   ▪ zonename policy – Adds the zone name to every audit event. If the global zone is the only 
configured zone, adds zone, global to every audit event.
The following audit record shows the use of the `ls` command. The `ex` class is being audited and the default policy is in use:

```
header,375,2,execve(2),,mach1,2009-08-06 11:19:57.388 -07:00
path,/usr/bin/ls
subject,jdoe,root,root,root,root,1401,737,0 0 mach1
return,success,0
```

The following is the same record when all policies are turned on:

```
header,375,2,execve(2),,mach1,2009-08-06 11:19:57.388 -07:00
path,/usr/bin/ls
attribute,100555,root,bin,136,432,0
exec_args,1,ls
exec_env,9,HOME=/,HZ=,LANG=C,LOGNAME=root,MAIL=/var/mail/root,PATH=/usr/sbin:/usr/bin,SHELL=/sbin/sh,TERM=xterm,TZ=US/Pacific
path,/lib/ld.so.1
attribute,100755,root,bin,136,4289,0
subject,jdoe,root,root,root,root,1401,737,0 0 mach1
group,root,other,bin,sys,adm,uucp,mail/tty,lp,nuucp,daemon
return,success,0
zone,global
sequence,313540
trailor,375
```

2 Use the `audit_syslog.so` plugin to send some audit events to syslog.

This strategy works only if you are not required to keep binary records of the audit events that you send to the syslog logs. By using the `audit_reduce` command, you can then strip the binary files of these records, thus reducing the size of the binary files.

3 Use the `audit_user` file to audit events for specific users and roles.

Reduce the amount of auditing for all users by reducing the number of audit classes in the `audit_control` file. In the `audit_user` file, add audit classes for specific users and roles.

4 Create your own customized audit class.

You can create audit classes at your site. Into these classes, put all the audit events that you need to monitor. For the procedure, see "How to Add an Audit Class" on page 583.

Note – If you modify existing audit class assignments, your modifications might be lost when you upgrade to a newer version of the Solaris OS. Carefully review the install logs.
How to Audit All Commands by Users

As part of site security policy, some sites require audit records of all commands that are run by the root user or by administrative roles. Some sites also require audit records of all commands that are run by users.

1 Audit the lo and ex classes.

The ex class audits all calls to the exec() and execve() functions. The lo class audits logins, logouts, and screen locks. The following output lists all the events in the ex and lo classes.

7: AUE_EXEC:exec(2):ps,ex
23: AUE_EXECVE:execve(2):ps,ex
...
6152: AUE_login:login - local:lo
6153: AUE_logout:logout:lo
6154: AUE_telnet:login - telnet:lo
6155: AUE_rlogin:login - rlogin:lo
6158: AUE_rshd:rsh access:lo
6159: AUE_su:su:lo
6162: AUE_rexedc:rexedc:lo
6163: AUE_passwd:passwd:lo
6164: AUE_rexd:rexd:lo
6165: AUE_ftpd:ftp access:lo
6171: AUE_ftpd_logout:ftp logout:lo
6172: AUE_ssh:login - ssh:lo
6173: AUE_role:login:role login:lo
6212: AUE_newgrp_login:newgrp login:lo
6213: AUE_admin:authenticate:admin login:lo
6221: AUE_screenlock:screenlock - lock:lo
6222: AUE_screenunlock:screenlock - unlock:lo
6227: AUE_zlogin:login - zlogin:lo

To audit these classes for administrators, modify the audit_user file.

In the following example, the site has created three roles, sysadm, auditadm, and netadm. These roles and the root account are audited for the exec and lo classes:

```
## audit_user file
root:lo,ex:no
sysadm:lo,ex:no
auditadm:lo,ex:no
netadm:lo,ex:no
```

To audit the lo class for non-attributable events, modify the audit_control file.

```
## audit_control file
...  
naf:lo
...  
```
To audit these classes for all users, modify the `audit_control` file.

```
## audit_control file
flags:lo,ex
naflags:lo
...
```

The output appears similar to the following:

```
header,375,2,execve(2),,mach1,2009-08-06 11:19:57.388 -07:00
path,/usr/bin/ls
subject,jdoe,root,root,root,root,1401,737,0 0 mach1
return,success,0
```

2 To record the arguments to commands, set the `argv` policy.

```
# auditconfig -setpolicy +argv
```

The `exec_args` token records the command arguments:

```
header,375,2,execve(2),,mach1,2009-08-06 11:19:57.388 -07:00
path,/usr/bin/ls
exec_args,1,ls
subject,jdoe,root,root,root,root,1401,737,0 0 mach1
return,success,0
```

3 To record the environment in which the command is run, set the `arge` policy.

```
# auditconfig -setpolicy +arge
```

The `exec_env` token records the command environment:

```
header,375,2,execve(2),,mach1,2009-08-06 11:19:57.388 -07:00
path,/usr/bin/ls
exec_env,9,HOME=/,HZ=,LANG=C,LOGNAME=root,MAIL=/var/mail/root,
PATH=/usr/sbin:/usr/bin,SHELL=/sbin/sh,TERM=xterm,TZ=US/Pacific
subject,jdoe,root,root,root,root,1401,737,0 0 mach1
return,success,0
```

4 To record the arguments and the command environment, set both policies.

```
# auditconfig -setpolicy +argv
```

The output appears similar to the following:

```
header,375,2,execve(2),,mach1,2009-08-06 11:19:57.388 -07:00
path,/usr/bin/ls
exec_args,1,ls
exec_env,9,HOME=/,HZ=,LANG=C,LOGNAME=root,MAIL=/var/mail/root,
PATH=/usr/sbin:/usr/bin,SHELL=/sbin/sh,TERM=xterm,TZ=US/Pacific
subject,jdoe,root,root,root,root,1401,737,0 0 mach1
return,success,0
```
How to Find Audit Records of Changes to Specific Files

If your goal is to log file writes against a limited number of files, such as /etc/passwd and the files in the /etc/default directory, you use the auditreduce command to locate the files.

1. **Audit the fw class.**
   Adding the class to the audit_user file generates fewer records than adding the class to the audit_control file.
   - **Add the fw class to the audit_user file.**
     ```
     ## audit_user file
     root:fw:no
     sysadm:fw:no
     auditadm:fw:no
     netadm:fw:no
     ```
   - **Add the fw class to the audit_control file.**
     ```
     ## audit_control file
     flags:lo,fw
     ```

2. **To find the audit records for specific files, use the auditreduce command.**
   ```
   # /usr/sbin/auditreduce -o file=/etc/passwd,/etc/default -O filechg
   ```
   The auditreduce command searches the audit trail for all instances of the file argument. The command creates a binary file with the suffix filechg which contains all records that include the pathnames of the files of interest. See the `auditreduce(1M)` man page for the syntax of the -o file=pathname option.

3. **To read the filechg file, use the praudit command.**
   ```
   # /usr/sbin/praudit *filechg
   ```

How to Modify a User's Preselection Mask

If you modify the audit_control or audit_user file, the preselection mask of users who are already logged in does not change. You must force the preselection mask to change.

Before You Begin
You enabled auditing, users logged in, and then you changed the value of flags or naflags in the audit_control file. You want the users who are already logged in to be audited for these newly selected audit classes.
1 Update the preselection mask of users who are already logged in.
   You have two options. You can terminate the existing sessions or use the auditconfig command to update the users' preselection masks.

   - **Terminate the users' existing sessions.**
     Users can log out and log back in, or the administrator can manually terminate (kill) active sessions. The new sessions will inherit the new preselection mask. However, terminating users could be impractical.

   - **Dynamically change each user's preselection mask.**
     Assume that the flags attribute in the audit_control file was changed from lo to lo, ex.

     a. **Determine the user's audit ID and audit session ID.**
        First, find all regular users. In the following example, the administrator finds all processes that are not owned by root, daemon, or lp:

        ```
        # /usr/bin/pgrep -v -u root,daemon,lp | more
        ..
        3941
        3948
        3949
        10640 ...
        
        Then, use one of the user's processes to find the user's audit ID:
        ```

        ```
        # auditconfig -getpinfo 3941
        audit id = jdoe(1002)
        process preselection mask = lo(0x1000,0x1000)
        terminal id (maj,min,host) = 9426,65559,mach1(192.168.123.234)
        audit session id = 713
        ```

        Note that the user's preselection mask includes the lo class and does not include the newly added ex class.

        The user's audit ID is 1002. The user's audit session ID is 713.

2 **Change the user's preselection mask**
   Use one of the following two methods:

   - **Use the user's audit session ID to change the user's preselection mask.**
     ```
     # /usr/sbin/auditconfig -setsmask lo,ex 713
     ```

   - **Use the user's audit ID to change the user's preselection mask.**
     ```
     # /usr/sbin/auditconfig -setumask lo,ex 1002
     ```
3 Verify that the preselection mask has changed.

```bash
# auditconfig -getinfo 3941
audit id = jdoe(1002)
process preselection mask = ex,lo(0x40001000,0x40001000)
terminal id (maj,min,host) = 9426,65559,mach1(192.168.123.234)
audit session id = 713
```

▼ How to Prevent the Auditing of Certain Events

For maintenance purposes, sometimes a site wants to prevent audit events from being audited.

1 Change the class of the event to the no class.

For example, events 26 and 27 belong to the pm class.

```bash
## audit_event file
...
25:AUE_VFORK:vfork(2):ps
26:AUE_SETGROUPS:setgroups(2):pm
27:AUE_SETPGRP:setpgrp(2):pm
28:AUE_SWAPON:swapon(2):no
...
```

Change these events to the no class.

```bash
## audit_event file
...
25:AUE_VFORK:vfork(2):ps
26:AUE_SETGROUPS:setgroups(2):no
27:AUE_SETPGRP:setpgrp(2):no
28:AUE_SWAPON:swapon(2):no
...
```

If the pm class is currently being audited, existing sessions will still audit events 26 and 27. To stop these events from being audited, you must update the users’ preselection masks.

**Caution** – Never comment out events in the audit_event file. This file is used by the praudit command to read binary audit files. Archived audit files might contain events that are listed in the file.

2 To update the preselection masks of users, follow the instructions in “How to Modify a User’s Preselection Mask” on page 619.
How to Limit the Size of Binary Audit Files

Binary audit files grow without limit. For ease of archiving and searching, you might want to limit the size. You can also create smaller binary files from the original file.

1. Use the `p_fsize` attribute to limit the size of individual binary audit files.
   The `p_fsize` attribute to the `audit_binfile.so` plugin enables you to limit the size of an audit file. The default value is zero (0), which allows the file to grow without limit. The value is specified in bytes, from 512,000 to 2,147,483,647. When the specified size is reached, the current audit file is closed and a new file is opened.

   In the following example, you limit the size of an audit file to 1Mbyte:
   ```
   plugin:name=audit_binfile.so; p_dir:/var/audit; p_fsize=1024000
   ```

2. Use the `auditreduce` command to select records and write those records to a file for further analysis.
   The `auditreduce` `- lowercase` options find specific records.

   The `auditreduce` `- Uppercase` options write your selections to a file. For more information, see the `auditreduce(1M)` man page.

How to Audit Logins From Other OSes

The Solaris OS can audit all logins, independent of source.

Audit the `lo` class for attributable and for non-attributable events.
This class audits logins, logouts, and screen locks.

```
## audit_control file
flags:lo
naflags:lo
...
```

Note – To audit ssh logins, your Solaris system must be running the Solaris ssh daemon. This daemon is modified for Solaris auditing. For more information, see “Solaris Secure Shell and the OpenSSH Project” on page 324.

How to Audit FTP and SFTP File Transfers

The FTP service creates logs of its file transfers. The SFTP service, which runs under the SSH protocol, can be audited by Solaris auditing. Logins to both services can be audited by Solaris auditing.
1. To log commands and file transfers of the FTP service, see the `ftpaccess(4)` man page.
   For the available logging options, read the “Logging Capabilities” section. In particular, the `log` commands and `log transfers` options might provide useful logs.

2. To log sftp file transfers, perform one or both of the following:
   - **Audit file-reads.**
     File transfers over an SSH connection use the `sftp` command. These transfers can be recorded by using the `+fr` audit flag. To audit failed `sftp` file transfers, audit the `-fr` audit flag.
     The following output is from a successful `sftp` session:
     ```
     header,138,2,open(2) - read,,ma2,2009-08-25 14:48:58.770 -07:00
     path,/home/jdoe/vpn_connect
     attribute,100644,jdoe,staff,391,437,0
     subject,jdoe,jdoe,staff,jdoe,staff,4444,120289379,8457 65558 ma1
     return,success,6
     ```
   - **Use the verbose option to the `sftp` command.**
     The `-v` option can be repeated up to three times.
     ```
     # sftp -vvv [ other options ] hostname
     ```

3. To record access to the FTP and SFTP services, audit the `lo` class.
   As the following output indicates, logging in to and out of the `ftpd` daemon generates audit records.
   ```
   % bsmrecord -c lo | more
   ...
   in.ftpd
   program/usr/sbin/in.ftpd See ftp access
   event ID 6165 AUE_ftpd
   class lo (0x00001000)
   header
   subject
   [text] error message
   return
   ...
   ```
The SSH login records all accesses to the `sftp` command.

...  
/usr/lib/ssh/sshd  
program /usr/lib/ssh/sshd  See login - ssh  
event ID 6172  AUE_ssh  
class lo  (0x00001000)  
header  
subject  
[text]  error message  
return
This chapter describes the important components of Solaris auditing. The following is a list of the reference information in this chapter.

- “Solaris Audit Service” on page 625
- “Audit Commands” on page 626
- “Files Used in the Audit Service” on page 631
- “Rights Profiles for Administering Auditing” on page 636
- “Auditing and Solaris Zones” on page 637
- “Audit Classes” on page 637
- “Audit Plugins” on page 640
- “Audit Policy” on page 641
- “Process Audit Characteristics” on page 641
- “Audit Trail” on page 642
- “Conventions for Binary Audit File Names” on page 642
- “Audit Record Structure” on page 643
- “Audit Token Formats” on page 644

For an overview of Solaris auditing, see Chapter 28, “Solaris Auditing (Overview).” For planning suggestions, see Chapter 29, “Planning for Solaris Auditing.” For procedures to configure auditing at your site, see Chapter 30, “Managing Solaris Auditing (Tasks).”

**Solaris Audit Service**

The Solaris audit service, auditd, is configured by using the following auditconfig subcommands:

- The -setpolicy subcommand modifies the audit policies.
- The -setqbufsz, -setqctrl, -setqdelay, -setqhiwater, and -setqlowater subcommands set audit queue parameters.
The `auditconfig -t subcommand` command creates a temporary, or active, value. Without the `-t` option, the values are persistent, or configured. For more information, see the `auditconfig(1M)` man page.

The `bsmconv` command enables the audit service initially. The service is stopped and restarted by using the `audit -t` and `audit -s` commands, respectively. The `bsmunconv` command disables the service permanently.

For more information, see the `audit(1M), auditconfig(1M), and bsmconv(1M)` man pages.

Audit Commands

This section provides information about the following commands:

- “auditd Daemon” on page 626
- “audit Command” on page 627
- “auditrecord Command” on page 627
- “auditreduce Command” on page 628
- “praudit Command” on page 629
- “auditconfig Command” on page 631

auditd Daemon

The following list summarizes the tasks of the auditd daemon:

- Configures audit policy.
- Configures the audit queue parameters.
- Configures the event-to-class mappings.
- Sets the non-attributable audit mask. For more information, see the `-aconf` option to the `auditconfig` command.
- Opens and closes audit files in the directories that are specified in the `audit_control` file. The files are opened in order of mention.
- Loads one or more plugins. Sun provides three plugins.
  - The `audit_binfile.so` plugin writes binary audit data to a file.
  - The `audit_remote.so` plugin sends binary audit data to an authenticated server with privacy and integrity protection.
  - The `audit_syslog.so` plugin delivers selected text summaries of audit records to the `syslog` log.
- Reads audit data from the kernel and outputs the data by using an `auditd` plugin.
Executes the audit_warn script to warn of various conditions. The binfile.so plugin executes the audit_warn script. The script, by default, sends warnings to the audit_warn email alias and to the console. The syslog.so plugin does not execute the audit_warn script.

By default, when all audit directories are full, processes that generate audit records are suspended. In addition, the auditd daemon writes a message to the console and to the audit_warn email alias. At this point, only the system administrator can fix the audit service. The administrator can log in to write audit files to offline media, delete audit files from the system, and do other cleanup tasks.

The audit policy can be reconfigured with the auditconfig command.

The auditd daemon can be started automatically when the system is booted into multiuser mode. Or, you can start the daemon from the command line by running the audit -s command after auditing has been enabled with the bsmconv command. When the auditd daemon is started, it calculates the amount of free space that is necessary for audit files.

The auditd daemon uses the list of audit directories in the audit_control file as possible locations for creating audit files. The daemon maintains a pointer into this list of directories, starting with the first directory. Every time the auditd daemon needs to create an audit file, the daemon puts the file into the first available directory in the list. The list starts at the auditd daemon's current pointer. You can reset the pointer to the beginning of the list by running the audit -s command. The audit -n command instructs the daemon to switch to a new audit file. The new file is created in the same directory as the current file.

**audit Command**

The audit command controls the actions of the auditd daemon. The audit command can do the following tasks:

- Enable and disable auditing
- Reset the auditd daemon
- Adjust the auditing preselection mask on the local system
- Write audit records to a different audit file

For a discussion of the available options, see the audit(1M) man page.

**auditrecord Command**

The auditrecord command displays the format of audit events that are defined in the /etc/security/audit_event file. The output includes the event's audit ID, audit class, audit flag, and the record's audit tokens in order. With no option, the auditrecord output displays in a terminal window. With the -h option, the output is suitable for viewing in a browser. For examples of the use of the auditrecord command, see "How to Display Audit Record Formats" on page 602. Also, see the auditrecord(1M) man page.
The auditreduce command summarizes audit records that are stored in binary format. The command can merge audit records from one or more input audit files. The command can also be used to perform a post selection of audit records. The records remain in binary format. To merge the entire audit trail, run this command on the audit server. The audit server is the system that mounts all the audit file systems for the installation. For more information, see the auditreduce(1M) man page.

The auditreduce command enables you to track all audited actions on multiple systems from a single location. The command can read the logical combination of all audit files as a single audit trail. You must identically configure all systems at a site for auditing, and create servers and local directories for the audit files. The auditreduce command ignores how the records were generated or where the records are stored. Without options, the auditreduce command merges audit records from all the audit files in all of the subdirectories in the audit root directory. Typically, /etc/security/audit is the audit root directory. The auditreduce command sends the merged results to standard output. You can also place the results into a single, chronologically ordered output file. The file contains binary data.

The auditreduce command can also select particular types of records for analysis. The merging functions and selecting functions of the auditreduce command are logically independent. The auditreduce command captures data from the input files as the records are read, before the files are merged and then written to disk.

By specifying options to the auditreduce command, you can also do the following:

- Request audit records that were generated by specified audit classes
- Request audit records that were generated by one particular user
- Request audit records that were generated on specific dates

With no arguments, the auditreduce command checks the subdirectories within the /etc/security/audit directory, the default audit root directory. The command checks for a files directory in which the start-time.end-time.hostname files reside. The auditreduce command is very useful when audit data resides in separate directories. Figure 31–1 illustrates audit data in separate directories for different hosts. Figure 31–2 illustrates audit data in separate directories for different audit servers.
If the partition for the /etc/security/audit directory is very small, you might not store audit data in the default directory. You can pass the auditreduce command another directory by using the -R option:

```
# auditreduce -R /var/audit-alt
```

You can also specify a particular subdirectory by using the -S option:

```
# auditreduce -S /var/audit-alt/host1
```

For other options and more examples, see the auditreduce(1M) man page.

**praudit Command**

The praudit command makes the binary output of the auditreduce command readable. The praudit command reads audit records in binary format from standard input and displays the records in a presentable format. The input can be piped from the auditreduce command or
from a single audit file. Input can also be produced with the cat command to concatenate several files, or the tail command for a current audit file.

The praudit command can generate four output formats. A fifth option, -l (long), prints one audit record per line of output. The default is to place one audit token per line of output. The -d option changes the delimiter that is used between token fields and between tokens. The default delimiter is a comma.

- **Default** – The praudit command with no options displays one audit token per line. The command displays the audit event by its description, such as the ioctl(2) system call. Any value that can be displayed as text is displayed in text format. For example, a user is displayed as the user name, not as the user ID.

- **-r option** – The raw option displays as a number any value that could be numeric. For example, a user is displayed by user ID, Internet addresses are in hexadecimal format, and modes are in octal format. The audit event is displayed as its event number, such as 158.

- **-s option** – The short option displays the audit event by its table name, for example, AUE_IOCTL. The option displays the other tokens as the default option displays them.

- **-x option** – The XML option displays the audit record in XML format. This option is useful as input to browsers, or as input to scripts that manipulate XML.

   The XML is described by a DTD that the audit service provides. Solaris software also provides a style sheet. The DTD and the style sheet are in the /usr/share/lib/xml directory.

   In the default output format of the praudit command, each record is easily identified as a sequence of audit tokens. Each token is presented on a separate line. Each record begins with a header token. You could, for example, further process the output with the awk command.

   Here is the output from the praudit -l command for a header token:

   ```
   header,173,2,settppriv(2),,example1,2003-10-13 13:46:02.174 -07:00
   ```

   Here is the output from the praudit -r command for the same header token:

   ```
   121,173,2,289,0x0000,192.168.86.166,1066077962,174352445
   ```

**EXAMPLE 31–1**  Processing praudit Output With a Script

You might want to process output from the praudit command as lines of text. For example, you might want to select records that the auditreduce command cannot select. You can use a simple shell script to process the output of the praudit command. The following simple example script puts one audit record on one line, searches for a user-specified string, then returns the audit file to its original form.

```bash
#!/bin/sh
#
## This script takes an argument of a user-specified string.
```
EXAMPLE 31–1  Processing praudit Output With a Script  (Continued)

# The sed command prefixes the header tokens with Control-A
# The first tr command puts the audit tokens for one record
# onto one line while preserving the line breaks as Control-A
#
praudit | sed -e '1,2d' -e 's/file. */' -e 's/^header/\^aheader/' \  
| tr '\012\001' '\002\012' \  
| grep "$1" \  Finds the user-specified string
| tr '\002' '\012' \  Restores the original newline breaks

Note that the \^a in the script is Control-A, not the two characters ^ and a. The prefix
distinguishes the header token from the string header that might appear as text.

**auditconfig Command**

The auditconfig command provides a command-line interface to retrieve and set audit
configuration parameters. The auditconfig command can do the following tasks:

- Display, check, and configure audit policy
- Determine if auditing is turned on or turned off
- Turn auditing off and turn auditing on
- Manage the audit directory and the audit file
- Manage the audit queue
- Get and set preselection masks
- Get and set audit event to audit class mappings
- Get and set configuration information, such as session ID and audit ID
- Configure audit characteristics for a process, a shell, and a session
- Reset audit statistics

For a discussion of the command options, see the auditconfig(1M) man page.

**Files Used in the Audit Service**

The audit service uses the following files:

- “system File” on page 632
- “syslog.conf File” on page 632
- “audit_class File” on page 632
- “audit_control File” on page 632
- “audit_event File” on page 634
- “audit_user Database” on page 634
- “audit_warn Script” on page 635
- “bsmconv Script” on page 636
system File

The /etc/system file contains commands that the kernel reads during initialization to customize the system operations. The bsmconv and bsmunconv shell scripts, which are used to activate and deactivate auditing, modify the /etc/system file. The bsmconv shell script adds the following line to the /etc/system file:

```
set c2audit:audit_load=1
```

The set c2audit:audit_load=1 entry causes the kernel module for auditing to be loaded when the system is booted. The bsmunconv shell script disables auditing when the system is rebooted. The command removes the c2audit line from the /etc/system file.

syslog.conf File

The /etc/syslog.conf file works with the audit_control file to store audit records in text format. The syslog.conf file can be configured to enable the syslog utility to store audit records. For an example, see “How to Configure syslog Audit Logs” on page 579.

audit_class File

The /etc/security/audit_class file defines the audit classes. Audit classes are groups of audit events. You use the class name in the audit_control file to preselect the classes whose events you want to audit. The classes accept prefixes to select only failed events or only successful events. For more information, see “Audit Class Syntax” on page 639.

The superuser, or an administrator in an equivalent role, can modify the definitions of audit classes. This administrator can define new audit classes, rename existing classes, or otherwise change existing classes by editing the audit_class file in a text editor. For more information, see the audit_class(4) man page.

audit_control File

The /etc/security/audit_control file on each system contains configuration information for the auditd daemon. The file enables every system to mount a remote audit file system to store their audit records.

You can specify five kinds of information in the audit_control file. Each line of information begins with a keyword.

- flags keyword – Begins the entry that preselects which classes of events are audited for all users on the system. The audit classes that are specified here determine the system-wide audit preselection mask. The audit classes are separated by commas.
### naflags keyword
- Begins the entry that preselects which classes of events are audited when an action cannot be attributed to a specific user. The audit classes are separated by commas.

The na event class belongs in this entry. The naflags entry can be used to log other event classes that are normally attributable but cannot be attributed. For example, if a program that starts at boot reads a file, then an `fr` in the naflags entry would create a record for that event.

### minfree keyword
- This keyword is deprecated. Use the `p_minfree` attribute to the `audit_binfile.so` plugin.

The `p_minfree` attribute defines the minimum free-space level for all audit file systems as a percentage. The percentage must be equal to 0 or greater than 0. The default is 20 percent. When an audit file system is 80 percent full, the audit data is then stored in the next available audit directory. For more information, see the `audit_warn(1M)` man page.

### dir keyword
- This keyword is deprecated. Use the `p_dir` attribute to the `audit_binfile.so` plugin.

The `p_dir` attribute lists the directory locations. Each line value defines an audit file system and directory that the system uses to store its audit files. You can specify one or more directory locations. The order of the values is significant. The `auditd` daemon creates audit files in the directories in the specified order. The first directory is the *primary audit directory* for the system. The second directory is the *secondary audit directory* where the `auditd` daemon creates audit files when the first directory becomes full, and so on. For more information, see the `audit(1M)` man page.

### plugin keyword
- Specifies the plugin path for the plugin modules `audit_binfile.so` and `audit_syslog.so`. The `audit_binfile.so` module handles the creation of binary audit files. The `audit_syslog.so` module provides real-time conversion of Solaris audit records to text. The audit classes that are specified in the `p_flags` attribute of the `audit_syslog.so` plugin must be a subset of the preselected audit classes.

For more information about the `audit_control` file, see the `audit_control(4)` man page. For information about the plugins, see "Audit Plugins" on page 640 and the `audit_binfile(5)` and `audit_syslog(5)` man pages.

#### EXAMPLE 31-2  Sample `audit_control` File

The following is a sample `audit_control` file for the system noddy. `noddy` uses two audit file systems on the audit server `blinken`, and a third audit file system that is mounted from the second audit server `winken`. The third file system is used only when the audit file systems on `blinken` become full or unavailable. The `p_minfree` value of 20 percent specifies that the warning script is run when the file systems are 80 percent full. The settings specify that logins and administrative operations are to be audited. The operations are audited for success and for failure. Failures of all types, except failures to create a file system object, are to be audited. Nonattributable events are also audited. The `syslog` audit log records fewer audit events. This log contains text summaries of failed logins and failed administrative operations.

In the following example, the `plugin` lines do not contain a line break.
EXAMPLE 31–2  Sample audit_control File  (Continued)
flags:lo,am,-all,^-fc
naflags:lo,nt
plugin:name=audit_binfile.so; p_minfree=20; p_dir=/var/audit/blinken/files,
/var/audit/blinken.1/files,/var/audit/winken
plugin:name=audit_syslog.so; p_flags=-lo,-am

audit_event File

The /etc/security/audit_event file contains the default audit event-class mappings. You can
edit this file to change the class mappings. When you change class mappings, you must reboot
the system or run the auditconfig -conf command to read the changed mappings into the
kernel. For more information, see the audit_event(4) man page.

audit_user Database

The /etc/security/audit_user database modifies the system-wide preselected classes for an
individual user. The classes that you add to a user’s entry in the audit_user database modify the
settings in the audit_control file in two ways:

- By specifying audit classes that are always to be audited for this user
- By specifying audit classes that are never to be audited for this user

Each user entry in the audit_user database contains three fields:

username:always-audit-classes:never-audit-classes

The audit fields are processed in sequence.

- The always-audit-classes field turns on the auditing of the classes in that field. Use this field
to modify system-wide settings. For example, putting all in the always-audit-classes field
audits everything for a user.
- The never-audit-classes field turns off the auditing of the classes in that field. Use this field to
override system settings. Putting all in the never-audit-classes field turns off all auditing for
the user, even the audit classes that are specified in the audit_control file.

Suppose that you want to apply the system-wide audit settings to the user tamiko, except for
successful reads of file system objects. Note the placement of the second colon (:) in the
following audit_user entry:

tamiko:^+fr:no  modify system defaults for fr

The preceding entry means, “always audit everything, except for successful file reads.”
If you want to audit everything for user tamiko with the exception of successful file reads, you use the following entry:

```
tamiko:all,^+fr:no audit everything except fr
```

Suppose that you want to override system defaults for successful file-reads for user tamiko. The following entry means “always audit everything, but never audit successful file reads.”

```
tamiko:all:+fr override system defaults for fr
```

**Note** – Successful events and failed events are treated separately. A process could generate more audit records for failed events than for successful events.

### `audit_warn` Script

The `/etc/security/audit_warn` script notifies an email alias when the auditd daemon encounters an unusual condition while writing audit records. You can customize this script for your site to warn of conditions that might require manual intervention. Or, you could specify how to handle those conditions automatically. For all error conditions, the `audit_warn` script writes a message to syslog with the severity of `daemon.alert`. You can use `syslog.conf` to configure console display of syslog messages. The `audit_warn` script also sends a message to the `audit_warn` email alias. You set up this alias as part of audit configuration.

When the `auditd` daemon detects the following conditions, the daemon invokes the `audit_warn` script. The script sends email to the `audit_warn` alias.

- An audit directory has become more full than the `p_minfree` value allows. The `p_minfree` value or soft limit is a percentage of the available space on an audit file system.
  
  The `audit_warn` script is invoked with the string `soft` and the name of the directory whose available space is below the minimum value. The `auditd` daemon switches automatically to the next suitable directory. The daemon writes the audit files in this new directory until the directory reaches its `minfree` limit. The `auditd` daemon then goes to each remaining directory in the order that is listed in the `audit_control` file. The daemon writes audit records until each directory is at its `minfree` limit.

- All the audit directories have reached the `minfree` threshold.
  
  The `audit_warn` script is invoked with the string `allsoft` A message is written to the console. Email is also sent to the `audit_warn` alias.

  When all audit directories that are listed in the `audit_control` file have reached their `minfree` threshold, the `auditd` daemon switches back to the first directory. The daemon writes audit records until the directory becomes completely full.

- An audit directory has become completely full with no space remaining.
The `audit_warn` script is invoked with the string `hard` and the name of the directory. A message is written to the console. Email is also sent to the `audit_warn` alias.

The `auditd` daemon switches automatically to the next suitable directory with any space available. The `auditd` daemon goes to each remaining directory in the order that is listed in the `audit_control` file. The daemon writes audit records until each directory is full.

- All the audit directories are completely full. The `audit_warn` script is invoked with the string `allhard` as an argument.

By default, a message is written to the console. Email is also sent to the `audit_warn` alias. Processes that would otherwise generate audit records continue to occur, but audit records are counted. Audit records are not generated. For an example of how to handle this situation, see Example 30–13 and "How to Prevent Audit Trail Overflow" on page 611.

- An internal error occurs. Possible internal errors include the following:
  - `ebusy` – Another `auditd` daemon process is already running
  - `tmpfile` – A temporary file cannot be used
  - `postsigterm` – A signal was received during auditing shutdown
  - `plugin name` – An error occurred during execution of the plugin

  A problem is discovered with the syntax of the `audit_control` file. By default, a message is sent to the console. Email is also sent to the `audit_warn` alias.

If the `perzone` audit policy is set, the non-global zone's instance of `auditd` calls the zone's `audit_warn` script. For further information, see the `audit_warn(1M)` man page.

### bsmconv Script

The `/etc/security/bsmconv` script enables the audit service. The `bsmunconv` command disables the audit service. After the `bsmconv` script is run, you configure the audit directories and audit configuration files. Upon reboot, auditing is enabled.

For further information, see the `bsmconv(1M)` man page.

### Rights Profiles for Administering Auditing

The Solaris OS provides rights profiles for configuring the audit service and for analyzing the audit trail.

- **Audit Control** – Enables a role to configure Solaris auditing. This rights profile grants authorizations to configure files that are used by the audit service. The profile also enables a role to run audit commands. A role with the Audit Control profile can run the following commands: `audit`, `auditd`, `auditconfig`, `bsmconv`, and `bsmunconv`.

- **Audit Review** – Enables a role to analyze Solaris audit records. This rights profile grants authorization to read audit records with the `praudit` and `auditreduce` commands. A role with this rights profile can also run the `auditstat` command.
**System Administrator** – Includes the Audit Review rights profile. A role with the System Administrator rights profile can analyze audit records.

To configure roles to handle the audit service, see “Configuring RBAC (Task Map)” on page 174.

### Auditing and Solaris Zones

Non-global zones can be audited exactly as the global zone is audited, or nonglobal zones can set their own flags, storage, and audit policy.

When all zones are being audited identically, the configuration files in the global zone provide the settings for auditing in every zone. The `+zonename` policy option is useful. When this option is set, the audit records from all zones include the name of the zone. Audit records can then be postselected by zone name. To understand audit policy, see “Determining Audit Policy” on page 568. For an example, see “How to Configure Audit Policy” on page 590.

Zones can also be audited individually. When the policy option, `perzone`, is set in the global zone, each non-global zone runs its own audit daemon, handles its own audit queue, and specifies the content and location of its audit records. A non-global zone can also set most audit policy options. It cannot set policy that affects the entire system, so a non-global zone cannot set the `ahlt` or `perzone` policy. For further discussion, see “Auditing on a System With Zones” on page 560 and “How to Plan Auditing in Zones” on page 564.

To learn about zones, see Part II, “Zones,” in *System Administration Guide: Virtualization Using the Solaris Operating System*.

### Audit Classes

System-wide defaults for Solaris auditing are preselected by specifying one or more classes of events. The classes are preselected for each system in the system’s `audit_control` file. Anyone who uses the system is audited for these classes of events. The file is described in “audit_control File” on page 632.

You can configure audit classes and make new audit classes. Audit class names can be up to 8 characters in length. The class description is limited to 72 characters. Numeric and non-alphanumeric characters are allowed.

You can modify what is audited for individual users by adding audit classes to a user’s entry in the `audit_user` database. The audit classes are also used as arguments to the `auditconfig` command. For details, see the `auditconfig(1M)` man page.
Definitions of Audit Classes

The following table shows each predefined audit class, the descriptive name for each audit class, and a short description.

<table>
<thead>
<tr>
<th>Audit Class</th>
<th>Descriptive Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>all</td>
<td>All classes (metaclass)</td>
</tr>
<tr>
<td>no</td>
<td>no_class</td>
<td>Null value for turning off event preselection</td>
</tr>
<tr>
<td>na</td>
<td>non_attrib</td>
<td>Nonattributable events</td>
</tr>
<tr>
<td>fr</td>
<td>file_read</td>
<td>Read of data, open for reading</td>
</tr>
<tr>
<td>fw</td>
<td>file_write</td>
<td>Write of data, open for writing</td>
</tr>
<tr>
<td>fa</td>
<td>file_attr_acc</td>
<td>Access of object attributes: stat, pathconf</td>
</tr>
<tr>
<td>fm</td>
<td>file_attr_mod</td>
<td>Change of object attributes: chown, flock</td>
</tr>
<tr>
<td>fc</td>
<td>file_creation</td>
<td>Creation of object</td>
</tr>
<tr>
<td>fd</td>
<td>file_deletion</td>
<td>Deletion of object</td>
</tr>
<tr>
<td>cl</td>
<td>file_close</td>
<td>close system call</td>
</tr>
<tr>
<td>ap</td>
<td>application</td>
<td>Application-defined event</td>
</tr>
<tr>
<td>ad</td>
<td>administrative</td>
<td>Administrative actions (old administrative metaclass)</td>
</tr>
<tr>
<td>am</td>
<td>administrative</td>
<td>Administrative actions (metaclass)</td>
</tr>
<tr>
<td>ss</td>
<td>system state</td>
<td>Change system state</td>
</tr>
<tr>
<td>as</td>
<td>system-wide</td>
<td>System-wide administration</td>
</tr>
<tr>
<td>ua</td>
<td>user administration</td>
<td>User administration</td>
</tr>
<tr>
<td>aa</td>
<td>audit administration</td>
<td>Audit utilization</td>
</tr>
<tr>
<td>ps</td>
<td>process start</td>
<td>Process start and process stop</td>
</tr>
<tr>
<td>pm</td>
<td>process modify</td>
<td>Process modify</td>
</tr>
<tr>
<td>pc</td>
<td>process</td>
<td>Process (metaclass)</td>
</tr>
<tr>
<td>ex</td>
<td>exec</td>
<td>Program execution</td>
</tr>
<tr>
<td>io</td>
<td>ioctl</td>
<td>ioctl() system call</td>
</tr>
<tr>
<td>ip</td>
<td>ipc</td>
<td>System V IPC operations</td>
</tr>
</tbody>
</table>
### Audit Classes

#### TABLE 31–1 Predefined Audit Classes (Continued)

<table>
<thead>
<tr>
<th>Audit Class</th>
<th>Descriptive Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lo</td>
<td>login_logout</td>
<td>Login and logout events</td>
</tr>
<tr>
<td>nt</td>
<td>network</td>
<td>Network events: bind, connect, accept</td>
</tr>
<tr>
<td>ot</td>
<td>other</td>
<td>Miscellaneous, such as device allocation and memcntl()</td>
</tr>
</tbody>
</table>

You can define new classes by modifying the `/etc/security/audit_class` file. You can also rename existing classes. For more information, see the `audit_class(4)` man page.

### Audit Class Syntax

Events can be audited for success, events can be audited for failure, and events can be audited for both. Without a prefix, a class of events is audited for success and for failure. With a plus (+) prefix, a class of events is audited for success only. With a minus (-) prefix, a class of events is audited for failure only. The following table shows some possible representations of audit classes.

#### TABLE 31–2 Plus and Minus Prefixes to Audit Classes

<table>
<thead>
<tr>
<th>[prefix]class</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>lo</td>
<td>Audit all successful attempts to log in and log out, and all failed attempts to log in. A user cannot fail an attempt to log out.</td>
</tr>
<tr>
<td>+lo</td>
<td>Audit all successful attempts to log in and log out.</td>
</tr>
<tr>
<td>-all</td>
<td>Audit all failed events.</td>
</tr>
<tr>
<td>+all</td>
<td>Audit all successful events.</td>
</tr>
</tbody>
</table>

**Caution** – The `all` class can generate large amounts of data and quickly fill audit file systems. Use the `all` class only if you have extraordinary reasons to audit all activities.

Audit classes that were previously selected can be further modified by a caret prefix, `^`. The following table shows how the caret prefix modifies a preselected audit class.

#### TABLE 31–3 Caret Prefix That Modifies Already-Specified Audit Classes

<table>
<thead>
<tr>
<th><code>^[prefix]class</code></th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>-all,^-fc</td>
<td>Audit all failed events, except do not audit failed attempts to create file objects</td>
</tr>
</tbody>
</table>
### TABLE 31-3  Caret Prefix That Modifies Already-Specified Audit Classes  *(Continued)*

<table>
<thead>
<tr>
<th>^[prefix]class</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>am,^+aa</td>
<td>Audit all administrative events for success and for failure, except do not audit successful attempts to administer auditing</td>
</tr>
<tr>
<td>am,^ua</td>
<td>Audit all administrative events for success and for failure, except do not audit user administration events</td>
</tr>
</tbody>
</table>

The audit classes and their prefixes can be used in the following files and commands:

- In the `flags` line in the `audit_control` file
- In the `plugin:name=audit_syslog.so; p_flags=` line in the `audit_control` file
- In the user's entry in the `audit_user` database
- As arguments to `auditconfig` command options

See “audit_control File” on page 632 for an example of using the prefixes in the `audit_control` file.

### Audit Plugins

Audit plugins specify how to handle the audit records in the audit queue. The audit plugins are specified by name in the `audit_control` file: `audit_binfile.so`, `audit_remote.so`, and `audit_syslog.so`. The plugins and their parameters can specify the following:

- Where to send binary data, by using the `audit_binfile.so` plugin, `p_dir` parameter
- The minimum space remaining on a disk before the administrator is warned, by using the `audit_binfile.so` plugin, `p_minfree` parameter
- The maximum size of an audit file, by using the `audit_binfile.so` plugin, `p_fsize` parameter
- A selection of audit records to be sent to `syslog`, by using the `audit_syslog.so` plugin, `p_flags` parameter
- The maximum number of audit records that are queued for the plugin, by using the `qsize` parameter
- A remote authenticated audit server to send the binary audit data to, by using the `audit_remote` plugin

Refer to the `audit_binfile(5)`, `audit_remote(5)`, `audit_syslog(5)`, and `audit_control(4)` man pages.
Audit Policy

Audit policy determines if additional information is added to the audit trail.

The following policies add tokens to audit records: arge, argv, group, path, seq, trail, windata_down, windata_up, and zonename.

The remaining policies do not add tokens. The ahlt and cnt policies determine what happens when kernel audit records cannot be delivered, the public policy limits auditing of public files, and the perzone policy establishes separate audit queues for non-global zones.

The effects of the different audit policy options are described in “Determining Audit Policy” on page 568. For a description of audit policy options, see the -setpolicy option in the auditconfig(1M) man page. For a list of available policy options, run the auditconfig -lspolicy command.

Process Audit Characteristics

The following audit characteristics are set at initial login:

- **Process preselection mask** – A combination of the audit classes from the audit_control file and the audit_user database. When a user logs in, the login process combines the preselected classes to establish the process preselection mask for the user’s processes. The process preselection mask specifies whether events in each audit class are to generate audit records.

  The following algorithm describes how the system obtains the user’s process preselection mask:

  \[(flags \text{ line} + always-audit-classes) - never-audit-classes\]

  Add the audit classes from the flags line in the audit_control file to the classes from the always-audit-classes field in the user’s entry in the audit_user database. Then, subtract from the total the classes from the user’s never-audit-classes field.

- **Audit ID** – A process acquires an audit ID when the user logs in. The audit ID is inherited by all child processes that were started by the user’s initial process. The audit ID helps enforce accountability. Even after a user becomes root, the audit ID remains the same. The audit ID that is saved in each audit record always allows you to trace actions back to the original user who had logged in.

- **Audit Session ID** – The audit session ID is assigned at login. The session ID is inherited by all child processes.

- **Terminal ID (port ID, machine ID)** – The terminal ID consists of the host name and the Internet address, followed by a unique number that identifies the physical device on which the user logged in. Most often, the login is through the console. The number that corresponds to the console device is 0.
Audit Trail

The audit trail contains binary audit files. The trail is created by the auditd daemon. Once the audit service has been enabled with the `bsmconv` command, the auditd daemon starts when the system is booted. The auditd daemon is responsible for collecting the audit trail data and writing the audit records.

The audit records are stored in binary format on file systems that are dedicated to audit files. Even though you can physically locate audit directories within file systems that are not dedicated to auditing, do not do so except for directories of last resort. Directories of last resort are directories where audit files are written only when no other suitable directory is available.

There is one other scenario where locating audit directories outside of dedicated audit file systems could be acceptable. You might do so in a software development environment where auditing is optional. To make full use of disk space might be more important than to keep an audit trail. However, in a security-conscious environment, the placement of audit directories within other file systems is not acceptable.

You should also consider the following factors when administering audit file systems:

- A host should have at least one local audit directory. The local directory can be used as a directory of last resort if the host is unable to communicate with the audit server.
- Mount audit directories with the read-write (`rw`) option. When you mount audit directories remotely, also use the `int r` and `noac` options.
- List the audit file systems on the audit server where they reside. The export list should include all systems that are being audited at the site.

Conventions for Binary Audit File Names

Each binary audit file is a self-contained collection of records. The file's name identifies the time span during which the records were generated and the system that generated them.

Binary Audit File Names

Audit files that are complete have names of the following form:

```
start-time.end-time.system
```

- `start-time` is the time that the first audit record in the audit file was generated
- `end-time` is the time that the last record was written to the file
- `system` is the name of the system that generated the file

An audit file that is still active has a name of the following form:
start-time, not_terminated, system

For examples of not_terminated and closed audit file names, see “How to Clean Up a not_terminated Audit File” on page 610.

Binary Audit File Timestamps

The timestamps in file names are used by the audit reduce command to locate records within a specific time range. These timestamps are important because there can be a month’s accumulation or more of audit files online. To search all the files for records that were generated in the last 24 hours would be unacceptably expensive.

The start-time and end-time are timestamps with one-second resolution. They are specified in Greenwich Mean Time (GMT). The format is four digits for the year, followed by two digits for each month, day, hour, minute, and second, as follows:

YYYYMMDDHHMMSS

The timestamps are in GMT to ensure that they sort in proper order, even across time zones. Because they are in GMT, the date and hour must be translated to the current time zone to be meaningful. Beware of this point whenever you manipulate these files with standard file commands rather than with the audit reduce command.

Audit Record Structure

An audit record is a sequence of audit tokens. Each audit token contains event information such as user ID, time, and date. A header token begins an audit record, and an optional trailer token concludes the record. Other audit tokens contain information relevant to the audit event. The following figure shows a typical audit record.

```
header token
arg token
data token
subject token
return token
```

FIGURE 31-3 Typical Audit Record Structure
Audit Record Analysis

Audit record analysis involves postselecting records from the audit trail. You can use one of two approaches to parsing the binary data that was collected.

- You can parse the binary data stream. To parse the data stream, you need to know the order of the fields in each token, and the order of tokens in each record. You also need to know the variants of an audit record. For example, the `ioctl()` system call creates an audit record for “Bad file name” that contains different tokens from the audit record for “Invalid file descriptor”.
  - For a description of the order of binary data in each audit token, see the `audit.log(4)` man page.
  - For a description of the order of tokens in an audit record, use the `auditrecord` command. Output from the `auditrecord` command includes the different formats that occur under different conditions. Square brackets ([ ]) indicate that an audit token is optional. For more information, see the `auditrecord(1M)` man page. For examples, see also “How to Display Audit Record Formats” on page 602.
  - You can use the `praudit` command. Options to the command provide different text outputs. For example, the `praudit -x` command provides XML for input to scripts and browsers. `praudit` outputs do not include fields whose sole purpose is to help to parse the binary data. The outputs do not necessarily follow the order of the binary fields. Also, the order and format of `praudit` output is not guaranteed between Solaris releases.
    
    For examples of `praudit` output, see “How to View the Contents of Binary Audit Files” on page 608, and the `praudit(1M)` man page.
    
    For a description of the `praudit` output for each audit token, see the individual tokens in the “Audit Token Formats” on page 644 section.

Audit Token Formats

Each audit token has a token type identifier, which is followed by data that is specific to the token. Each token type has its own format. The following table shows the token names with a brief description of each token. Obsolete tokens are maintained for compatibility with previous Solaris releases.

<table>
<thead>
<tr>
<th>Token Name</th>
<th>Description</th>
<th>For More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>acl</td>
<td>Access Control List (ACL) information</td>
<td>“acl Token” on page 646</td>
</tr>
<tr>
<td>arbitrary</td>
<td>Data with format and type information</td>
<td>See the <code>audit.log(4)</code> man page.</td>
</tr>
<tr>
<td>Token Name</td>
<td>Description</td>
<td>For More Information</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>arg</td>
<td>System call argument value</td>
<td>“arg Token” on page 646</td>
</tr>
<tr>
<td>attribute</td>
<td>File vnode tokens</td>
<td>“attribute Token” on page 646</td>
</tr>
<tr>
<td>cmd</td>
<td>Command arguments and environment variables</td>
<td>“cmd Token” on page 647</td>
</tr>
<tr>
<td>exec_args</td>
<td>Exec system call arguments</td>
<td>“exec_args Token” on page 647</td>
</tr>
<tr>
<td>exec_env</td>
<td>Exec system call environment variables</td>
<td>“exec_env Token” on page 647</td>
</tr>
<tr>
<td>exit</td>
<td>Program exit information</td>
<td>See the audit.log(4) man page.</td>
</tr>
<tr>
<td>file</td>
<td>Audit file information</td>
<td>“file Token” on page 648</td>
</tr>
<tr>
<td>fmri</td>
<td>Framework Management Resource Indicator token</td>
<td>“fmri Token” on page 648</td>
</tr>
<tr>
<td>group</td>
<td>Process groups information</td>
<td>See “groups Token” on page 649.</td>
</tr>
<tr>
<td>groups</td>
<td>Process groups information</td>
<td>“groups Token” on page 649</td>
</tr>
<tr>
<td>header</td>
<td>Indicates start of audit record</td>
<td>“header Token” on page 649</td>
</tr>
<tr>
<td>in_addr</td>
<td>Internet address</td>
<td>“in_addr Token” on page 649</td>
</tr>
<tr>
<td>ip</td>
<td>IP header information</td>
<td>See the audit.log(4) man page.</td>
</tr>
<tr>
<td>ipc</td>
<td>System V IPC information</td>
<td>“ipc Token” on page 649</td>
</tr>
<tr>
<td>ipc_perm</td>
<td>System V IPC object tokens</td>
<td>“ipc_perm Token” on page 650</td>
</tr>
<tr>
<td>iport</td>
<td>Internet port address</td>
<td>“iport Token” on page 650</td>
</tr>
<tr>
<td>opaque</td>
<td>Unstructured data (unspecified format)</td>
<td>See the audit.log(4) man page.</td>
</tr>
<tr>
<td>path</td>
<td>Path information</td>
<td>“path Token” on page 651</td>
</tr>
<tr>
<td>path_attr</td>
<td>Access path information</td>
<td>“path_attr Token” on page 651</td>
</tr>
<tr>
<td>privilege</td>
<td>Privilege set information</td>
<td>“privilege Token” on page 651</td>
</tr>
<tr>
<td>process</td>
<td>Process token information</td>
<td>“process Token” on page 651</td>
</tr>
<tr>
<td>return</td>
<td>Status of system call</td>
<td>“return Token” on page 652</td>
</tr>
<tr>
<td>sequence</td>
<td>Sequence number token</td>
<td>“sequence Token” on page 652</td>
</tr>
<tr>
<td>socket</td>
<td>Socket type and addresses</td>
<td>“socket Token” on page 652</td>
</tr>
<tr>
<td>subject</td>
<td>Subject token (same format as process token)</td>
<td>“subject Token” on page 653</td>
</tr>
<tr>
<td>text</td>
<td>ASCII string</td>
<td>“text Token” on page 653</td>
</tr>
<tr>
<td>trailer</td>
<td>Indicates end of audit record</td>
<td>“trailer Token” on page 654</td>
</tr>
<tr>
<td>uauth</td>
<td>Use of authorization</td>
<td>“uauth Token” on page 654</td>
</tr>
</tbody>
</table>
 Audit Token Formats

<table>
<thead>
<tr>
<th>Token Name</th>
<th>Description</th>
<th>For More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>upriv</td>
<td>Use of privilege</td>
<td>&quot;upriv Token&quot; on page 654</td>
</tr>
<tr>
<td>zonename</td>
<td>Name of zone</td>
<td>&quot;zonename Token&quot; on page 654</td>
</tr>
</tbody>
</table>

An audit record always begins with a header token. The header token indicates where the audit record begins in the audit trail. In the case of attributable events, the subject and the process tokens refer to the values of the process that caused the event. In the case of nonattributable events, the process token refers to the system.

**acl Token**

The acl token records information about Access Control Lists (ACLs).

The `praudit -x` command shows the fields of the acl token:

```
<acl type="1" value="root" mode="6"/>
```

**arg Token**

The arg token contains information about the arguments to a system call: the argument number of the system call, the argument value, and an optional description. This token allows a 32-bit integer system-call argument in an audit record.

The `praudit -x` command shows the fields of the arg token:

```
<argument arg-num="2" value="0x0" desc="new file uid"/>
```

**attribute Token**

The attribute token contains information from the file vnode.

The attribute token usually accompanies a path token. The attribute token is produced during path searches. If a path-search error occurs, there is no vnode available to obtain the necessary file information. Therefore, the attribute token is not included as part of the audit record. The `praudit -x` command shows the fields of the attribute token:

```
<attribute mode="100644" uid="adm" gid="adm" fsid="136" nodeid="2040" device="0"/>
```
cmd Token

The cmd token records the list of arguments and the list of environment variables that are associated with a command.

The `praudit -x` command shows the fields of the cmd token. The following is a truncated cmd token. The line is wrapped for display purposes.

```xml
<cmd>
<arge>WINDOWID=6823679</arge>
<arge>COLORTERM=gnome-terminal</arge>
<arge>...LANG=C</arge>...<arge>HOST=machine1</arge>
<arge>LPDEST=printer1</arge>...
</cmd>
```

exec_args Token

The exec_args token records the arguments to an exec() system call. The exec_args token has two fixed fields:

- A token ID field that identifies this token as an exec_args token
- A count that represents the number of arguments that are passed to the exec() system call

The remainder of this token is composed of count strings. The `praudit -x` command shows the fields of the exec_args token:

```xml
<exec_args>
<arg>/usr/bin/sh</arg><arg>/usr/bin/hostname</arg>
</exec_args>
```

Note – The exec_args token is output only when the argv audit policy option is active.

exec_env Token

The exec_env token records the current environment variables to an exec() system call. The exec_env token has two fixed fields:

- A token ID field that identifies this token as an exec_env token
- A count that represents the number of arguments that are passed to the exec() system call

The remainder of this token is composed of count strings. The `praudit -x` command shows the fields of the exec_env token. The line is wrapped for display purposes.

```xml
<exec_env>
<env>_/=/usr/bin/hostname</env>
<env>DTXSERVERLOCATION=local</env><env>SESSIONTYPE=altDt</env>
<env>LANG=C</env><env>SDT_NO_TOOLTALK=1</env><env>SDT_ALT_HELLO=/bin/true</env>
<env>PATH=/usr/bin:/usr/openwin/bin:/usr/ucb</env>
```
The exec_env token is output only when the large audit policy option is active.

file Token

The file token is a special token that is generated by the auditd daemon. The token marks the beginning of a new audit file and the end of an old audit file as the old file is deactivated. The initial file token identifies the previous file in the audit trail. The final file token identifies the next file in the audit trail. The auditd daemon builds a special audit record that contains this token to “link” together successive audit files into one audit trail.

The file token has four fields:

- A token ID that identifies this token as a file token
- A timestamp that identifies the date and the time that the file was created or was closed
- The file name length
- A field that holds the file null-terminated name

The praudit -x command shows the fields of the file token. This token identifies the next file in the audit trail. The line is wrapped for display purposes.

```
<file iso8601="2009-04-08 14:18:26.200 -07:00">
/var/audit/machine1/files/20090408211826.not_terminated.machine1</file>
```

fmri Token

The fmri token records the use of a fault management resource indicator (FMRI). For more information, see the smf(5) man page.

The praudit -x command shows the content of the fmri token:

```
<fmri service_instance="svc:/system/cryptosvc"></fmri>
```
groups **Token**

The groups token replaces the group token. The groups token records the group entries from the process's credential.

The `praudit -x` command shows the fields of the groups token:

```xml
<group><gid>staff</gid><gid>other</gid></group>
```

**Note** – The groups token is output only when the group audit policy option is active.

**header** **Token**

The header token is special in that it marks the beginning of an audit record. The header token combines with the trailer token to bracket all the other tokens in the record.

On 64-bit systems, the header token is displayed with a 64-bit timestamp, in place of the 32-bit timestamp.

The `praudit` command displays the header token as follows:

```
header,69,2,su,,machine1,2009-04-08 13:11:58.209 -07:00
```

The `praudit -x` command displays the fields of the header token at the beginning of the audit record. The line is wrapped for display purposes.

```
<record version="2" event="su" host="machine1"
iso8601="2009-04-08 13:11:58.209 -07:00">
```

**in_addr** **Token**

The in_addr token contains an Internet Protocol address. Since the Solaris 8 release, the Internet address can be displayed in IPv4 format or IPv6 format. The IPv4 address uses 4 bytes. The IPv6 address uses 1 byte to describe the address type, and 16 bytes to describe the address.

The `praudit -x` command shows the content of the in_addr token:

```
<ip_address>machine1</ip_address>
```

**ipc** **Token**

The ipc token contains the System V IPC message handle, semaphore handle, or shared-memory handle that is used by the caller to identify a particular IPC object.
Note – The IPC object identifiers violate the context-free nature of the Solaris audit tokens. No global “name” uniquely identifies IPC objects. Instead, IPC objects are identified by their handles. The handles are valid only during the time that the IPC objects are active. However, the identification of IPC objects should not be a problem. The System V IPC mechanisms are seldom used, and the mechanisms all share the same audit class.

The following table shows the possible values for the IPC object type field. The values are defined in the `/usr/include/bsm/audit.h` file.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AU_IPC_MSG</td>
<td>1</td>
<td>IPC message object</td>
</tr>
<tr>
<td>AU_IPC_SEM</td>
<td>2</td>
<td>IPC semaphore object</td>
</tr>
<tr>
<td>AU_IPC_SHM</td>
<td>3</td>
<td>IPC shared-memory object</td>
</tr>
</tbody>
</table>

The `praudit -x` command shows the fields of the `ipc` token:

```
<IPC ipc-type="shm" ipc-id="15"/>
```

**ipc_perm Token**

The `ipc_perm` token contains a copy of the System V IPC access permissions. This token is added to audit records that are generated by IPC shared-memory events, IPC semaphore events, and IPC message events.

The `praudit -x` command shows the fields of the `ipc_perm` token. The line is wrapped for display purposes.

```
<IPC_perm uid="jdoe" gid="staff" creator-uid="jdoe"
creator-gid="staff" mode="100600" seq="0" key="0x0"/>
```

The values are taken from the `ipc_perm` structure that is associated with the IPC object.

**iport Token**

The `iport` token contains the TCP or UDP port address.

The `praudit` command displays the `iport` token as follows:

```
ip port, 0xf6d6
```
The path token contains access path information for an object. The praudit -x command shows the content of the path token:

<path>/etc/security/audit_user</path>

The path_attr token contains access path information for an object. The access path specifies the sequence of attribute file objects below the path token object. Systems calls such as openat() access attribute files. For more information on attribute file objects, see the fsattr(5) man page.

The praudit command displays the path_attr token as follows:

path_attr,1,attr_file_name

The privilege token records the use of privileges on a process. The privilege token is not recorded for privileges in the basic set. If a privilege has been removed from the basic set by administrative action, then the use of that privilege is recorded. For more information on privileges, see "Privileges (Overview)" on page 164.

The praudit -x command shows the fields of the privilege token. The line is wrapped for display purposes.

<privilege set-type="Effective">file_chown,file_dac_read, file_dac_write,net_privaddr,proc_exec,proc_fork,proc_setid</privilege>

The process token contains information about a user who is associated with a process, such as the recipient of a signal.

The praudit -x command shows the fields of the process token. The line is wrapped for display purposes.

<process audit-uid="-2" uid="root" gid="root" ruid="root" rgid="root" pid="9" sid="0" tid="0 0 0.0 0.0.0"/>
**return Token**

The return token contains the return status of the system call (**u_error**) and the process return value (**u_rval1**).

The return token is always returned as part of kernel-generated audit records for system calls. In application auditing, this token indicates exit status and other return values.

The `praudit` command displays the return token for a system call as follows:

```
return, failure: Operation now in progress, -1
```

The `praudit -x` command shows the fields of the return token:

```
<return errval="failure: Operation now in progress" rval="-1"/>
```

**sequence Token**

The sequence token contains a sequence number. The sequence number is incremented every time an audit record is added to the audit trail. This token is useful for debugging.

The `praudit -x` command shows the content of the sequence token:

```
<sequence seq-num="1292"/>
```

**Note**—The sequence token is output only when the `seq` audit policy option is active.

**socket Token**

The socket token contains information that describes an Internet socket. In some instances, the token has four fields:

- A token ID that identifies this token as a socket token
- A socket type field that indicates the type of socket referenced, either TCP, UDP, or UNIX
- The local port
- The local IP address

The `praudit` command displays this instance of the socket token as follows:

```
socket, 0x0002, 0x83b1, localhost
```
In most instances, the token has eight fields:

- A token ID that identifies this token as a socket token
- The socket domain
- A socket type field that indicates the type of socket referenced, either TCP, UDP, or UNIX
- The local port
- The address type, either IPv4 or IPv6
- The local IP address
- The remote port
- The remote IP address

Since the Solaris 8 release, the Internet address can be displayed in IPv4 format or IPv6 format. The IPv4 address uses 4 bytes. The IPv6 address uses 1 byte to describe the address type, and 16 bytes to describe the address.

The `praudit -x` command shows the fields of the socket token. The line is wrapped for display purposes.

```xml
<socket sock_domain="0x0002" sock_type="0x0002" lport="0x83cf" laddr="example1" fport="0x2383" faddr="server1.Subdomain.Domain.COM"/>
```

**subject Token**

The subject token describes a user who performs or attempts to perform an operation. The format is the same as the process token.

The subject token is always returned as part of kernel-generated audit records for system calls. The `praudit` command displays the subject token as follows:

```xml
<subject audit-uid="jdoe" uid="root" gid="root" ruid="root" rgid="root" pid="1631" sid="1421584480" tid="8243 65558 machine1"/>
```

**text Token**

The text token contains a text string.

The `praudit -x` command shows the content of the text token:

```xml
<text>booting kernel</text>
```
**trailer Token**

The two tokens, header and trailer, are special in that they distinguish the end points of an audit record and bracket all the other tokens. A header token begins an audit record. A trailer token ends an audit record. The trailer token is an optional token. The trailer token is added as the last token of each record only when the trail audit policy option has been set.

When an audit record is generated with trailers turned on, the audit reduce command can verify that the trailer correctly points back to the record header. The trailer token supports backward seeks of the audit trail.

The praudit command displays the trailer token as follows:

```
trailer,136
```

**uauth Token**

The uauth token records the use of authorization with a command or action.

The praudit command displays the uauth token as follows:

```
use of authorization,solaris.admin.printer.delete
```

**upriv Token**

The upriv token records the use of privilege with a command or action.

The praudit -x command shows the fields of the upriv token:

```
<use_of_privilege result="successful use of priv">proc_setid</use_of_privilege>
```

**zonename Token**

The zonename token records the zone in which the audit event occurred. The string "global" indicates audit events that occur in the global zone.

The praudit -x command shows the content of the zonename token:

```
<zone name="graphzone"/>
```
Glossary

**Access Control List (ACL)**
An access control list (ACL) provides finer-grained file security than traditional UNIX file protection provides. For example, an ACL enables you to allow group read access to a file, while allowing only one member of that group to write to the file.

**admin principal**
A user principal with a name of the form `username/admin` (as in `jdoe/admin`). An admin principal can have more privileges (for example, to change policies) than a regular user principal. See also principal name, user principal.

**AES**

**algorithm**
A cryptographic algorithm. This is an established, recursive computational procedure that encrypts or hashes input.

**application server**
See network application server.

**asynchronous audit event**
Asynchronous events are the minority of system events. These events are not associated with any process, so no process is available to be blocked and later woken up. Initial system boot and PROM enter and exit events are examples of asynchronous events.

**audit files**
Binary audit logs. Audit files are stored separately in an audit partition.

**audit partition**
A hard disk partition that is configured to hold audit files.

**audit policy**
The global and per-user settings that determine which audit events are recorded. The global settings that apply to the audit service typically affect which pieces of optional information are included in the audit trail. Two settings, `cnt` and `ahlt`, affect the operation of the system when the audit queue fills. For example, audit policy might require that a sequence number be part of every audit record.

**audit trail**
The collection of all audit files from all hosts.

**authentication**
The process of verifying the claimed identity of a principal.

**authenticator**
Authenticators are passed by clients when requesting tickets (from a KDC) and services (from a server). They contain information that is generated by using a session key known only by the client and server, that can be verified as of recent origin, thus indicating that the transaction is secure. When used with a ticket, an authenticator can be used to authenticate a user principal. An authenticator includes the principal name of the user, the IP address of the user’s host, and a time stamp. Unlike a ticket, an authenticator can be used only once, usually when access to a service is requested. An authenticator is encrypted by using the session key for that client and that server.
authorization

1. In Kerberos, the process of determining if a principal can use a service, which objects the principal is allowed to access, and the type of access that is allowed for each object.

2. In role-based access control (RBAC), a permission that can be assigned to a role or user (or embedded in a rights profile) for performing a class of actions that are otherwise prohibited by security policy.

Basic Security Module (BSM)

The Solaris auditing service and device allocation. Together, these features satisfy the C2 level of security.

basic set

The set of privileges that are assigned to a user’s process at login. On an unmodified system, each user’s initial inheritable set equals the basic set at login.

Blowfish

A symmetric block cipher algorithm that takes a variable-length key from 32 bits to 448 bits. Its author, Bruce Schneier, claims that Blowfish is optimized for applications where the key does not change often.

client

Narrowly, a process that makes use of a network service on behalf of a user; for example, an application that uses rlogin. In some cases, a server can itself be a client of some other server or service.

More broadly, a host that a) receives a Kerberos credential, and b) makes use of a service that is provided by a server.

Informally, a principal that makes use of a service.

client principal

(RPCSEC_GSS API) A client (a user or an application) that uses RPCSEC_GSS-secured network services. Client principal names are stored in the form of rpc_gss_principal_t structures.

clock skew

The maximum amount of time that the internal system clocks on all hosts that are participating in the Kerberos authentication system can differ. If the clock skew is exceeded between any of the participating hosts, requests are rejected. Clock skew can be specified in the krb5.conf file.

confidentiality

See privacy.

consumer

In the Solaris Cryptographic Framework, a consumer is a user of the cryptographic services that come from providers. Consumers can be applications, end users, or kernel operations. Kerberos, IKE, and IPsec are examples of consumers. For examples of providers, see provider.

credential

An information package that includes a ticket and a matching session key. Used to authenticate the identity of a principal. See also ticket, session key.

credential cache

A storage space (usually a file) that contains credentials that are received from the KDC.

cryptographic algorithm

See algorithm.

DES

Data Encryption Standard. A symmetric-key encryption method developed in 1975 and standardized by ANSI in 1981 as ANSI X.3.92. DES uses a 56-bit key.

device allocation

Device protection at the user level. Device allocation enforces the exclusive use of a device by one user at a time. Device data is purged before device reuse. Authorizations can be used to limit who is permitted to allocate a device.
| **device policy** | Device protection at the kernel level. Device policy is implemented as two sets of privileges on a device. One set of privileges controls read access to the device. The second set of privileges controls write access to the device. See also policy. |
| **Diffie-Hellman protocol** | Also known as public key cryptography. An asymmetric cryptographic key agreement protocol that was developed by Diffie and Hellman in 1976. The protocol enables two users to exchange a secret key over an insecure medium without any prior secrets. Diffie-Hellman is used by Kerberos. |
| **digest** | See message digest. |
| **DSA** | Digital Signature Algorithm. A public key algorithm with a variable key size from 512 to 4096 bits. The U.S. Government standard, DSS, goes up to 1024 bits. DSA relies on SHA1 for input. |
| **effective set** | The set of privileges that are currently in effect on a process. |
| **flavor** | Historically, security flavor and authentication flavor had the same meaning, as a flavor that indicated a type of authentication (AUTH_UNIX, AUTH_DES, AUTH_KERB). RPCSEC_GSS is also a security flavor, even though it provides integrity and privacy services in addition to authentication. |
| **forwardable ticket** | A ticket that a client can use to request a ticket on a remote host without requiring the client to go through the full authentication process on that host. For example, if the user david obtains a forwardable ticket while on user jennifer's machine, he can log in to his own machine without being required to get a new ticket (and thus authenticate himself again). See also proxiable ticket. |
| **FQDN** | Fully qualified domain name. For example, central.example.com (as opposed to simply denver). |
| **GSS-API** | The Generic Security Service Application Programming Interface. A network layer that provides support for various modular security services, including the Kerberos service. GSS-API provides for security authentication, integrity, and privacy services. See also authentication, integrity, privacy. |
| **hardening** | The modification of the default configuration of the operating system to remove security vulnerabilities that are inherent in the host. |
| **hardware provider** | In the Solaris Cryptographic Framework, a device driver and its hardware accelerator. Hardware providers offload expensive cryptographic operations from the computer system, thus freeing CPU resources for other uses. See also provider. |
| **host** | A machine that is accessible over a network. |
| **host principal** | A particular instance of a service principal in which the principal (signified by the primary name host) is set up to provide a range of network services, such as ftp, rcp, or rlogin. An example of a host principal is host/central.example.com@example.com. See also server principal. |
| **inheritable set** | The set of privileges that a process can inherit across a call to exec. |
| **initial ticket** | A ticket that is issued directly (that is, not based on an existing ticket-granting ticket). Some services, such as applications that change passwords, might require tickets to be marked initial so as to assure themselves that the client can demonstrate a knowledge of its secret key. This assurance is important because an initial ticket indicates that the client has recently authenticated itself (instead of relying on a ticket-granting ticket, which might existed for a long time). |
instance

The second part of a principal name, an instance qualifies the principal’s primary. In the case of a service principal, the instance is required. The instance the host’s fully qualified domain name, as in host/central.example.com. For user principals, an instance is optional. Note, however, that jdoe and jdoe/admin are unique principals. See also primary, principal name, service principal, user principal.

integrity

A security service that, in addition to user authentication, provides for the validity of transmitted data through cryptographic checksumming. See also authentication, privacy.

invalid ticket

A postdated ticket that has not yet become usable. An invalid ticket is rejected by an application server until it becomes validated. To be validated, an invalid ticket must be presented to the KDC by the client in a TGS request, with the VALIDATE flag set, after its start time has passed. See also postdated ticket.

KDC

Key Distribution Center. A machine that has three Kerberos V5 components:

- Principal and key database
- Authentication service
- Ticket-granting service

Each realm has a master KDC and should have one or more slave KDCs.

Kerberos

An authentication service, the protocol that is used by that service, or the code that is used to implement that service.

The Solaris Kerberos implementation that is closely based on Kerberos V5 implementation.

While technically different, “Kerberos” and “Kerberos V5” are often used interchangeably in the Kerberos documentation.

Kerberos (also spelled Cerberus) was a fierce, three-headed mastiff who guarded the gates of Hades in Greek mythology.

Kerberos policy

A set of rules that governs password usage in the Kerberos service. Policies can regulate principals’ accesses, or ticket parameters, such as lifetime.
key

1. Generally, one of two main types of keys:
   ■ A symmetric key – An encryption key that is identical to the decryption key. Symmetric keys are used to encrypt files.
   ■ An asymmetric key or public key – A key that is used in public key algorithms, such as Diffie-Hellman or RSA. Public keys include a private key that is known only by one user, a public key that is used by the server or general resource, and a private-public key pair that combines the two. A private key is also called a secret key. The public key is also called a shared key or common key.
   ■ 2. An entry (principal name) in a keytab file. See also keytab file.

3. In Kerberos, an encryption key, of which there are three types:
   ■ A private key – An encryption key that is shared by a principal and the KDC, and distributed outside the bounds of the system. See also private key.
   ■ A service key – This key serves the same purpose as the private key, but is used by servers and services. See also service key.
   ■ A session key – A temporary encryption key that is used between two principals, with a lifetime limited to the duration of a single login session. See also session key.

keytab file

A key table file that contains one or more keys (principals). A host or service uses a keytab file in the much the same way that a user uses a password.

kvno

Key version number. A sequence number that tracks a particular key in order of generation. The highest kvno is the latest and most current key.

limit set

The outside limit of what privileges are available to a process and its children.

MAC

1. See message authentication code (MAC).

2. Also called labeling. In government security terminology, MAC is Mandatory Access Control. Labels such as Top Secret and Confidential are examples of MAC. MAC contrasts with DAC, which is Discretionary Access Control. UNIX permissions are an example of DAC.

3. In hardware, the unique machine address on a LAN. If the machine is on an Ethernet, the MAC is the Ethernet address.

master KDC

The main KDC in each realm, which includes a Kerberos administration server, kadmind, and an authentication and ticket-granting daemon, krb5kdc. Each realm must have at least one master KDC, and can have many duplicate, or slave, KDCs that provide authentication services to clients.

MD5

An iterative cryptographic hash function that is used for message authentication, including digital signatures. The function was developed in 1991 by Rivest.

mechanism

1. A software package that specifies cryptographic techniques to achieve data authentication or confidentiality. Examples: Kerberos V5, Diffie-Hellman public key.

2. In the Solaris Cryptographic Framework, an implementation of an algorithm for a particular purpose. For example, a DES mechanism that is applied to authentication, such as CKM_DES_MAC, is a separate mechanism from a DES mechanism that is applied to encryption, CKM_DES_CBC_PAD.
**message authentication code (MAC)**

MAC provides assurance of data integrity and authenticates data origin. MAC does not protect against eavesdropping.

**message digest**

A message digest is a hash value that is computed from a message. The hash value almost uniquely identifies the message. A digest is useful for verifying the integrity of a file.

**minimization**

The installation of the minimal operating system that is necessary to run the server. Any software that does not directly relate to the operation of the server is either not installed, or deleted after the installation.

**name service scope**

The scope in which a role is permitted to operate, that is, an individual host or all hosts that are served by a specified name service such as NIS, NIS+, or LDAP. Scopes are applied to Solaris Management Console toolboxes.

**network application server**

A server that provides a network application, such as ftp. A realm can contain several network application servers.

**nonattributable audit event**

An audit event whose initiator cannot be determined, such as the AUE_BOOT event.

**NTP**

Network Time Protocol. Software from the University of Delaware that enables you to manage precise time or network clock synchronization, or both, in a network environment. You can use NTP to maintain clock skew in a Kerberos environment. See also clock skew.

**PAM**

Pluggable Authentication Module. A framework that allows for multiple authentication mechanisms to be used without having to recompile the services that use them. PAM enables Kerberos session initialization at login.

**passphrase**

A phrase that is used to verify that a private key was created by the passphrase user. A good passphrase is 10-30 characters long, mixes alphabetic and numeric characters, and avoids simple prose and simple names. You are prompted for the passphrase to authenticate use of the private key to encrypt and decrypt communications.

**password policy**

The encryption algorithms that can be used to generate passwords. Can also refer to more general issues around passwords, such as how often the passwords must be changed, how many mis-entries are permitted, and other security considerations. Security policy requires passwords. Password policy might require passwords to be encrypted with the MD5 algorithm, and might make further requirements related to password strength.

**permitted set**

The set of privileges that are available for use by a process.

**policy**

Generally, a plan or course of action that influences or determines decisions and actions. For computer systems, policy typically means security policy. Your site’s security policy is the set of rules that define the sensitivity of the information that is being processed and the measures that are used to protect the information from unauthorized access. For example, security policy might require that systems be audited, that devices be protected with privileges, and that passwords be changed every six weeks.

For the implementation of policy in specific areas of the Solaris OS, see audit policy, policy in the cryptographic framework, device policy, Kerberos policy, password policy, and RBAC policy.
### policy for public key technologies

In the Key Management Framework (KMF), policy is the management of certificate usage. The KMF policy database can put constraints on the use of the keys and certificates that are managed by the KMF library.

### policy in the cryptographic framework

In the Solaris Cryptographic Framework, policy is the disabling of existing cryptographic mechanisms. The mechanisms then cannot be used. Policy in the cryptographic framework might prevent the use of a particular mechanism, such as `CKM_DES_CBC`, from a provider, such as DES.

### postdated ticket

A postdated ticket does not become valid until some specified time after its creation. Such a ticket is useful, for example, for batch jobs that are intended to run late at night, since the ticket, if stolen, cannot be used until the batch job is run. When a postdated ticket is issued, it is issued as *invalid* and remains that way until a) its start time has passed, and b) the client requests validation by the KDC. A postdated ticket is normally valid until the expiration time of the ticket-granting ticket. However, if the postdated ticket is marked *renewable*, its lifetime is normally set to be equal to the duration of the full life time of the ticket-granting ticket. See also invalid ticket, renewable ticket.

### principal

1. A uniquely named client/user or server/service instance that participates in a network communication. Kerberos transactions involve interactions between principals (service principals and user principals) or between principals and KDCs. In other words, a principal is a unique entity to which Kerberos can assign tickets. See also principal name, service principal, user principal.

2. (RPCSEC_GSS API) See client principal, server principal.

### principal name

1. The name of a principal, in the format `primary/instance@REALM`. See also instance, primary, realm.

2. (RPCSEC_GSS API) See client principal, server principal.

### privacy

A security service, in which transmitted data is encrypted before being sent. Privacy also includes data integrity and user authentication. See also authentication, integrity, service.

### private key

A key that is given to each user principal, and known only to the user of the principal and to the KDC. For user principals, the key is based on the user's password. See also key.

### private-key encryption

In private-key encryption, the sender and receiver use the same key for encryption. See also public-key encryption.

### privilege

A discrete right on a process in a Solaris system. Privileges offer a finer-grained control of processes than does root. Privileges are defined and enforced in the kernel. For a full description of privileges, see the privileges(5) man page.

### privilege model

A stricter model of security on a computer system than the superuser model. In the privilege model, processes require privilege to run. Administration of the system can be divided into discrete parts that are based on the privileges that administrators have in their processes. Privileges can be assigned to an administrator's login process. Or, privileges can be assigned to be in effect for certain commands only.

### privilege set

A collection of privileges. Every process has four sets of privileges that determine whether a process can use a particular privilege. See limit set, effective set, permitted set, and inheritable set set.

Also, the basic set set of privileges is the collection of privileges that are assigned to a user's process at login.
privileged application
An application that can override system controls. The application checks for security attributes, such as specific UIDs, GIDs, authorizations, or privileges.

profile shell
In RBAC, a shell that enables a role (or user) to run from the command line any privileged applications that are assigned to the role's rights profiles. The profile shells are pfsh, pfcsh, and pfksh. They correspond to the Bourne shell (sh), C shell (csh), and Korn shell (ksh), respectively.

provider
In the Solaris Cryptographic Framework, a cryptographic service that is provided to consumers. PKCS #11 libraries, kernel cryptographic modules, and hardware accelerators are examples of providers. Providers plug in to the Solaris Cryptographic Framework, so are also called plugins. For examples of consumers, see consumer.

proxiable ticket
A ticket that can be used by a service on behalf of a client to perform an operation for the client. Thus, the service is said to act as the client's proxy. With the ticket, the service can take on the identity of the client. The service can use a proxiable ticket to obtain a service ticket to another service, but it cannot obtain a ticket-granting ticket. The difference between a proxiable ticket and a forwardable ticket is that a proxiable ticket is only valid for a single operation. See also forwardable ticket.

public-key encryption
An encryption scheme in which each user has two keys, one public key and one private key. In public-key encryption, the sender uses the receiver's public key to encrypt the message, and the receiver uses a private key to decrypt it. The Kerberos service is a private-key system. See also private-key encryption.

QOP
Quality of Protection. A parameter that is used to select the cryptographic algorithms that are used in conjunction with the integrity service or privacy service.

RBAC
Role-Based Access Control. An alternative to the all-or-nothing superuser model. RBAC lets an organization separate superuser's capabilities and assign them to special user accounts called roles. Roles can be assigned to specific individuals according to their responsibilities.

RBAC policy
The security policy that is associated with a command. Currently, suser and solars are the valid policies. The suser policy recognizes privileges and setuid security attributes. The suser policy recognizes only setuid security attributes. Trusted Solaris™ systems, which can interoperate with a Solaris system, provide a t sol policy, which recognizes privileges, setuid security attributes, and labels on processes.

realm
1. The logical network that is served by a single Kerberos database and a set of Key Distribution Centers (KDCs).
2. The third part of a principal name. For the principal name jdoe/admin@ENG.EXAMPLE.COM, the realm is ENG.EXAMPLE.COM. See also principal name.

relation
A configuration variable or relationship that is defined in the kdc.conf or krb5.conf files.

renewable ticket
Because having tickets with very long lives is a security risk, tickets can be designated as renewable. A renewable ticket has two expiration times: a) the time at which the current instance of the ticket expires, and b) maximum lifetime for any ticket. If a client wants to continue to use a ticket, the client renews the ticket before the first expiration occurs. For example, a ticket can be valid for one hour, with all tickets having a maximum lifetime of ten hours. If the client that holds the ticket wants to keep it for more than an hour, the client must renew the ticket. When a ticket reaches the maximum ticket lifetime, it automatically expires and cannot be renewed.
<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>rights profile</td>
<td>Also referred to as a right or a profile. A collection of overrides used in RBAC that can be assigned to a role or user. A rights profile can consist of authorizations, commands with security attributes, and other rights profiles.</td>
</tr>
<tr>
<td>role</td>
<td>A special identity for running privileged applications that only assigned users can assume.</td>
</tr>
<tr>
<td>RSA</td>
<td>A method for obtaining digital signatures and public key cryptosystems. The method was first described in 1978 by its developers, Rivest, Shamir, and Adleman.</td>
</tr>
<tr>
<td>scan engine</td>
<td>A third-party application, residing on an external host, that examines a file for known viruses.</td>
</tr>
<tr>
<td>SEAM</td>
<td>Sun Enterprise Authentication Mechanism. The product name for the initial versions of a system for authenticating users over a network, based on the Kerberos V5 technology that was developed at the Massachusetts Institute of Technology. The product is now called the Kerberos service. SEAM refers to parts the Kerberos service that were not included in various Solaris releases.</td>
</tr>
<tr>
<td>secret key</td>
<td>See private key.</td>
</tr>
<tr>
<td>Secure Shell</td>
<td>A special protocol for secure remote login and other secure network services over an insecure network.</td>
</tr>
<tr>
<td>security attributes</td>
<td>In RBAC, overrides to security policy that enable an administrative command to succeed when the command is run by a user other than superuser. In the superuser model, the setuid and setgid programs are security attributes. When these attributes are applied to a command, the command succeeds no matter who runs the command. In the privilege model, security attributes are privileges. When a privilege is given to a command, the command succeeds. The privilege model is compatible with the superuser model, in that the privilege model also recognizes the setuid and setgid programs as security attributes.</td>
</tr>
<tr>
<td>security flavor</td>
<td>See flavor.</td>
</tr>
<tr>
<td>security mechanism</td>
<td>See mechanism.</td>
</tr>
<tr>
<td>security policy</td>
<td>See policy.</td>
</tr>
<tr>
<td>security service</td>
<td>See service.</td>
</tr>
<tr>
<td>seed</td>
<td>A numeric starter for generating random numbers. When the starter originates from a random source, the seed is called a random seed.</td>
</tr>
<tr>
<td>server</td>
<td>A principal that provides a resource to network clients. For example, if you rlogin to the machine central.example.com, then that machine is the server that provides the rlogin service. See also service principal.</td>
</tr>
<tr>
<td>server principal</td>
<td>(RPCSEC_GSSAPI) A principal that provides a service. The server principal is stored as an ASCII string in the form service@host. See also client principal.</td>
</tr>
<tr>
<td>service</td>
<td>1. A resource that is provided to network clients, often by more than one server. For example, if you rlogin to the machine central.example.com, then that machine is the server that provides the rlogin service.</td>
</tr>
<tr>
<td></td>
<td>2. A security service (either integrity or privacy) that provides a level of protection beyond authentication. See also integrity and privacy.</td>
</tr>
<tr>
<td>term</td>
<td>definition</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>service key</td>
<td>An encryption key that is shared by a service principal and the KDC, and is distributed outside the bounds of the system. See also key.</td>
</tr>
<tr>
<td>service principal</td>
<td>A principal that provides Kerberos authentication for a service or services. For service principals, the primary name is a name of a service, such as ftp, and its instance is the fully qualified host name of the system that provides the service. See also host principal, user principal.</td>
</tr>
<tr>
<td>session key</td>
<td>A key that is generated by the authentication service or the ticket-granting service. A session key is generated to provide secure transactions between a client and a service. The lifetime of a session key is limited to a single login session. See also key.</td>
</tr>
<tr>
<td>SHA1</td>
<td>Secure Hashing Algorithm. The algorithm operates on any input length less than $2^{64}$ to produce a message digest. The SHA1 algorithm is input to DSA.</td>
</tr>
<tr>
<td>single-system image</td>
<td>A single-system image is used in Solaris auditing to describe a group of audited machines that use the same naming service. These machines send their audit records to a central audit server, where the records can be compared as if the records came from one machine.</td>
</tr>
<tr>
<td>slave KDC</td>
<td>A copy of a master KDC, which is capable of performing most functions of the master. Each realm usually has several slave KDCs (and only one master KDC). See also KDC, master KDC.</td>
</tr>
<tr>
<td>software provider</td>
<td>In the Solaris Cryptographic Framework, a kernel software module or a PKCS #11 library that provides cryptographic services. See also provider.</td>
</tr>
<tr>
<td>stash file</td>
<td>A stash file contains an encrypted copy of the master key for the KDC. This master key is used when a server is rebooted to automatically authenticate the KDC before it starts the kadmin and krb5kdc processes. Because the stash file includes the master key, the stash file and any backups of it should be kept secure. If the encryption is compromised, then the key could be used to access or modify the KDC database.</td>
</tr>
<tr>
<td>superuser model</td>
<td>The typical UNIX model of security on a computer system. In the superuser model, an administrator has all-or-nothing control of the machine. Typically, to administer the machine, a user becomes superuser (root) and can do all administrative activities.</td>
</tr>
<tr>
<td>synchronous audit event</td>
<td>The majority of audit events. These events are associated with a process in the system. A non-attributable event that is associated with a process is a synchronous event, such as a failed login.</td>
</tr>
<tr>
<td>TGS</td>
<td>Ticket-Granting Service. That portion of the KDC that is responsible for issuing tickets.</td>
</tr>
<tr>
<td>TGT</td>
<td>Ticket-Granting Ticket. A ticket that is issued by the KDC that enables a client to request tickets for other services.</td>
</tr>
<tr>
<td>ticket</td>
<td>An information packet that is used to securely pass the identity of a user to a server or service. A ticket is valid for only a single client and a particular service on a specific server. A ticket contains the principal name of the service, the principal name of the user, the IP address of the user's host, a time stamp, and a value that defines the lifetime of the ticket. A ticket is created with a random session key to be used by the client and the service. Once a ticket has been created, it can be reused until the ticket expires. A ticket only serves to authenticate a client when it is presented along with a fresh authenticator. See also authenticator, credential, service, session key.</td>
</tr>
<tr>
<td>ticket file</td>
<td>See credential cache.</td>
</tr>
</tbody>
</table>
### user principal
A principal that is attributed to a particular user. A user principal’s primary name is a user name, and its optional instance is a name that is used to described the intended use of the corresponding credentials (for example, jdoe or jdoe/admin). Also known as a user instance. See also service principal.

### virtual private network (VPN)
A network that provides secure communication by using encryption and tunneling to connect users over a public network.
Index

Numbers and Symbols

$$(double dollar sign), parent shell process number, 222
[] (square brackets), auditrecord output, 644
*(asterisk)
  checking for in RBAC authorizations, 203
device_allocate file, 101, 102
  wildcard character
    in RBAC authorizations, 210, 213
@ (at sign), device_allocate file, 102
\ (backslash)
  device_allocate file, 101
devices file, 100
^ (caret) in audit class prefixes, 639
. (dot)
  authorization name separator, 210
displaying hidden files, 137
= (equal sign), file permissions symbol, 132
- (minus sign)
  audit class prefix, 639
  file permissions symbol, 132
  file type symbol, 128
  suloq file, 72
+ (plus sign)
  ACL entry, 144
  audit class prefix, 639
  file permissions symbol, 132
  suloq file, 72
# (pound sign)
  device_allocate file, 101
devices file, 100
; (semicolon)
  device_allocate file, 101
  separator of security attributes, 217
> (redirect output), preventing, 49
>> (append output), preventing, 49
~/.gkadmin file, description, 533
~/.k5login file, description, 533
~/.rhosts file, description, 353
~/.hosts file, description, 353
~/.ssh/authorized_keys file
  description, 353
  override, 354
~/.ssh/config file
  description, 354
  override, 354
~/.ssh/environment file, description, 353
~/.ssh/id_dsa file, override, 354
~/.ssh/id_rsa file, override, 354
~/.ssh/identity file, override, 354
~/.ssh/known_hosts file
  description, 353
  override, 354
~/.ssh/rc file, description, 353
3des-cbc encryption algorithm, ssh_config file, 347
3des encryption algorithm, ssh_config file, 347

A
  -A option, auditreduce command, 605
  -a option
    auditrecord command, 602

667
administering (Continued)
admitting records, 557
audit trail overflow prevention, 611
auditreduce command, 604-606
cost control, 571
description, 552
efficiency, 573
process preselection mask, 627
reducing storage-space requirements, 572-573
task map, 575
in zones, 560-561, 637
auditing in zones, 564-565
cryptographic framework, 246
cryptographic framework and zones, 248
cryptographic framework task map, 263-264
device allocation, 87-88
device policy, 84
dial-up logins, 65
file permissions, 137
Kerberos
keytabs, 509-515
policies, 497-505
principals, 483-496
metaslot, 246
NFS client-server file security, 293-296
password algorithms, 67
privileges, 221
properties of a role, 195-196
RBAC properties, 196-199
remote logins with Solaris Secure Shell, 332-334
rights profiles, 196-199
role password, 193-194
roles, 177-180
roles to replace superuser, 175-177
Secure RPC task map, 296
Solaris Secure Shell
clients, 346
overview, 343-345
servers, 346
task map, 325
without privileges, 166
administrative (old) audit class, 638
administrative audit class, 638
AES kernel provider, 264

Index
aes128-cbc encryption algorithm, ssh_config file, 347
aes128-ctr encryption algorithm, ssh_config file, 347
agent daemon, Solaris Secure Shell, 336-337
ahlt audit policy
description, 568
setting, 591
algorithms
definition in cryptographic framework, 244
listing in the cryptographic framework, 264-267
password
configuration, 68-69
password encryption, 42
all, in user audit fields, 634
All (RBAC), rights profile, 209
all audit class
cautions for using, 639
description, 638
all hard string, audit_warn script, 636
allocate command
allocate error state, 100
authorizations for, 99
authorizations required, 220
description, 99
tape drive, 94
user authorization, 89
using, 93-94
allocate error state, 100
allocating devices
by users, 93-94
forcibly, 90
task map, 93
troubleshooting, 94
AllowGroups keyword, sshd_config file, 346
AllowTcpForwarding keyword
changing, 330
sshd_config file, 346
AllowUsers keyword, sshd_config file, 347
allsoft string, audit_warn script, 635
ALTSHELL in Solaris Secure Shell, 351
always-audit classes
audit_user database, 634
process preselection mask, 641
analysis, praudit command, 629
antivirus software, See virus scanning
appending arrow (>>, preventing appending, 49
application audit class, 638
application server, configuring, 411-413
arcfour encryption algorithm, ssh_config file, 347
ARCFOUR kernel provider, 264
Archive tape drive device-clean script, 102
archiving, audit files, 611
arg audit token, format, 646
args audit policy
and exec_env token, 647-648
description, 569
args policy, setting, 618
argv audit policy
and exec_args token, 647
description, 569
argv policy, setting, 618
assigning
privileges to commands in a rights profile, 225-226
privileges to commands in a script, 229
privileges to user or role, 226-227
to a role, 178, 179
to a user locally, 183-184
assuming role
how to, 188
in a terminal window, 188-190
in Solaris Management Console, 190-192
Primary Administrator, 189
root, 189-190
System Administrator, 190
asterisk (*)
checking for in RBAC authorizations, 203
device_allocate file, 101, 102
wildcard character
in RBAC authorizations, 210, 213
at command, authorizations required, 219
at sign (@), device_allocate file, 102
atq command, authorizations required, 219
attribute audit token, 646
attributes, keyword in BART, 124
audio devices, security, 103
audit administration audit class, 638
audit characteristics
  audit ID, 641
  process preselection mask, 627
  processes, 641
  session ID, 641
  terminal ID, 641
  user process preselection mask, 641

audit class file
  adding a class, 583-584
  description, 632
  troubleshooting, 584

audit class preselection, effect on public objects, 555

audit classes
  adding, 583-584
  definitions, 638
  description, 554, 555
  entries in audit control file, 632
  exceptions in audit user database, 634-635
  exceptions to system-wide settings, 556
  mapping events, 557
  modifying default, 583-584
  overview, 556-557
  prefixes, 639
  preselecting, 577-579
  preselection, 555
  process preselection mask, 641
  setting system-wide, 637
  syntax, 639
  system-wide, 632

audit command
  description, 627
  preselection mask for existing processes (-s option), 597
  rereading audit files (-s option), 627
  resetting directory pointer (-n option), 627
  updating audit service, 596-597
  verifying syntax of audit control file (-v option), 577

audit configuration file, See audit_control file

audit_control file
  audit daemon rereading after editing, 597
  changing kernel mask for nonattributable events, 597
  configuring, 577-579

audit_control file (Continued)
  description, 632
  entries, 632
  entries and zones, 637
  examples, 633
  exceptions to flags in audit_user database, 634-635
  flags line
    process preselection mask, 641
  minfree warning, 635
  plugin line, 579
  prefixes in flags line, 639
  syntax problem, 636
  system-wide audit, 556
  verifying classes, 614
  verifying syntax, 577

Audit Control rights profile, 636

audit daemon, See auditd daemon

audit directory
  description, 554
  partitioning for, 587-589
  sample structure, 628

audit event file
  changing class membership, 584-586
  description, 555
  removing events safely, 621

audit event-to-class mappings, auditd daemon’s role, 626-627

audit events
  audit_event file, 555
  changing class membership, 584-586
  description, 555
  mapping to classes, 557
  selecting from audit trail, 606-608
  selecting from audit trail in zones, 637
  summary, 554
  viewing from binary files, 608-610

audit files
  auditreduce command, 628
  combining, 604-606, 628
  configuring, 576-586
  copying messages to single file, 608
  limiting size of, 622
  managing, 611
audit files (Continued)
minimum free space for file systems, 633
names, 642, 643
order for opening, 633
partitioning disk for, 587-589
printing, 609
reducing, 604-606, 628
reducing storage-space requirements, 572-573, 573
switching to new file, 627
time stamps, 643
audit ID
mechanism, 641
overview, 551-553
audit logs
See also audit files
comparing binary and textual, 558
configuring textual audit logs, 579-581
in text, 633
modes, 558
audit messages, copying to single file, 608
audit.notice entry, syslog.conf file, 580
audit plugins, summary, 640
audit policy
audit tokens from, 641
auditd daemon's role, 626-627
defaults, 568-571
description, 555
effects of, 568-571
public, 570
setting, 590-592
setting ahlt, 591
setting arge, 618
setting argv, 618
setting in global zone, 560-561, 625-626, 637
setting per zone, 592
that does not affect tokens, 641
tokens added by, 641
audit preselection mask
modifying for existing users, 619-621
modifying for individual users, 581-583
audit queue control parameters, auditd daemon's role, 626-627
audit records
audit directories full, 627, 636
audit records (Continued)
converting to readable format, 609, 629
description, 555
displaying, 608-610
displaying formats of
procedure, 602-604
summary, 627
displaying formats of a program, 603
displaying formats of an audit class, 603-604
displaying in XML format, 610
events that generate, 553
format, 643
formatting example, 602
merging, 604-606
overview, 557
reducing audit files, 604-606
sequence of tokens, 643
syslog.conf file, 553
/var/adm/auditlog file, 580
Audit Review rights profile, 636
audit session ID, 641
audit threshold, 633
audit tokens
See also individual audit token names
added by audit policy, 641
audit record format, 643
description, 555, 557
format, 644
list of, 644
audit trail
analysis costs, 572
analysis with praudit command, 629
cleaning up not terminated files, 610-611
creating
auditd daemon's role, 627
description, 555
effect of audit policy on, 568
events included, 557
merging all files, 628
monitoring in real time, 573
no public objects, 555
overview, 553
preventing overflow, 611
selecting events from, 606-608
audit trail (Continued)

viewing events from, 608-610
viewing events from different zones, 637

audit_user database
exception to system-wide audit classes, 556
prefixes for classes, 639
process preselection mask, 641
specifying user exceptions, 581-583
user audit fields, 634-635

audit_user file, verifying classes, 614

audit_warn script
auditd daemon execution of, 627
conditions invoking, 635
configuring, 589-590
description, 635
strings, 636

auditconfig command
audit classes as arguments, 557, 637
description, 631
prefixes for classes, 639

auditconfig command, setting audit policy, 618

auditconfig command
setting audit policy temporarily, 591

auditd daemon
audit trail creation, 626-627, 642
audit_warn script
description, 635
execution of, 627
configuration functions, 626-627
functions, 626-627
order audit files are opened, 633
plugins loaded by, 626-627
rereading information for the kernel, 596
rereading the audit_control file, 597

auditd service
configuring policy, 590-592
configuring queue parameters, 592-594

auditd service, enable on the command line, 627

for auditing, 568

auditing
all commands by users, 617-618
changes in device policy, 86
configuring identically for all zones, 598-600

auditing (Continued)
configuring in global zone, 564, 591
configuring per-zone, 600-601
configuring queue parameters, 593-594
device allocation, 92-93
disabling, 595-596
enabling, 594-595
finding changes to specific files, 619
logins, 622
planning, 564-568
planning in zones, 564-565
preselection definition, 555
privileges and, 239
rights profiles for, 636-637
roles, 184
setting queue parameters, 592-594
setting -setqctrl options, 593-594
sftp file transfers, 622-624
troubleshooting, 612-624
troubleshooting praudit command, 610
updating information, 596-597
zones and, 560-561, 637

auditlog file, text audit records, 580

audit_record command
[] (square brackets) in output, 644
description, 627
displaying audit record formats, 602-604
example, 602
listing all formats, 602
listing formats of class, 603-604
listing formats of program, 603
optional tokens ([]), 644

audit_reduce command, 628
-c option, 608

cleaning up audit files, 610-611
description, 628
displays audit files, 604-606
elements, 604-606
filtering options, 606
merging audit records, 604-606
-o option, 604-606
options, 628
selecting audit records, 606-608
timestamp use, 643
trialer tokens, and, 654
authreduce command (Continued)
  using lowercase options, 606
  using uppercase options, 605
  without options, 628
auth_attr database
  description, 214-215
  summary, 211
AUTH_DES authentication, See AUTH_DH authentication
AUTH_DH authentication, and NFS, 291
authentication
  AUTH_DH client-server session, 293-296
  configuring cross-realm, 409-411
  description, 54-56
  DH authentication, 292-296
  disabling with -X option, 528
  Kerberos and, 359
  name services, 291
  network security, 54-56
  NFS-mounted files, 301
  overview of Kerberos, 542
  Secure RPC, 291
Solaris Secure Shell
  methods, 322-324
  process, 344-345
  terminology, 537-538
  types, 54-56
  use with NFS, 291
authentication methods
  GSS-API credentials in Solaris Secure Shell, 322
  host-based in Solaris Secure Shell, 323, 326-328
  keyboard-interactive in Solaris Secure Shell, 323
  password in Solaris Secure Shell, 323
  public keys in Solaris Secure Shell, 323
  Solaris Secure Shell, 322-324
authenticator
  in Kerberos, 537, 543
authlog file, saving failed login attempts, 64-65
authorizations
  Kerberos and, 359
  types, 54-56
authorizations (RBAC) (Continued)
  database, 211-218
  definition, 160
  delegating, 211
  description, 157, 210-211
  for allocating device, 89
  for device allocation, 99-100
  granularity, 210
  naming convention, 210
  not requiring for device allocation, 91
  solaris.device.allocate, 89, 99
  solaris.device.revoke, 99
authorized_keys file, description, 353
AuthorizedKeysFile keyword, sshd_config file, 347
auths command, description, 218
AUTHS_GRANTED keyword, policy.conf file, 217
auto_transition option, SASL and, 319
automatic login
  disabling, 528
  enabling, 527
automatically configuring
  Kerberos
    master KDC server, 389-390
    slave KDC server, 402-403
automating principal creation, 484-485
auxprop_login option, SASL and, 319

B
  -b option, authreduce command, 607
backup
  Kerberos database, 443-444
  slave KDCs, 380-381
Banner keyword, sshd_config file, 347
BART
  components, 106-108
  overview, 105-108
  programmatic output, 126
  security considerations, 109
  task map, 108-109
  verbose output, 126
  bart command, 105
  bart compare command, 107
  bart create command, 106-107, 109
Basic Audit Reporting Tool, See BART
basic privilege set, 168
Basic Security Module (BSM)
  See auditing
  See device allocation
Basic Solaris User rights profile, 208
Batchmode keyword, ssh_config file, 347
BindAddress keyword, ssh_config file, 347
binding control flag, PAM, 311
blowfish-cbc encryption algorithm, ssh_config file, 347
Blowfish encryption algorithm
  kernel provider, 264
  policy.conf file, 69
  ssh_config file, 347
  using for password, 69
Bourne shell, privileged version, 163
bsmconv script
  creating device_maps file, 100-101
  description, 636
  enabling audit service, 594-595
bsmunconv script, disabling audit service, 595-596

C
-C option, auditreduce command, 605
C shell, privileged version, 163
-c option
  auditrecord command, 603-604
  auditreduce command, 607, 608
c2audit:audit_load entry, system file, 632
c2audit module, verifying is loaded, 613
  cache, credential, 542
canon_user_plugin option, SASL and, 319
  caret (^) in audit class prefixes, 639
CD-ROM drives
  allocating, 95-96
  security, 103
cd rw command, authorizations required, 219
certificates
  exporting for use by another system, 285-286
  generating with pktool gencert command, 282-283
  importing into keystore, 283-285
ChallengeResponseAuthentication keyword, See
  KbdInteractiveAuthentication keyword
changing
  ACL entries, 146-147
  allocatable devices, 91-92
  audit_class file, 583-584
  audit_control file, 577-579
  audit_event file, 584-586
  default password algorithm, 67
  device policy, 85-86
  file ownership, 139
  file permissions
    absolute mode, 141-142
    special, 142-143
    symbolic mode, 140-141
  group ownership of file, 140
  NFS secret keys, 293
  passphrase for Solaris Secure Shell, 334-335
  password algorithm for a domain, 69
  password algorithm task map, 67
  password of role, 193-194
  properties of role, 195-196
  rights profile contents, 196-199
  rights profile from command line, 197
  root user into role, 184-188
  special file permissions, 142-143
  user properties from command line, 201
  your password with kpasswd, 522
  your password with passwd, 522
CheckHostIP keyword, ssh_config file, 347
chgrp command
  description, 128
  syntax, 140
chkey command, 293, 300
chmod command
  changing special permissions, 142-143, 143
  description, 128
  syntax, 142
choosing, your password, 521-522
chown command, description, 127
Cipher keyword, sshd_config file, 347
Ciphers keyword, Solaris Secure Shell, 347
classes, See audit classes
cleaning up, binary audit files, 610-611
Index

clear protection level, 528
ClearAllForwardings keyword, Solaris Secure Shell
port forwarding, 347
client names, planning for in Kerberos, 379-380
ClientAliveCountMax keyword, Solaris Secure Shell
port forwarding, 347
ClientAliveInterval keyword, Solaris Secure Shell
port forwarding, 347
clients
AUTH_DH client-server session, 293-296
configuring for Solaris Secure Shell, 344, 346
configuring Kerberos, 419-436
definition in Kerberos, 537
clnctconfig principal
creating, 395, 402
clock skew
Kerberos and, 437-438
Kerberos planning and, 382
clock synchronizing
Kerberos master KDC and, 396, 402
Kerberos planning and, 382
Kerberos slave KDC and, 408
Kerberos slave server and, 453
cmd audit token, 647
cnt audit policy, description, 569
combining audit files
auditreduce command, 604-606, 628
from different zones, 637
command execution, Solaris Secure Shell, 345
command-line equivalents of SEAM Administration
Tool, 480-481
commands
See also individual commands
ACL commands, 135-136
auditing commands, 626-631
cryptographic framework commands, 246
determining user’s privileged commands, 231-233
device allocation commands, 99
device policy commands, 97-98
file protection commands, 127
for administering privileges, 237
Kerberos, 535-536
RBAC administration commands, 218-219
Secure RPC commands, 293
commands (Continued)
Solaris Secure Shell commands, 354-356
that assign privileges, 170
that check for privileges, 161
user-level cryptographic commands, 247
common keys
calculating, 295
DH authentication and, 292-296
components
BART, 106-108
device allocation mechanism, 98-99
RBAC, 157-160
Solaris Secure Shell user session, 345
Compression keyword, Solaris Secure Shell, 347
CompressionLevel keyword, ssh_config file, 347
Computer Emergency Response Team/Coordination Center (CERT/CC), 57
computer security, See system security
computing
DH key, 299
digest of a file, 257-258
MAC of a file, 258-260
secret key, 250-252, 252-257
configuration decisions
auditing
file storage, 565-566
policy, 568-571
who and what to audit, 566-568
zones, 564-565
Kerberos
client and service principal names, 379-380
clients, 382-383
clock synchronization, 382
database propagation, 382
encryption types, 384
KDC server, 383-384
mapping host names onto realms, 379
number of realms, 378
ports, 380
realm hierarchy, 379
realm names, 378
realms, 378-379
slave KDCs, 380-381
password algorithm, 42
configuration files
  audit_class file, 632
  audit_control file, 577-579, 627, 632
  audit_event file, 634
  audit_user database, 634-635
  device_maps file, 100
  nsswitch.conf file, 40
  for password algorithms, 42
  policy.conf file, 42, 68-69, 218
  Solaris Secure Shell, 344
  syslog.conf file, 64-65, 239, 632
  system file, 632
  with privilege information, 238-239
configuring
  ahlt audit policy, 591
  audit_class file, 583-584
  audit_control file, 577-579
  audit_event file, 584-586
  audit files, 576-586
  audit files task map, 576
  audit policy, 590-592
  audit policy temporarily, 591
  audit queue parameters, 592-594
  audit service task map, 586-587
  audit trail overflow prevention, 611
  audit_user database, 581-583
  audit_warn script, 589-590
  auditconfig command, 631
  auditd queue parameters, 592-594
  auditd service policy, 590-592
  auditing, 625-626
  auditing in zones, 560-561, 637
  custom roles, 182
  device allocation, 87-88
  device policy, 84
  devices task map, 83
  DH key for NIS+ user, 298-299
  DH key for NIS user, 300-301
  DH key in NIS, 299-300
  DH key in NIS+, 297-298
  dial-up logins, 65
  exceptions to Solaris Secure Shell system defaults, 330-331
  hardware security, 74-76
configuring (Continued)
  host-based authentication for Solaris Secure Shell, 326-328
  identical auditing for non-global zones, 598-600
  Kerberos
    adding administration principals, 394, 401
    clients, 419-436
    cross-realm authentication, 409-411
    master KDC server, 389-390, 390-391, 391-396
    master KDC server using LDAP, 396-402
    NFS servers, 414-415
    overview, 387-461
    slave KDC server, 402-403, 403-404, 404-408
    task map, 387-388
  name service, 186
  password for hardware access, 74-75
  per-zone auditing, 600-601
  per-zone audit policy, 592
  port forwarding in Solaris Secure Shell, 329-330
  RBAC, 175-188
  RBAC task map, 174-175
  rights profile from command line, 197
  rights profiles, 196-199
  roles, 177-180, 195-196
    from command line, 180-182
  root user as role, 184-188
  Solaris Secure Shell, 325
    clients, 346
    servers, 346
  Solaris Secure Shell task map, 325
  textual audit logs, 579-581
configuring application servers, 411-413
  Connection Attempts keyword, ssh_config file, 347
  console, displaying su command attempts, 73-74
  CONSOLE in Solaris Secure Shell, 350
  Console User (RBAC), rights profile, 209
  CONSOLE_USER keyword, policy.conf file, 217
  consumers, definition in cryptographic framework, 244
  context-sensitive help, SEAM Administration Tool, 481
  control manifests (BART), 105
controlling
  access to system hardware, 74
controlling (Continued)
  system access, 59-60
  system usage, 47-51
conversation keys
  decrypting in secure RPC, 295
  generating in secure RPC, 294
converting
  audit records to readable format, 609, 629
copying
  ACL entries, 146
  files using Solaris Secure Shell, 339-340
  copying audit messages to single file, 608
  cost control, and auditing, 571
  crammd5.so.1 plug-in, SASL and, 318
creating
  audit trail
    auditd daemon, 642
    auditd daemon's role, 626-627
  credential table, 415-416
  customized role, 182
  d_passwd file, 66
  dial-up passwords, 65-67
  /etc/d_passwd file, 66
  file digests, 257-258
  local user, 185
  new device-cleanup scripts, 103-104
  new policy (Kerberos), 489, 501-502
  new principal (Kerberos), 489-492
  Operator role, 178-179
  partitions for binary audit files, 587-589
  passwords for temporary user, 66
  rights profiles, 196-199
  rights profiles with Solaris Management Console, 198
  roles
    for particular profiles, 177-180
    on command line, 180-182
    with limited scope, 179
  root user as role, 184-188
  secret keys
    for encryption, 250-252, 252-257
    security-related roles, 179
    Solaris Secure Shell keys, 332-334
    stash file, 408, 453
creating (Continued)
  System Administrator role, 178
  tickets with kinit, 518
cred database
  adding client credential, 298
  adding user credential, 298
  DH authentication, 292-296
cred table
  DH authentication and, 293
  information stored by server, 295
credentialed
  cache, 542
  description, 294, 537
  obtaining for a server, 543-544
  obtaining for a TGS, 542-543
  or tickets, 361
  credential table, adding single entry to, 416
  credentials, mapping, 381
crontab files, authorizations required, 219
cross-realm authentication, configuring, 409-411
CRYPT_ALGORITHMS_ALLOW keyword, policy.conf file, 43
CRYPT_ALGORITHMS_DEPRECATED keyword, policy.conf file, 43
crypt_bsdbf password algorithm, 42
crypt_crammd5 password algorithm, 42
crypt command, file security, 52
crypt.conf file
  changing with new password module, 70-71
  third-party password modules, 70-71
CRYPT_DEFAULT keyword, policy.conf file, 43
CRYPT_DEFAULT system variable, 68
crypt_sha256 password algorithm, 42
crypt_sunmd5 password algorithm, 42
crypt_unix password algorithm, 42, 68-71
Crypto Management (RBAC)
  creating role, 183-184
  use of rights profile, 269, 271
cryptoadm command
  description, 246
  disabling cryptographic mechanisms, 269, 271
  disabling hardware mechanisms, 274-276
  installing PKCS #11 library, 269
  listing providers, 264
cryptoadm command (Continued)
- m option, 269, 271
- p option, 269, 271
restoring kernel software provider, 271
cryptoadm install command, installing PKCS #11 library, 269
cryptographic framework
administering with role, 183-184
connecting providers, 247-248
consumers, 244
cryptoadm command, 246
definition of terms, 244
description, 243-244
elfsign command, 246, 247
error messages, 263
hardware plugins, 244
installing providers, 248
interacting with, 246
listing providers, 264-267
PKCS #11 library, 244
providers, 244
refreshing, 276-277
registering providers, 248
restarting, 276-277
signing providers, 248
task maps, 249
user-level commands, 247
zones and, 248, 276-277
cryptographic services, See cryptographic framework
Cryptoki, See PKCS #11 library
csh command, privileged version, 163
Custom Operator (RBAC), creating role, 182
customizing, manifests, 112-115
customizing a report (BART), 121-122
d_option
auditreduce command, 608
getfacl command, 148
praudit command, 630
setfacl command, 148
daemons
auditd, 626-627
crfd, 246
keyserv, 297
nsd (name service cache daemon), 178, 218
rpc.nispasswd, 69
running with privileges, 166
ssh-agent, 336-337
sshd, 343-345
table of Kerberos, 536
vold, 89
Data Encryption Standard, See DES encryption
data forwarding, Solaris Secure Shell, 345
databases
audit_user, 634-635
auth_attr, 214-215
backing up and propagating KDC, 443-444
creating KDC, 393
cred for Secure RPC, 293, 298
exec_attr, 216-217
KDC propagation, 382
NFS secret keys, 293
prof_attr, 215-216
publickey for Secure RPC, 293
RBAC, 211-218
user_attr, 213
with privilege information, 238-239
dd command, generating secret keys, 250-252
deallocate command
allocate error state, 100
authorizations for, 99
authorizations required, 220
description, 99
device-clean scripts and, 103-104
using, 96-97
deallocating
devices, 96-97
forcibly, 91
microphone, 96-97
debugging, privileges, 224
debugging sequence number, 652
decrypt command
description, 247
syntax, 262
decrypting
collection keys for Secure RPC, 295
files, 262
NFS secret keys, 293
secret keys, 293
default/login file, description, 353
default_realm section
  krb5.conf file, 392, 399
default_priv keyword, user_attr database, 238
defaults
  ACL entries for directories, 135
  praudit output format, 630
  privilege settings in policy.conf file, 238
  system-wide auditing, 637
  system-wide in policy.conf file, 42
  umask value, 131
delegating, RBAC authorizations, 211
delete_entry command, ktutil command, 514
deleting
  ACL entries, 135-136, 147-148
  archived audit files, 611
  audit files, 604
  host’s service, 514
  not_terminated audit files, 610-611
  policies (Kerberos), 504-505
  principal (Kerberos), 494
  rights profiles, 197
DenyGroups keyword, sshd_config file, 347
DenyUsers keyword, sshd_config file, 347
DES encryption
  kernel provider, 264
  Secure NFS, 292
destroying, tickets with kdestroy, 520
determining
  audit_control flags are correct, 614
  audit ID of a user, 620
  audit_user flags are correct, 614
  auditing is running, 612-615
  c2audit module is loaded, 613
determining (Continued)
  files with setuid permissions, 149
  if file has ACL, 144
  privileges on a process, 222-223
  privileges task map, 229
/dev/arp device, getting IP MIB-II information, 87
/dev/urandom device, 250-252
devfsadm command, description, 98
device_allocate file
description, 101-102
format, 101
sample, 91, 101
device allocation
  adding devices, 87-88
  allocatable devices, 102
  allocate command, 99
  allocate error state, 100
  allocating devices, 93-94
  auditing, 92-93
  authorizations for commands, 99-100
  authorizing users to allocate, 89
  changing allocatable devices, 91-92
  commands, 99
  components of mechanism, 98-99
  configuration file, 100
  deallocate command, 99
    device-clean scripts and, 103-104
    using, 96-97
  deallocating devices, 96-97
  device_allocate file, 101-102
  device-clean scripts
    audio devices, 103
    CD-ROM drives, 103
    description, 102-104
    diskette drives, 103
    options, 103
    tape drives, 102
    writing new scripts, 103-104
  device_maps file, 100-101
  disabling, 595
  enabling, 88-89
  examples, 94
  forcibly allocating devices, 90
  forcibly deallocating devices, 91
device allocation (Continued)
  making device allocatable, 88-89
  managing devices, 87-88
  mounting devices, 94-96
  not requiring authorization, 91
  preventing, 92
  requiring authorization, 91-92
  task map, 87-88
  troubleshooting, 94, 96
  troubleshooting permissions, 90
  unmounting allocated device, 97
  user procedures, 93
  using, 93
  using allocate command, 93-94
  viewing information, 90
device-clean scripts
  and object reuse, 102-104
  audio devices, 103
  CD-ROM drives, 103
  description, 102-104
  diskette drives, 103
  options, 103
  tape drives, 102
  writing new scripts, 103-104
device management, See device policy
device_maps file
  description, 100
  format, 100
  sample entries, 100
device policy
  add_drv command, 97
  auditing changes, 86
  changing, 85-86
  commands, 97
  configuring, 84-87
  kernel protection, 97-104
  managing devices, 84
  overview, 45-47
  removing from device, 86
  task map, 84
  update_drv command, 85-86, 97
  viewing, 84-85
Device Security (RBAC), creating role, 179
devices
  adding device policy, 85-86
  allocating for use, 93
  auditing allocation of, 92-93
  auditing policy changes, 86
  authorizing users to allocate, 89
  changing device policy, 85-86
  changing which are allocatable, 91-92
  deallocating a device, 96-97
  /dev/urandom device, 250-252
device allocation
  See device allocation
  forcibly allocating, 90
  forcibly deallocating, 91
  getting IP MIB-II information, 87
  listing, 84-85
  listing device names, 90
  login access control, 44
  making allocatable, 88-89
  managing, 84
  managing allocation of, 87-88
  mounting allocated devices, 94-96
  not requiring authorization for use, 91
  policy commands, 97-98
  preventing use of all, 92
  preventing use of some, 92
  privilege model and, 171
  protecting by device allocation, 46
  protecting in the kernel, 45
  removing policy, 86
  security, 45-47
  superuser model and, 171
  unmounting allocated device, 97
  viewing allocation information, 90
  viewing device policy, 84-85
  zones and, 46
dfs tab file, sharing files, 53
DH authentication
  configuring in NIS, 299-300
  configuring in NIS+, 297-298
  description, 292-296
  for NIS+ client, 298
  for NIS client, 299-300
  mounting files with, 301
DH authentication (Continued)
sharing files with, 301
DHCP Management (RBAC), creating role, 179
dial-up passwords
creating, 65-67
disabling, 45
disable temporarily, 67
/etc/d_passwd file, 45
security, 45
dialups file, creating, 65
Diffie-Hellman authentication, See DH authentication
digest command
description, 247
equation, 257
digestmdso.1 plug-in, SASL and, 318
digests
computing for file, 257-258
of files, 257-258
dir line, audit_control file, 633
direct realms, 410-411
directories
See also files
ACL entries, 135
audit_control file definitions, 633
audit directories full, 627, 636
auditd daemon pointer, 627
displaying files and related information, 127, 137-138
mounting audit directories, 642
permissions
defaults, 131
description, 128-129
public directories, 131
disabling
abort sequence, 75-76
audit policy, 590-592
audit service, 595-596
cryptographic mechanisms, 269
device allocation, 595
dial-up logins temporarily, 67
dial-up passwords, 67
executable stacks, 150-151
executables that compromise security, 136
disabling (Continued)
hardware mechanisms, 274-276
keyboard abort, 75-76
keyboard shutdown, 75-76
logging of executable stack messages, 151
logins temporarily, 62-63
programs from using executable stacks, 150-151
remote root access, 73-74
service on a host (Kerberos), 513-515
system abort sequence, 75-76
user logins, 62-63
disk partitioning, for binary audit files, 587-589
disk-space requirements, 572-573
diskette drives
allocating, 95
device-clean scripts, 103
displaying
ACL entries, 135-136, 144, 148-149
allocatable devices, 90
audit policies, 590
audit queue parameter values, 592
audit record formats, 602-604
audit records, 608-610
audit records in XML format, 610
device policy, 84-85
file information, 137-138
files and related information, 127
format of audit records, 602-604
providers in the cryptographic framework, 264-267
roles you can assume, 188, 218
root access attempts, 73-74
selected audit records, 604-606
su command attempts, 73-74
sublist of principals (Kerberos), 486
user's login status, 60-61, 61
users with no passwords, 61-62
dminfo command, 100
DNS, Kerberos and, 379-380
domain_realm section
krb5.conf file, 379, 392, 399
dot(.)
authorization name separator, 210
displaying hidden files, 137
double dollar sign ($$), parent shell process number, 222
DSAAuthentication keyword, See
PubkeyAuthentication keyword
DTD for praudit command, 630
duplicating, principals (Kerberos), 492
DynamicForward keyword, ssh_config file, 347

E
-e option
auditreduce command, 608
ppriv command, 224
ebusys string, audit_warn script, 636
eeprom command, 40, 74-76
effective privilege set, 168
efficiency, auditing and, 573
eject command, device cleanup and, 103
eftsign command
description, 246, 247
enabling
audit service, 594-595
audit service task map, 586-587
auditing, 594-595
cryptographic mechanisms, 270
device allocation, 88-89
Kerberized applications only, 459-460
kernel software provider use, 271
keyboard abort, 75-76
mechanisms and features on hardware provider, 275
encrypt command
description, 247
error messages, 263
syntax, 251
troubleshooting, 263
encrypting (Continued)
using user-level commands, 247
encryption
algorithms
Kerberos and, 384
DES algorithm, 292
generating symmetric key
using the dd command, 250-252
using the pktool command, 252-257
installing third-party password modules, 70-71
list of password algorithms, 42
modes
Kerberos and, 384
password algorithm, 42
privacy service, 359
specifying algorithms in ssh_config file, 347
specifying password algorithm
locally, 67
specifying password algorithms in policy.conf file, 42
types
Kerberos and, 384, 545-547
with -x option, 528
ending, signal received during auditing shutdown, 636
environment variables
See also variables
audit token for, 647-648
overriding proxy servers and ports, 340
PATH, 48
presence in audit records, 569, 645
Solaris Secure Shell and, 350-351
use with ssh-agent command, 355
equal sign (=), file permissions symbol, 132
error messages
encrypt command, 263
Kerberos, 463-475
with kpasswd, 522
errors
allocate error state, 100
audit directories full, 627, 636
internal errors, 636
EscapeChar keyword, ssh_config file, 347
/etc/d_passwd file
and /etc/passwd file, 45
Index

/etc/d_passwd file (Continued)
creating, 66
disabling dial-up logins temporarily, 67
/etc/default/kbd file, 75-76
/etc/default/login file
description, 353
login default settings, 64
restricting remote root access, 73-74
Solaris Secure Shell and, 350-351
/etc/default/su file
displaying su command attempts, 73-74
monitoring access attempts, 73-74
monitoring su command, 72-73
/etc/dfs/dfstab file, sharing files, 53
/etc/dialups file, creating, 65
/etc/hosts.equiv file, description, 353
/etc/krb5/kadm5.acl file, description, 533
/etc/krb5/kadm5.keytab file, description, 534
/etc/krb5/kdc.conf file, description, 534
/etc/krb5/kpropd.acl file, description, 534
/etc/krb5/krb5.conf file, description, 534
/etc/krb5/krb5.keytab file, description, 534
/etc/krb5/warn.conf file, description, 534
/etc/logindewperm file, 44
/etc/nologin file
description, 353
disabling user logins temporarily, 62-63
/etc/nsswitch.conf file, 40
/etc/pam.conf file, Kerberos and, 534
/etc/publickey file, DH authentication and, 293
/etc/security/audit_event file, audit events
and, 555
/etc/security/audit_warn script, 635
/etc/security/bsconv script, 100-101
description, 636
/etc/security/crypt.conf file
changing with new password module, 70-71
third-party password modules, 70-71
/etc/security/device_allocate file, 101
/etc/security/device_maps file, 100
/etc/security/policy.conf file, algorithms
configuration, 68-69
/etc/ssh_host_dsa_key.pub file, description, 352
/etc/ssh_host_key.pub file, description, 352
/etc/ssh_host_rsa_key.pub file, description, 352
/etc/ssh/hosts.equiv file, description, 353
/etc/ssh/ssh_config file
describing Solaris Secure Shell, 346
description, 354
host-specific parameters, 350
keywords, 346-351
override, 354
/etc/ssh/ssh_host_dsa_key file, description, 352
/etc/ssh/ssh_host_key file
description, 352
override, 354
/etc/ssh/ssh_host_rsa_key file, description, 352
/etc/ssh/ssh_known_hosts file
two controlling distribution, 351
description, 353
override, 354
secure distribution, 351
/etc/ssh/sshhd_config file
description, 352
keywords, 346-351
/etc/ssh/sshrc file, description, 354
/etc/syslog.conf file
auditing and, 580, 632
executable stack messages and, 136
failed logins and, 64-65
PAM and, 309
/etc/system file, 632
event, description, 555
eexec_args audit token
argv policy and, 647
format, 647
eexec_attr database
description, 216-217
summary, 211
eexec audit class, 638
eexec_env audit token, format, 647-648
executable stacks
disabling logging messages, 151
logging messages, 136
protecting against, 136, 150-151
execute permissions, symbolic mode, 132
export subcommand, pktool command, 285-286

684  System Administration Guide: Security Services  •  January 2010 (Beta)
EXTERNAL security mechanism plug-in, SASL and, 318

F
-f option
  Kerberized commands, 527, 529
  setfacl command, 146
  st_clean script, 104
-F option
  deallocate command, 100
  Kerberized commands, 527, 529
failed login attempts
  loginlog file, 63
  syslog.conf file, 64-65
failure
  audit class prefix, 639
  turning off audit classes for, 639
FallBackToRsh keyword, ssh_config file, 347
fd_clean script, description, 103
file_attr_acc audit class, 638
file_attr_mod audit class, 638
file audit token, format, 648
file_close audit class, 638
file_creation audit class, 638
file_deletion audit class, 638
file permission modes
  absolute mode, 132
  symbolic mode, 132
FILE privileges, 165
file_read audit class, 638
file systems
  adding a virus scan engine, 80
  enabling virus scanning, 80
  excluding files from virus scans, 81-82
  NFS, 291
  scanning for viruses, 79
security
  authentication and NFS, 291
  TMPFS file system, 131
sharing files, 53
 TMPFS, 131
file transfers, auditing, 622-624
file vnode audit token, 646
file_write audit class, 638
files
ACL entries
  adding or modifying, 146-147
  checking, 144
  deleting, 135-136, 147-148
  displaying, 135-136, 148-149
  setting, 144-146
  valid entries, 134-135
auditing modifications to, 619
BART manifests, 123
changing ACL, 146-147
changing group ownership, 140
changing ownership, 127, 139
changing special file permissions, 142-143
computing a digest, 257-258
computing digests of, 257-258
computing MAC of, 258-260
copying ACL entries, 146
copying with Solaris Secure Shell, 339-340
decrypting, 262
deleting ACL, 147-148
determining if has ACL, 144
digest of, 257-258
displaying ACL entries, 148-149
displaying file information, 137-138
displaying hidden files, 137
displaying information about, 127
decrypting, 250, 260-263
file types, 128
finding files with setuid permissions, 149
for administering Solaris Secure Shell, 352
hashing, 250
kdc.conf, 539
Kerberos, 533-535
manifests (BART), 123
mounting with DH authentication, 301
ownership
  and setgid permission, 130
  and setuid permission, 130
permissions
  absolute mode, 132, 141-142
  changing, 128, 131-133, 141
defaults, 131
files, permissions (Continued)
  description, 128-129
  setgid, 130
  setuid, 130
  sticky bit, 130-131
  symbolic mode, 132, 140-141, 141
  umask value, 131
PKCS #12, 285
privileges relating to, 165
protecting with ACLs, 143-149
protecting with UNIX permissions, 137-143
public objects, 555
security
  access restriction, 49
  ACL, 52
  changing ownership, 139
  changing permissions, 131-133, 141
  directory permissions, 128-129
  displaying file information, 127, 138
  encryption, 52, 250
  file permissions, 128-129
  file types, 128
  special file permissions, 133
  umask default, 131
  UNIX permissions, 127-133
  user classes, 128
setting ACL, 144-146
sharing with DH authentication, 301
special files, 129-131
symbols of file type, 128
syslog.conf file, 632
verifying integrity with digest, 257-258
with privilege information, 238-239
find command, finding files with setuid permissions, 149
firewall systems
  connecting from outside, 341
  outside connections with Solaris Secure Shell from command line, 341
  from configuration file, 340-341
  packet smashing, 57
  packet transfers, 57
  secure host connections, 340
  security, 56-57
firewall systems (Continued)
  trusted hosts, 56
flags line
  audit_control file, 632
  process preselection mask, 641
fMRI audit token, format, 648
forced cleanup, st_clean script, 104
format of audit records, audit_record command, 602
forwardable tickets
  definition, 538
  description, 360
  example, 518
  with -F option, 527, 529
  with -f option, 527, 529
ForwardAgent keyword, Solaris Secure Shell forwarded authentication, 347
ForwardX11 keyword, Solaris Secure Shell port forwarding, 347
FQDN (Fully Qualified Domain Name), in Kerberos, 379-380
ftp command
  Kerberos and, 526-529, 535
  logging file transfers, 622-624
  setting protection level in, 528
ftpd daemon, Kerberos and, 536
G
GatewayPorts keyword, Solaris Secure Shell, 347
gateways, See firewall systems
gencert subcommand, pktool command, 282-283
generating certificates with pktool command, 282-283
keys for Solaris Secure Shell, 332-334
NFS secret keys, 293
passphrases with pktool command, 286-287
random number
  using the dd command, 250-252
  using the pktool command, 252-257
Solaris Secure Shell keys, 332-334
symmetric key
  using the dd command, 250-252
  using the pktool command, 252-257
Generic Security Service API, See GSS-API
getdevpolicy command, description, 98
getfacl command
  -a option, 148
  -d option, 148
description, 135-136
displaying ACL entries, 148-149
elements, 148-149
verifying ACL entries, 145
getting
access to a specific service, 544-545
credential for a server, 543-544
credential for a TGS, 542-543
gadmin command
  See also SEAM Administration Tool
description, 535
.gadmin file
description, 533
SEAM Administration Tool and, 481
GlobalKnownHostsFile keyword, ssh_config file, 347
GlobalKnownHostsFile2 keyword, See
  GlobalKnownHostsFile keyword
granting access to your account, 524-526
group ACL entries
default entries for directories, 135
description, 134-135
setting, 144-146
group audit policy
  and groups token, 569, 649
description, 569
group audit token, replaced by groups token, 649
group ID numbers (GIDs), special logins and, 44
groups
  changing file ownership, 140
  exceptions to Solaris Secure Shell defaults, 330-331
  groups audit token, 649
GSS-API
  authentication in Solaris Secure Shell, 322
credentials in secure RPC, 297-298
credentials in Solaris Secure Shell, 344
  Kerberos and, 360, 374
gssapi.so.1 plug-in, SASL and, 318
GSSAPIAuthentication keyword, Solaris Secure
Shell, 347
GSSAPIDelegateCredentials keyword, Solaris Secure
Shell, 348
GSSAPIKeyExchange keyword, Solaris Secure Shell, 348
GSSAPIStoreDelegatedCredentials keyword,
  ssh_config file, 348
gsscred command, description, 535
gsscred table, using, 547
gssd daemon, Kerberos and, 536
H
  -h option, auditrecord command, 602
  hard disk, space requirements for auditing, 572-573
  hard string, audit_warn script, 635
hardware
  listing attached hardware accelerators, 274
  protecting, 40, 74-76
  requiring password for access, 74-75
hardware providers
  disabling cryptographic mechanisms, 274-276
  enabling mechanisms and features on, 275
  listing, 274
  loading, 274
hash
  algorithms
    Kerberos and, 384
  hashing, files, 250
  header audit token
    format, 649
    order in audit record, 649
help
  SEAM Administration Tool, 481-482
  URL for online, 385
Help Contents, SEAM Administration Tool, 482
hierarchical realms
  configuring, 409-410
  in Kerberos, 365, 379
hmac-md5 algorithm, ssh_config file, 348
hmac-sha1 encryption algorithm, ssh_config file, 348
host-based authentication
  configuring in Solaris Secure Shell, 326-328
description, 322
Host keyword
  ssh_config file, 348, 350
host names, mapping onto realms, 379
host principal
    creating, 395,401
HostbasedAuthentication keyword, Solaris Secure Shell, 348
HostbasedUsesNamesFromPacketOnly keyword, sshd_config file, 348
HostKey keyword, sshd_config file, 348
HostKeyAlgorithms keyword, ssh_config file, 348
HostKeyAlias keyword, ssh_config file, 348
hosts
    disabling Kerberos service on, 513-515
    exceptions to Solaris Secure Shell defaults, 330-331
    Solaris Secure Shell hosts, 322
    trusted hosts, 56
hosts.equiv file, description, 353

I
    -I option
        bart create command, 110
        st_clean script, 103
    -i option
        bart create command, 110, 115
        encrypt command, 261
        st_clean script, 103
identity files (Solaris Secure Shell), naming conventions, 352
IdentityFile keyword, ssh_config file, 348
IDs
    audit
        mechanism, 641
        overview, 551-553
        audit session, 641
        mapping UNIX to Kerberos principals, 547
IgnoreRhosts keyword, sshd_config file, 348
IgnoreUserKnownHosts keyword, sshd_config file, 348
import subcommand, pktool command, 283-285
in_addr audit token, format, 649
in.ftpd daemon, Kerberos and, 536
in.rlogind daemon, Kerberos and, 536
in.rshd daemon, Kerberos and, 536
in.telnetd daemon, Kerberos and, 536
include control flag, PAM, 311
inheritable privilege set, 168
initial ticket, definition, 538
install subcommand, cryptoadm command, 269
installing
    password encryption module, 70-71
    providers in cryptographic framework, 248
    Secure by Default, 50
instance, in principal names, 364-365
integrity
    Kerberos and, 359
    security service, 367
interactively configuring
    Kerberos
        master KDC server, 390-391
        slave KDC server, 403-404
INTERNAL plug-in, SASL and, 318
Internet firewall setup, 56-57
Internet-related tokens
    in_addr token, 649
    iport token, 650
    socket token, 652-653
invalid ticket, definition, 538
ioctl auditclass, 638
ioctl() system calls, 638
    AUDIO_SETINFO(), 103
IP addresses
    exceptions to Solaris Secure Shell defaults, 330-331
    Solaris Secure Shell checking, 347
IP MIB-II, getting information from /dev/arp, 87
ipc audit class, 638
i pc audit token, 649-650
    format, 649-650
i pc_pern audit token, format, 650
IPC privileges, 166
i pc type field values (i pc token), 649-650
i port audit token, format, 650

J
    JASS toolkit, pointer to, 50
K

-k option
  encrypt command, 261
  Kerberized commands, 528
  mac command, 259
-K option
  encrypt command, 261
  mac command, 259
  Kerberized commands, 528
  usermod command, 226
.k5.REALM file, description, 534
.k5login file
description, 524-526, 533
rather than revealing password, 525

kadm5.acl file
description, 533
format of entries, 496
master KDC entry, 394, 400, 441
new principals and, 490, 492

kadm5.keytab file, description, 534

kadmin command
  creating host principal, 395, 401
description, 535
ktadd command, 510-511
ktrmcmd command, 512
removing principals from keytab with, 512
SEAM Administration Tool and, 479

kadmin.local command
  adding administration principals, 394, 401
  automating creation of principals, 484
description, 535

kadmin.log file, description, 534

kadmind daemon
  Kerberos and, 536
  master KDC and, 536

kdb, 75-76

KbdInteractiveAuthentication keyword, Solaris Secure Shell, 348

kcfd daemon, 276-277

kclient command, description, 535

kdb5_ldapUtil command, description, 535

kdb5_util command (Continued)
description, 535

KDC
  backing up and propagating, 443-444
  configuring master
    automatic, 389-390
    interactive, 390-391
    manual, 391-396
    with LDAP, 396-402
  configuring slave
    automatic, 402-403
    interactive, 403-404
    manual, 404-408
  copying administration files from slave to master, 406, 451
  creating database, 393
  creating host principal, 395, 401
  database propagation, 382
  master
    definition, 536
    planning, 380-381
    ports, 380
    restricting access to servers, 460-461
  slave, 380-381
    definition, 536
    slave or master, 366, 388
    starting daemon, 408, 453
    swapping master and slave, 438-442
    synchronizing clocks
      master KDC, 396, 402
      slave KDC, 408, 453

kdc.conf file
description, 534
ticket lifetime and, 539

kdc.log file, description, 534

kdcmgr command
  configuring master
    automatic, 390
    interactive, 390
  configuring slave
    automatic, 403
    interactive, 403
  server status, 391
Index

kdestroy command
  example, 520
  Kerberos and, 535
KeepAlive keyword, Solaris Secure Shell, 348
Kerberos
  administering, 479-515
  Administration Tool
    See SEAM Administration Tool commands, 526-532, 535-536
  components of, 368-369
  configuration decisions, 377-385
  configuring KDC servers, 388-408
  daemons, 536
  enabling Kerberized applications only, 459-460
  encryption types
    overview, 384
    using, 545-547
  error messages, 463-475
  examples of using Kerberized commands, 530-532
  files, 533-535
  gaining access to server, 542-545
  granting access to your account, 524-526
  Kerberos V5 protocol, 359
  online help, 385
  options to Kerberized commands, 527
  overview
    authentication system, 360-366, 542
    Kerberized commands, 526-529
    password management, 521-526
  planning for, 377-385
  realms
    See realms (Kerberos)
    reference, 533-548
  remote applications, 364
  table of network command options, 528
  terminology, 536-541
  troubleshooting, 475
  using, 517-532
  Kerberos authentication, and Secure RPC, 292
  Kerberos commands, 526-532
    enabling only Kerberized, 459-460
    examples, 530-532
  kern.notice entry, syslog.conf file, 136
  kernel providers, listing, 264
  Key Distribution Center, See KDC
  key management framework (KMF), See KMF
  KEYBOARD_ABORT system variable, 75-76
  keylogin command
    use for Secure RPC, 293
    verifying DH authentication setup, 298
  KeyRegenerationInterval keyword, sshd_config file, 348
  keys
    creating DH key for NIS user, 300-301
    creating for Solaris Secure Shell, 332-334
    definition in Kerberos, 537
    generating for Solaris Secure Shell, 332-334
    generating symmetric key
      using the dd command, 250-252
      using the pktool command, 252-257
    service key, 509-515
    session keys
      Kerberos authentication and, 542
      using for MAC, 259
  keyserv daemon, 297
  keyserver
    description, 293
    starting, 297
  keystores
    exporting certificates, 285-286
    importing certificates, 283-285
    listing contents, 282
    managed by KMF, 280
    protecting with password in KMF, 286-287
  keytab file
    adding master KDC's host principal to, 395, 402
    adding service principal to, 509, 510-511
    administering, 509-515
    administering with ktutil command, 510
    disabling a host's service with delete_entry command, 514
    read into keytab buffer with read_kt command, 513, 514
    removing principals with kt remove command, 512
    removing service principal from, 512
    viewing contents with ktutil command, 512, 513
    viewing keylist buffer with list command, 513, 514
    keytab option, SASL and, 319
See also specific keyword
attribute in BART, 124
command-line overrides in Solaris Secure Shell, 355
Solaris Secure Shell, 346-351
kgcmsg command, description, 535
kinit command
example, 518
-F option, 518
Kerberos and, 535
ticket lifetime, 539
klist command
example, 519-520
-f option, 519-520
Kerberos and, 535
KMF
adding plugin, 287-288
creating
passphrases for keystores, 281
password for keystore, 286-287
self-signed certificate, 282-283
exporting certificates, 285-286
importing certificates into keystore, 283-285
library, 280
listing plugins, 287-288
managing
keystores, 281
PKI policy, 280
plugins, 280-281
public key technologies (PKI), 279
removing plugin, 287-288
utilities, 280
kmfcfg command
list plugin subcommand, 287-288
plugin subcommands, 279, 280-281
known hosts file
controlling distribution, 351
description, 353
Korn shell, privileged version, 163
kpasswd command
error message, 522
example, 523
Kerberos and, 535
passwd command and, 522
kprop command, description, 535
kpropd.acl file, description, 534
kpropd daemon, Kerberos and, 536
kproplog command, description, 535
krb5.conf file
description, 534
domain_realm section, 379
editing, 392, 399
ports definition, 380
krb5.keytab file, description, 534
krb5cc_uid file, description, 534
krb5kdc daemon
Kerberos and, 536
master KDC and, 536
starting, 408, 453
ksh command, privileged version, 163
ktadd command
adding service principal, 509, 510-511
syntax, 511
ktkt_warn daemon, Kerberos and, 536
ktremove command, 512
ktutil command
administering keytab file, 510
delete_entry command, 514
Kerberos and, 535
list command, 513, 514
read_kt command, 513, 514
viewing list of principals, 512, 513

L
-L option, ssh command, 337-339
-l option
digest command, 257
encrypt command, 251
mac command, 258
praudit command, 630
LDAP configuring master KDC using, 396-402
LDAP name service
passwords, 41
specifying password algorithm, 70
least privilege, principle of, 165
libraries, user-level providers, 264
lifetime of ticket, in Kerberos, 539-540
logging in (Continued)
  disabling temporarily, 62-63
  displaying user's login status, 60-61, 61
  log of failed logins, 64-65
  monitoring failures, 63
  root login
    account, 44
    restricting to console, 73-74
    tracking, 47-48
  security
    access control on devices, 44
    access restrictions, 40
    saving failed attempts, 63
    system access control, 40
    tracking root login, 47-48
  system logins, 44
    task map, 60
    users with no passwords, 61-62
LocalForward keyword, sshd_config file, 348
log files
  audit records, 558, 609
  BART
    programmatic output, 125-126
    verbose output, 125-126
  configuring for audit service, 579-581
  examining audit records, 628
  failed login attempts, 64-65
  monitoring su command, 72-73
  space for audit records, 627
  syslog audit records, 632
  /var/adm/messages, 615
  /var/log/syslog, 615
log_level option, SASL and, 319
logadm command, archiving textual audit files, 611
logging, ftp file transfers, 622-624
logging in
  and AUTH_DH, 293
  auditing logins, 622
 芽
machine security, See system security
MACS keyword, Solaris Secure Shell, 348
mail, using with Solaris Secure Shell, 338
makedbm command, description, 218
managing
See also administering
audit files, 604-606, 611
audit records task map, 601-602
audit trail overflow, 611
auditing, 575
auditing in zones, 560-561, 637
device allocation task map, 87-88
devices, 87-88
file permissions, 137
keystores with KMF, 281
passwords with Kerberos, 521-526
privileges task map, 221
RBAC task map, 192
manifests
See also bart create
customizing, 112-115
file format, 123
test, 107
manually configuring
Kerberos
master KDC server, 391-396
master KDC server using LDAP, 396-402
slave KDC server, 404-408
mapping
host names onto realms (Kerberos), 379
UIDs to Kerberos principals, 547
mapping GSS credentials, 381
mappings, events to classes (auditing), 557
mask (auditing)
description of process preselection, 641
system-wide process preselection, 632
mask ACL entries
default entries for directories, 135
description, 134-135
setting, 144-146
master KDC
automatically configuring, 389-390
configuring with LDAP, 396-402
master KDC (Continued)
definition, 536
interactively configuring, 390-391
manually configuring, 391-396
slave KDCs and, 366, 388
swapping with slave KDC, 438-442
match blocks, exceptions to Solaris Secure Shell
defaults, 330-331
max_life value, description, 539
max_renewable_life value, description, 540
maxAuthTries keyword, sshd_config file, 349
maxAuthTriesLog keyword, sshd_config file, 349
MaxStartups keyword, sshd_config file, 349
MD5 encryption algorithm
kernel provider, 264
policy.conf file, 68-69
mech_dh mechanism
GSS-API credentials, 345
secure RPC, 297-298
mech_krb mechanism, GSS-API credentials, 345
mech_list option, SASL and, 319
mechanism, definition in cryptographic
framework, 245
mechanisms
disabling all on hardware provider, 274-276
enabling some on hardware provider, 275
merging, binary audit records, 604-606
message authentication code (MAC), computing for
file, 258-260
messages file, executable stack messages, 136
metaslot
administering, 246
definition in cryptographic framework, 245
microphone
allocating, 93-94
deallocating, 96-97
minfree line
audit_control file, 633
audit_warn condition, 635
minus sign (-)
audit class prefix, 639
entry in sulog file, 72
file permissions symbol, 132
symbol of file type, 128
Index

mode, definition in cryptographic framework, 245
modifying
  policies (Kerberos), 503-504
  principal's password (Kerberos), 493
  principals (Kerberos), 493
  role assignment to a user, 179
  roles (RBAC), 195-196
  users (RBAC), 200-201
modules, password encryption, 42
monitoring
  audit trail in real time, 573
  failed logins, 63
  su command attempts, 47-48, 72-73
  superuser access attempts, 73-74
  superuser task map, 71-72
  system usage, 51
  use of privileged commands, 184
mount command, with security attributes, 89
mounting
  allocated CD-ROM, 95-96
  allocated devices, 94-96
  allocated diskette, 95
  audit directories, 642
  files with DH authentication, 301
mt command, tape device cleanup and, 103

naming conventions
  audit directories, 578, 633
  audit files, 642
  devices, 90
  RBAC authorizations, 210
  Solaris Secure Shell identity files, 352
ncp driver
  hardware plugin to cryptographic framework, 244
  listing mechanisms, 274
NET privileges, 166
netservices limited installation option, 50
network, privileges relating to, 166
network audit class, 639
network security
  authentication, 54-56
  authorizations, 54-56
  controlling access, 53-57
  firewall systems
    need for, 56
    packet smashing, 57
    trusted hosts, 56
  overview, 54
  reporting problems, 57
Network Security (RBAC), creating role, 179
Network Time Protocol, See NTP
never-audit classes, audit_user database, 634
new features
  BART, 105-126
commands
  bart compare, 107
  bart create, 106-107
  cryptoadm, 264
  decrypt, 262
  digest, 257-258
  encrypt, 260-263
  getdevpolicy, 84-85
  kcfid, 276-277
  kclient, 371
  kpropd, 371
  mac, 258-260
  ppriv, 222-223
  praudit -x, 609
  ssh-keyscan, 355
  ssh-keysign, 355
new features (Continued)
cryptographic framework, 243-248
device policy, 46-47
Kerberos enhancements, 371-373
PAM enhancements, 305-306
privileges, 164-172
process rights management, 164-172
SASL, 317
Solaris Cryptographic Framework, 243-248
Solaris Secure Shell enhancements, 324-325
strong password encryption, 42
newkey command
creating key for NIS user, 300-301
generating keys, 293
NFS file systems
authentication, 291
providing client-server security, 293-296
secure access with AUTH_DH, 301
NFS servers, configuring for Kerberos, 414-415
NIS+ name service
adding authenticated user, 299
authentication, 291
cred database, 298
cred table, 293
passwords, 41
specifying password algorithm, 69
NIS name service
authentication, 291
passwords, 41
specifying password algorithm, 69
nsswitch command
adding client credential, 298
generating keys, 293
no_class audit class, 638
nobody user, 53
noexec_user_stack_log variable, 136, 151
noexec_user_stack_variable, 136, 151
NoHostAuthenticationForLocalHost keyword, ssh_config file, 349
nologin file, description, 353
non_attrib audit class, 638
non-attributable audit mask, auditd daemon’s role, 626-627
nonattributable classes, 633
nonhierarchical realms, in Kerberos, 365
nscd (name service cache daemon)
starting with svcadm command, 178
use, 218
NSS, managing keystore, 281
nsswitch.conf file, login access restrictions, 40
NTP
Kerberos planning and, 382
master KDC and, 396, 402
slave KDC and, 408, 453
null audit class, 638
NumberOfPasswordPrompts keyword, ssh_config file, 349
O
-0 option, audit reduce command, 604-606
-o option, encrypt command, 261
object reuse requirements
device-clean scripts
tape drives, 102
writing new scripts, 103-104
for devices, 102-104
obtaining
access to a specific service, 544-545
credential for a server, 543-544
credential for a TGS, 542-543
forwardable tickets, 518
privileged commands, 195-196
privileges, 169, 170, 226-227
privileges on a process, 222-223
tickets with kinit, 518
online help
SEAM Administration Tool, 481-482
URL for, 385
OpenSSH, See Solaris Secure Shell
OpenSSL, managing keystore, 281
Operator (RBAC)
contents of rights profile, 207
creating role, 178-179
recommended role, 156
optional control flag, PAM, 311
options to Kerberized commands, 527
other ACL entries, description, 134-135
other audit class, 639
overflow prevention, audit trail, 611
ovsec_adm.xxxxx file, description, 534

ownership of files
ACLs and, 52, 133-136
changing, 127, 139
changing group ownership, 140

P
p_minfree attribute, audit_warn condition, 635
-p option
auditrecord command, 603
bart create, 115
cryptoadm command, 269, 271
logins command, 62
packages, Solaris Secure Shell, 352
packet transfers
firewall security, 56
packet smashing, 57
PAM
adding a module, 308
configuration file
control flags, 311
introduction, 309
stacking diagrams, 312
stacking example, 314
stacking explained, 310
syntax, 310
/etc/syslog.conf file, 309
framework, 304
Kerberos and, 369, 373
overview, 303
planning, 307
task map, 306
pam.conf file
See PAM configuration file
Kerberos and, 534
pam_roles command, description, 218
PAMAuthenticationViaKBDInt keyword, sshd_config file, 349
panels, table of SEAM Administration Tool, 505-508
passphrases
changing for Solaris Secure Shell, 334-335

passphrases (Continued)
encrypt command, 260
example, 335
generating in KMF, 286-287
mac command, 258
storing safely, 262
using for MAC, 259
using in Solaris Secure Shell, 334, 336-337
PASSREQ in Solaris Secure Shell, 350
passwd command
and kpasswd command, 522
and name services, 41
changing password of role, 193-194
passwd file, and /etc/d_passwd file, 45
password authentication, Solaris Secure Shell, 322
PasswordAuthentication keyword, Solaris Secure Shell, 349
passwords
authentication in Solaris Secure Shell, 322
changing role password, 193-194
changing with kpasswd command, 522
changing with passwd -r command, 41
changing with passwd command, 522
creating for dial-up, 65-67
dial-up passwords
disabling temporarily, 67
/etc/d_passwd file, 45
disabling dial-up temporarily, 67
displaying users with no passwords, 62
eliminating in Solaris Secure Shell, 336-337
encryption algorithms, 42
finding users with no passwords, 61-62
granting access without revealing, 524-526
hardware access and, 74-75
installing third-party encryption module, 70-71
LDAP, 41
specifying new password algorithm, 70
local, 41
login security, 40, 41
managing, 521-526
modifying a principal’s password, 493
NIS, 41
specifying new password algorithm, 69
NIS+, 41
passwords, NIS+ (Continued)
   specifying new password algorithm, 69
   policies and, 522
   PROM security mode, 40, 74-76
   protecting
   keystore, 285
   PKCS #12 file, 285
   requiring for hardware access, 74-75
   secret-key decryption for Secure RPC, 293
   specifying algorithm, 68-69
   in name services, 69
   locally, 67
   suggestions on choosing, 521-522
   system logins, 41, 44
   task map, 60
   UNIX and Kerberos, 521-526
   using Blowfish encryption algorithm for, 69
   using MD5 encryption algorithm for, 68-69
   using new algorithm, 68
   path_attr audit token, 651
   path audit policy, description, 569
   path audit token, format, 651
   PATH environment variable
   and security, 48
   setting, 49
   PATH in Solaris Secure Shell, 351
   permissions
   ACLs and, 52, 133-136
   changing file permissions
   absolute mode, 132, 141-142
   chmod command, 128
   symbolic mode, 132, 140-141, 141
   defaults, 131
   directory permissions, 128-129
   file permissions
   absolute mode, 132, 141-142
   changing, 131-133, 141
   description, 128-129
   special permissions, 131, 133
   symbolic mode, 132, 140-141, 141
   finding files with setuid permissions, 149
   setgid permissions
   absolute mode, 133, 143
   description, 130
   permissions, setgid permissions (Continued)
   symbolic mode, 132
   setuid permissions
   absolute mode, 133, 143
   description, 130
   security risks, 130
   symbolic mode, 132
   special file permissions, 129-131, 131, 133
   sticky bit, 130-131
   umask value, 131
   user classes and, 128
   PermitEmptyPasswords keyword, sshd_config file, 349
   PermitRootLogin keyword, sshd_config file, 349
   permitted privilege set, 168
   PermitUserEnvironment keyword, sshd_config file, 349
   perzone audit policy
   description, 569
   setting, 592
   using, 565, 600-601, 637
   when to use, 560-561
   pfch command, description, 163
   pfexec command, description, 218
   pfksh command, description, 163
   pfsh command, description, 163
   physical security, description, 40
   PKCS #11 library
   adding as provider, 268-269
   in Solaris Cryptographic Framework, 244
   PKCS #11 soft tokens, managing keystore, 281
   PKCS #12 files, protecting, 285
   pkcs11_kernel.so user-level provider, 264
   pkcs11_softtoken.so user-level provider, 264
   pkgadd command
   installing third-party providers, 268
   installing third-party software, 71
   PKI
   managed by KMF, 279
   policy managed by KMF, 280
   pktool command
   creating self-signed certificate, 282-283
   export subcommand, 285-286
   gencert subcommand, 282-283
pktool command (Continued)
generating secret keys, 252-257
import subcommand, 283-285
list subcommand, 282
managing PKI objects, 279
setpin subcommand, 286-287
plain.so.1 plug-in, SASL and, 318
planning
auditing, 564-568
auditing in zones, 564-565
auditing task map, 563
Kerberos
client and service principal names, 379-380
clock synchronization, 382
configuration decisions, 377-385
database propagation, 382
number of realms, 378
ports, 380
realm hierarchy, 379
realm names, 378
realms, 378-379
slave KDCs, 380-381
PAM, 307
RBAC, 175-177
pluggable authentication module, See PAM
plugin line
audit_control file, 633
p_* attributes, 579
qsize attribute, 579
plugin_list option, SASL and, 319
plugins
adding to KMF, 287-288
in audit service, 579
in cryptographic framework, 244
loaded by auditd daemon, 626-627
managed in KMF, 280-281
removing from KMF, 287-288
SASL and, 318
plus sign (+)
ACL entry, 144
audit class prefix, 639
entry in sulog file, 72
file permissions symbol, 132
policies
administering, 479-515
creating (Kerberos), 489
creating new (Kerberos), 501-502
deleting, 504-505
for auditing, 568-571
modifying, 503-504
on devices, 84-85
overview, 35-36
passwords and, 522
SEAM Administration Tool panels for, 505-508
specifying password algorithm, 67
task map for administering, 497
viewing attributes, 499-501
viewing list of, 497-499
policy
definition in cryptographic framework, 245
definition in Solaris OS, 35-36
policy.conf file
adding password encryption module, 70-71
Basic Solaris User rights profile, 208
description, 217-218, 218
keywords
for password algorithms, 43
for privileges, 217, 238
for RBAC authorizations, 217
for rights profiles, 217
for workstation owner, 217
specifying encryption algorithms in, 68-69
specifying password algorithm
in name services, 69
specifying password algorithms, 68-69
port forwarding
configuring in Solaris Secure Shell, 329-330
Solaris Secure Shell, 338, 339
Port keyword, Solaris Secure Shell, 349
ports, for Kerberos KDC, 380
postdated ticket
definition, 538
description, 360
postsigterm string, audit_warn script, 636
pound sign (#)
device_allocate file, 101
device_maps file, 100
ppriv command
for debugging, 224
listing privileges, 222
praudit command
converting audit records to readable format, 609, 629
DTD for -x option, 630
options, 629
output formats, 629
piping audit reduce output to, 609
use in a script, 630-631
viewing audit records, 608-610
with no options, 630
XML format, 610
PreferredAuthentications keyword, ssh_config file, 349
prefixes for audit classes, 639
preselecting, audit classes, 577-579
preselection in auditing, 555
preselection mask (auditing)
description, 641
reducing storage costs, 627
system-wide, 632
preventing
access to system hardware, 74
audit trail overflow, 611
executables from compromising security, 136
kernel software provider use, 271-273
use of hardware mechanism, 274-276
primary, in principal names, 364-365
Primary Administrator (RBAC)
assuming role, 189
recommended role, 156
rights profile contents, 206
primary audit directory, 633
principal
adding administration, 394, 401
adding service principal to keytab, 509, 510-511
administering, 479-515
automating creation of, 484-485
creating, 489-492
creating clntconfig, 395, 402
creating host, 395, 401
deleting, 494
principal (Continued)
duplicating, 492
Kerberos, 364-365
modifying, 493
principal name, 364-365
removing from keytab file, 512
removing service principal from keytab, 512
SEAM Administration Tool panels for, 505-508
service principal, 365
setting up defaults, 494-495
task map for administering, 483-484
user ID comparison, 415-416
user principal, 365
viewing attributes, 487-489
viewing list of, 485-487
viewing sublist of principals, 486
principal file, description, 534
principal.kadm5 file, description, 534
principal.kadm5.lock file, description, 534
principal.ok file, description, 534
principal.ulog file, description, 534
principle of least privilege, 165
Printer Management rights profile, 207
printing, audit log, 609
PrintMotd keyword, sshd_config file, 349
priv.debug entry, syslog.conf file, 239
PRIV_DEFAULT keyword
policy.conf file, 217, 238
PRIV_LIMIT keyword
policy.conf file, 217, 238
PRIV_PROC_LOCK_MEMORY privilege, 167-168
privacy
availability, 528
Kerberos and, 359
security service, 367
private keys
See also secret keys
definition in Kerberos, 537
Solaris Secure Shell identity files, 352
private protection level, 528
privilege audit token, 651
privilege checking, in applications, 161
privilege sets
adding privileges to, 170
privilege sets (Continued)
   basic, 168
   effective, 168
   inheritable, 168
   limit, 168
   listing, 169
   permitted, 168
   removing privileges from, 171
privileged application
   authorization checking, 161
   description, 158
   ID checking, 160
   privilege checking, 161
privileged ports, alternative to Secure RPC, 55
privileges
   adding to command, 225-226
   administering, 221
   assigning to a command, 170
   assigning to a script, 171
   assigning to a user, 170
   assigning to user or role, 226-227
   auditing and, 239
   categories, 165
   commands, 237
   compared to superuser model, 164-172
   debugging, 172, 224
   description, 157, 165
   determining directly assigned ones, 230-231
   devices and, 171
   differences from superuser model, 166
   effects on SEAM Administration Tool, 508
   escalation, 239
   executing commands with privilege, 170
   files, 238-239
   finding missing, 224-225
   how to use, 229
   implemented in sets, 168
   inherited by processes, 169
   limiting use by user or role, 227-228
   listing on a process, 222-223
   PRIV_PROC_LOCK_MEMORY, 167-168
   processes with assigned privileges, 169
   programs aware of privileges, 169
   protecting kernel processes, 164
   privileges (Continued)
      removing from a user, 171
      removing from basic set, 228
      removing from limit set, 228
      task map, 221
      troubleshooting requirements for, 224-225
      using in shell script, 229
   privileges file, description, 166
   PROC privileges, 166
   process audit characteristics
      audit ID, 641
      audit session ID, 641
      process preselection mask, 641
      terminal ID, 641
   process audit class, 638
   process audit token, format, 651
   process modify audit class, 638
   process preselection mask, description, 641
   process privileges, 166
   process rights management, See privileges
   process start audit class, 638
   processing time costs, of audit service, 572
   prof_attr database
      description, 215-216
      summary, 211
   profile shells, description, 163
   profiles, See rights profiles
   profiles command, description, 218
   PROFS_GRANTED keyword, policy.conf file, 217
   programs
      checking for RBAC authorizations, 202
      privilege-aware, 168, 169
      project.max-locked-memory resource
      control, 167-168
   PROM security mode, 74-76
   propagation
      KDC database, 382
      Kerberos database, 443-444
   protecting
      BIOS, pointer to, 74-75
      by using passwords with cryptographic
      framework, 281-282
      contents of keystore, 285
      files with cryptographic framework, 250
protecting (Continued)
  PROM, 74-75
  system from risky programs, 149-151
protection level
  clear, 528
  private, 528
  safe, 528
  setting in ftp, 528
Protocol keyword, ssh_config file, 349
providers
  adding library, 268-269
  adding software provider, 267-269
  adding user-level software provider, 268-269
  connecting to cryptographic framework, 247
  definition as plugins, 244
  definition in cryptographic framework, 245
  disabling hardware mechanisms, 274-276
  installing, 248
  listing hardware providers, 274
  listing in cryptographic framework, 264-267
  preventing use of kernel software provider, 271-273
  registering, 248
  restoring use of kernel software provider, 271
  signing, 248
proxiable ticket, definition, 539
proxy ticket, definition, 539
ProxyCommand keyword, ssh_config file, 349
pseudo-tty, use in Solaris Secure Shell, 345
PubkeyAuthentication keyword, Solaris Secure Shell, 349
public audit policy
  description, 570
  read-only events, 570
public directories
  auditing, 555
  sticky bit and, 131
public key authentication, Solaris Secure Shell, 322
public key cryptography
  AUTH_DH client-server session, 293-296
  changing NFS public keys and secret keys, 293
  common keys
    calculation, 295
  database of public keys for Secure RPC, 293
  generating keys
    conversation keys for Secure NFS, 294
    using Diffie-Hellman, 293
  NFS secret keys, 293
public key technologies, See PKI
public keys
  changing passphrase, 334-335
  DH authentication and, 292-296
  generating public-private key pair, 332-334
  Solaris Secure Shell identity files, 352
public objects, auditing, 555
publickey map, DH authentication, 292-296
Proccheck_method option, SASL and, 319
Q
qsize attribute, plugin entry, 579
quoting syntax in BART, 125
R
-R option
  bart create, 110, 115
  ssh command, 337-339
-r option
  bart create, 115
  passwd command, 41
  praudit command, 630
random numbers
  dd command, 250-252
  pktool command, 252-257
raw praudit output format, 630
RBAC
  adding custom roles, 182
  adding new rights profile, 198
  adding roles, 177-180
  adding roles from command line, 180-182
RBAC (Continued)
administration commands, 218-219
audit profiles, 637
auditing roles, 184
authorization database, 214-215
authorizations, 160
basic concepts, 157-160
changing role passwords, 193-194
changing user properties
from command line, 201
checking scripts or programs for
authorizations, 202
commands for managing, 218-219
compared to superuser model, 155-157
configuring, 175-188
database relationships, 211-212
databases, 211-218
editing rights profiles, 196-199
elements, 157-160
modifying roles, 195-196
modifying users, 200-201
name services and, 212
planning, 175-177
profile shells, 163
rights profile database, 215-216
rights profiles, 162
securing scripts, 202
using privileged applications, 190-192

RC4, See ARCFOUR kernel provider
rcp command
Kerberos and, 526-529, 535
rdist command, Kerberos and, 535
read_kt command, 513, 514
read permissions, symbolic mode, 132
readable audit record format
converting audit records to, 609, 629
realms (Kerberos)
configuration decisions, 378-379
configuring cross-realm authentication, 409-411
contents of, 366
direct, 410-411
hierarchical, 409-410
hierarchical or nonhierarchical, 365
hierarchy, 379
realms (Kerberos) (Continued)
in principal names, 364-365
mapping host names onto, 379
names, 378
number of, 378
requesting tickets for specific, 528
servers and, 366
reauth_timeout option, SASL and, 319
redirecting arrow (>), preventing redirection, 49
reducing
audit files, 604-606, 628
storage-space requirements for audit files, 573
refreshing, cryptographic services, 276-277
registering providers, cryptographic framework, 248
rem_drv command, description, 98
remote logins
authentication, 54-56
authorization, 54-56
preventing superuser from, 73-74
security and, 295
RemoteForward keyword, ssh_config file, 349
removing
ACL entries, 147-148
audit events from audit_event file, 621
cryptographic providers, 270, 271
device policy, 86
plugins from KMF, 287-288
policy from device, 86
principals with kt remove command, 512
privileges from basic set, 228
privileges from limit set, 228
service principal from keytab file, 512
software providers
permanently, 273
temporarily, 272
renewable ticket, definition, 539
replacing, superuser with roles, 175-177
replayed transactions, 295
reporting tool, See bart compare
reports, BART, 105
required control flag, PAM, 312
requisite control flag, PAM, 312
resource controls
privileges, and, 167-168
resource controls (Continued)
project.max-locked-memory, 167-168
zone.max-locked-memory, 167-168
restarting
audit daemon, 596
cryptographic services, 276-277
ssh service, 330
sshd daemon, 330
restoring, cryptographic providers, 271
restricted shell (rsh), 49
restricting
remote superuser access, 73-74
superuser task map, 71-72
user privileges, 228
restricting access for KDC servers, 460-461
RETRIES in Solaris Secure Shell, 351
return audit token, format, 652
rewolfl option
mt command
tape device cleanup and, 103
.rhosts file, description, 353
RhostsAuthentication keyword, Solaris Secure Shell, 349
RhostsRSAAuthentication keyword, Solaris Secure Shell, 349
right, See rights profiles
rights profiles
for audit service, 636-637
changing contents of, 196-199
changing from command line, 197
contents of typical, 205
creating
in Solaris Management Console, 198
on command line, 197
creating roles for, 177-180
databases
See prof_attr database and exec_attr database
description, 158, 162
major rights profiles descriptions, 205
methods of creating, 196-199
modifying, 196-199
ordering, 209
troubleshooting, 199
using the System Administrator profile, 74
rights profiles (Continued)
viewing contents, 209-210
Rights tool, description, 196-199
rlogin command
Kerberos and, 526-529, 535
rlogind daemon, Kerberos and, 536
role-based access control, See RBAC
roleadd command
description, 219
using, 180
roledel command, description, 219
rolemod command
creating properties of role, 195
description, 219
roles
adding custom roles, 182
adding for particular profiles, 177-180
adding from command line, 180-182
assigning privileges to, 226-227
assigning with usermod command, 183-184
assuming, 188-190, 190-192
assuming after login, 162
assuming in a terminal window, 163, 188-190
assuming in Solaris Management Console, 190-192
assuming Primary Administrator role, 189
assuming root role, 189-190
assuming System Administrator role, 190
auditing, 184
changing password of, 193-194
changing properties of, 195-196
creating
Crypto Management role, 183-184
Custom Operator role, 182
Device Security role, 179
DHCP Management role, 179
for particular profiles, 177-180
Network Security role, 179
on command line, 180-182
Operator role, 178-179
role with limited scope, 179
root role, 184-188
security-related roles, 179
System Administrator role, 178
description, 162-163
roles (Continued)
determining directly assigned privileges, 231
determining role's privileged commands, 233-235
listing local roles, 188, 218
making root user into role, 184-188
modifying, 195-196
modifying assignment to a user, 179
recommended roles, 156
summary, 158
troubleshooting, 179
use in RBAC, 156
using an assigned role, 188-190, 190-192
using to access the hardware, 74-75
roles command
description, 218
using, 188
root principal, adding to host's keytab, 509
root role (RBAC)
assuming role, 189-190
changing back into root user, 187
troubleshooting, 188
root user
changing from root role, 187
changing to root role, 184-188
displaying access attempts on console, 73-74
login account
description, 44
monitoring su command attempts, 47-48, 72-73
replacing in RBAC, 162
restricting access, 53
restricting remote access, 73-74
tracking logins, 47-48
RPCSEC_GSS API, Kerberos and, 374
RSA kernel provider, 264
RSAAuthentication keyword, Solaris Secure Shell, 349
rsh command
Kerberos and, 526-529, 535
rsh command (restricted shell), 49
rshd daemon, Kerberos and, 536
rshchown system variable, 139
rules file (BART), 107-108
rules file attributes, See keywords
rules file format (BART), 124-125
rules file specification language, See quoting syntax
S
-S option, st_clean script, 104
-s option
audit command, 627
praudit command, 630
safe protection level, 528
SASL
environment variable, 318
options, 319
overview, 317
plug-ins, 318
saslauthd_path option, SASL and, 319
saving, failed login attempts, 63
scope (RBAC), description, 163
scp command
copying files with, 339-340
description, 355
scripts
audit_warn script, 635
bsmconv effect, 632
bsmconv for device allocation, 88
bsmconv script, 636
bsmconv to enable auditing, 594-595
checking for RBAC authorizations, 202
device-clean scripts
See also device-clean scripts
for cleaning devices, 102-104
monitoring audit files example, 573
processing praudit output, 630-631
running with privileges, 171
securing, 202
use of privileges in, 229
SCSI devices, st_clean script, 102
SEAM Administration Tool
and limited administration privileges, 508-509
and list privileges, 508
and X Window system, 480-481
command-line equivalents, 480-481
context-sensitive help, 481
creating a new policy, 489, 501-502
creating a new principal, 489-492
default values, 483
deleting a principal, 494
deleting policies, 504-505
SEAM Administration Tool (Continued)
displaying sublist of principals, 486
duplicating a principal, 492
files modified by, 481
Filter Pattern field, 486
gkadmin command, 479
.gkadmin file, 481
help, 481-482
Help Contents, 482
how affected by privileges, 508
kadmin command, 479
login window, 482
modifying a policy, 503-504
modifying a principal, 493
online help, 481-482
or kadmin command, 480
overview, 480-483
panel descriptions, 505-508
privileges, 508
setting up principal defaults, 494-495
starting, 482-483
table of panels, 505-508
viewing a principal's attributes, 487-489
viewing list of policies, 497-499
viewing list of principals, 485-487
viewing policy attributes, 499-501
secondary audit directory, 633
secret keys
creating, 250-252, 252-257
generating
using the dd command, 250-252
using the pktool command, 252-257
generating for Secure RPC, 293
Secure by Default installation option, 50
secure connection
across a firewall, 340
logging in, 335-336
Secure NFS, 292
Secure RPC
alternative, 55
and Kerberos, 292
description, 291
implementation of, 293-296
keyserver, 293
Secure RPC (Continued)
overview, 54-56
securing
logins task map, 60
network at installation, 50
passwords task map, 60
scripts, 202
security
across insecure network, 340
auditing and, 554
BART, 109
computing digest of files, 257-258
computing MAC of files, 258-260
devices, 45-47
DH authentication, 293-296
genrerating files, 260-263
installation options, 50
netservices limited installation option, 50
NFS client-server, 293-296
password encryption, 42
pointer to JASS toolkit, 50
policy overview, 35-36
preventing remote login, 73-74
protecting against denial of service, 50
protecting against Trojan horse, 48
protecting devices, 102-104
protecting hardware, 74-76
protecting PROM, 74-76
Secure by Default, 50
system hardware, 74-76
security attributes
checking for, 160
considerations when directly assigning, 163-164
description, 158
Printer management rights profile, 159
privileges on commands, 161
special ID on commands, 161
using to mount allocated device, 89
security mechanism, specifying with -m option, 528
security modes, setting up environment with multiple, 418-419
security policy, default (RBAC), 211
security service, Kerberos and, 367
selecting
    audit classes, 577-579
    audit records, 606-608
    events from audit trail, 606-608
semicolon (;)
    device_allocate file, 101
    separator of security attributes, 217
sendmail command, authorizations required, 220
seq audit policy
    and sequence token, 570, 652
    description, 570
sequence audit token
    and seq audit policy, 652
    format, 652
ServerKeyBits keyword, sshd_config file, 349
servers
    AUTH_DH client-server session, 293-296
    configuring for Solaris Secure Shell, 346
    definition in Kerberos, 537
    gaining access with Kerberos, 542-545
    obtaining credential for, 543-544
    realms and, 366
service
    definition in Kerberos, 537
    disabling on a host, 513-515
    obtaining access for specific service, 544-545
    service keys
        definition in Kerberos, 537
        keytab files and, 509-515
    service management facility
        enabling keyserver, 297
        refreshing cryptographic framework, 268
        restarting cryptographic framework, 276-277
        restarting Solaris Secure Shell, 330
    Service Management Facility (SMF), See SMF
service principal
    adding to keytab file, 509, 510-511
    description, 365
    planning for names, 379-380
    removing from keytab file, 512
session ID, audit, 641
session keys
    definition in Kerberos, 537
    Kerberos authentication and, 542
setfacl command
    -d option, 148
    description, 135-136
    examples, 147
    -f option, 146
    syntax, 144-146
setgid permissions
    absolute mode, 133, 143
    description, 130
    security risks, 130
    symbolic mode, 132
setpin subcommand, pktool command, 286-287
    -setqctrl option, auditconfig command, 593-594
setting
    arg policy, 618
    argv policy, 618
    audit policy, 590-592
    audit queue parameters, 592-594
    principal defaults (Kerberos), 494-495
setuid permissions
    absolute mode, 133, 143
    description, 130
    finding files with permissions set, 149
    security risks, 49, 130
    symbolic mode, 132
sftp command
    auditing file transfers, 622-624
    copying files with, 340
    description, 355
sh command, privileged version, 163
SHA1 kernel provider, 264
sharing files
    and network security, 53
    with DH authentication, 301
shell, privileged versions, 163
shell commands
    /etc/d_passwd file entries, 45
    passing parent shell process number, 222
    shell process, listing its privileges, 222-223
    shell scripts, writing privileged, 229
    short praudit output format, 630
    shosts.equiv file, description, 353
    .shosts file, description, 353
signal received during auditing shutdown, 636
signing providers, cryptographic framework, 248
single-sign-on system, 526-532
Kerberos and, 359
size of audit files
  reducing, 604-606, 628
  reducing storage-space requirements, 573
slave datatrans file
description, 534
KDC propagation and, 443-444
slave datatrans_slave file, description, 534
slave KDCs
  automatically configuring, 402-403
  configuring, 404-408
definition, 536
  interactively configuring, 403-404
master KDC and, 366
  or master, 388
planning for, 380-381
swapping with master KDC, 438-442
slot, definition in cryptographic framework, 245
smattrpop command, description, 219
smexec command, description, 219
SMF
  auditd service, 625-626
  managing auditd service, 568
  managing Secure by Default configuration, 50
smmultiuser command, description, 219
smprofile command
  changing rights profile, 197
description, 219
smrole command
  changing properties of role, 193, 195
description, 219
using, 182
smuser command
  changing user's RBAC properties, 200
description, 219
socket audit token, 652-653
soft limit
  audit_warn condition, 635
  minfree line description, 633
soft string, audit_warn script, 635
Solaris auditing task map, 575
Solaris Cryptographic Framework, See cryptographic framework
solaris.device.revoke authorization, 99
Solaris Secure Shell
  adding to system, 352
  administering, 343-345
  administrator task map, 325
  authentication
    requirements for, 322-324
    authentication methods, 322-324
    authentication steps, 344-345
    basis from OpenSSH, 324-325
    changes in current release, 324-325
    changing passphrase, 334-335
    command execution, 345
    configuring clients, 346
    configuring port forwarding, 329-330
    configuring server, 346
    connecting across a firewall, 340
    connecting outside firewall
      from command line, 341
      from configuration file, 340-341
copying files, 339-340
  creating keys, 332-334
data forwarding, 345
description, 321
files, 352
forwarding mail, 338
generating keys, 332-334
keywords, 346-351
local port forwarding, 338, 339
logging in fewer prompts, 336-337
logging in to remote host, 335-336
login environment variables and, 350-351
naming identity files, 352
packages, 352
protocol versions, 321
public key authentication, 322
remote port forwarding, 339
scp command, 339-340
specifying exceptions to system defaults, 330-331
TCP and, 329
typical session, 343-345
user procedures, 331-332
Solaris Secure Shell (Continued)
  using port forwarding, 337-339
  using without password, 336-337
solaris security policy, 216
special permissions
  setgid permissions, 130
  setuid permissions, 130
  sticky bit, 130-131
square brackets ([ ]), audit record output, 644
sr_clean script, description, 103
ssh-add command
  description, 355
  example, 336-337, 337
  storing private keys, 336-337
ssh-agent command
  description, 355
  from command line, 336-337
ssh command
  description, 354
  overriding keyword settings, 355
  port forwarding options, 337-339
  using, 335-336
  using a proxy command, 341
.ssh/config file
  description, 354
  override, 354
.ssh_config file
  configuring Solaris Secure Shell, 346
  host-specific parameters, 350
  keywords, 346-351
  See specific keyword
  override, 354
.ssh/environment file, description, 353
.ssh_host_dsa_key file, description, 352
.ssh_host_dsa_key.pub file, description, 352
.ssh_host_key file
  description, 352
  override, 354
.ssh_host_key.pub file, description, 352
.ssh_host_rsa_key file, description, 352
.ssh_host_rsa_key.pub file, description, 352
.ssh/id_dsa file, 354
.ssh/id_rsa file, 354
.ssh/identity file, 354
ssh-keygen command
  description, 355
  using, 332-334
ssh-keyscan command, description, 355
ssh-keysign command, description, 355
.ssh/known_hosts file
  description, 353
  override, 354
.ssh_known_hosts file, 353
.ssh/rc file, description, 353
sshd command, description, 355
sshd_config file
  description, 352
  keywords, 346-351
  See specific keyword
  overrides of /etc/default/login entries, 350-351
sshd.pid file, description, 353
.ssh/rc file, description, 354
st_clean script
  description, 102-103
  for tape drives, 102
standard cleanup, st_clean script, 104
starting
  audit daemon, 597
  auditing, 594-595, 625-626
  device allocation, 88-89
  KDC daemon, 408, 453
  Secure RPC key server, 297
stash file
  creating, 408, 453
  definition, 536
sticky bit permissions
  absolute mode, 133, 143
  description, 130-131
  symbolic mode, 132
stopping, dial-up logins temporarily, 67
storage costs, and auditing, 572-573
storage overflow prevention, audit trail, 611
storing
  audit files, 565-566, 587-589
  passphrase, 262
StrictHostKeyChecking keyword, ssh_config file, 349
StrictModes keyword, sshd_config file, 349
su command
  displaying access attempts on console, 73-74
  in role assumption, 188-190, 190-192
  monitoring use, 72-73
su file, monitoring su command, 72-73
subject audit token, format, 653
Subsystem keyword, sshd_config file, 350
success
  audit class prefix, 639
  turning off audit classes for, 640
sufficient control flag, PAM, 312
sulog file, 72-73
  monitoring contents of, 72
Sun Crypto Accelerator 1000 board, listing mechanisms, 274-276
Sun Crypto Accelerator 6000 board
  hardware plugin to cryptographic framework, 244
  listing mechanisms, 274
SUPATH in Solaris Secure Shell, 351
superuser
  compared to privilege model, 164-172
  compared to RBAC model, 155-157
  differences from privilege model, 166
  eliminating in RBAC, 162
  monitoring access attempts, 73-74
  troubleshooting becoming root as a role, 188
  troubleshooting remote access, 74
suser security policy, 216
svcadm command
  administering cryptographic framework, 246
  enabling cryptographic framework, 276-277
  enabling keyservers daemon, 297
  refreshing cryptographic framework, 267-269
  restarting name service, 178
  restarting NFS server, 589
  restarting Solaris Secure Shell, 330
  restarting syslog daemon, 64, 580
svcs command
  listing cryptographic services, 276-277
  listing keyservers service, 297
  swapping master and slave KDCs, 438-442
symbolic links, file permissions, 129
symbolic mode
  changing file permissions, 132, 140-141, 141
symbolic mode (Continued)
  description, 132
synchronizing clocks
  master KDC, 396, 402
  overview, 437-438
  slave KDC, 408, 453
SYS privileges, 166
syslog.conf file
  and auditing, 632
  audit notice level, 580
  audit records, 553
  executable stack messages, 136
  kern notice level, 136
  priv debug entry, 239
  saving failed login attempts, 64-65
SYSLOG_FAILED_LOGINS
  in Solaris Secure Shell, 351
system variable, 64
syslog format, audit records, 632
SyslogFacility keyword, sshd_config file, 350
System Administrator (RBAC)
  assuming role, 190
  creating role, 178
  protecting hardware, 74
  recommended role, 156
  rights profile, 206
system calls
  arg audit token, 646
  close, 638
  exec_arg audit token, 647
  exec_env audit token, 647-648
  ioctl(), 638
  ioctl to clean audio device, 103
  return audit token, 652
system file, bsmconv effect on, 632
system hardware, controlling access to, 74-76
system properties, privileges relating to, 166
system security
  ACL, 133-136
  dial-up logins and passwords, 45
  dial-up passwords
    disabling temporarily, 67
    displaying
      user's login status, 60-61, 61
system security, displaying *(Continued)*

users with no passwords, 62
firewall systems, 56-57
hardware protection, 40, 74-76
login access restrictions, 40
machine access, 40
overview, 39
password encryption, 42
passwords, 41
privileges, 164-172
protecting from risky programs, 149-151
restricted shell, 49
restricting remote root access, 73-74
role-based access control (RBAC), 48, 155-157
root access restrictions, 53, 73-74
saving failed login attempts, 63
special logins, 44
su command monitoring, 47-48, 72-73
task map, 149
system state audit class, 638
System V IPC
   ipc audit class, 638
   ipc audit token, 649-650
   ipc_perm audit token, 650
   privileges, 166
system variables
   See also variables
   CRYPT_DEFAULT, 68
   KEYBOARD_ABORT, 75-76
   noexec_user_stack, 151
   noexec_user_stack_log, 151
   rstchown, 139
   SYSLOG_FAILED_LOGINS, 64
system-wide administration audit class, 638
systems, protecting from risky programs, 149-151
tape drives
   allocating, 94
cleaning of data, 102-103
device-clean scripts, 102
task maps
   administering cryptographic framework, 263-264
   administering policies (Kerberos), 497
   administering principals (Kerberos), 483-484
   administering Secure RPC, 296
   allocating devices, 93
   auditing, 575
   changing default algorithm for password encryption, 67
   configuring audit files, 576
   configuring audit service, 586-587
   configuring device policy, 84
   configuring devices, 83
   configuring Kerberos NFS servers, 413
   configuring RBAC, 174-175
   configuring Solaris Secure Shell, 325
   controlling access to system hardware, 74
cryptographic framework, 249
device allocation, 87-88
device policy, 84
devices, 83
enabling audit service, 586-587
Kerberos configuration, 387-388
Kerberos maintenance, 388
managing and using privileges, 221
managing audit records, 601-602
managing device allocation, 87-88
managing device policy, 84
managing RBAC, 192
monitoring and restricting superuser, 71-72
PAM, 306
planning auditing, 563
protecting against programs with security risk, 149
protecting files, 137
protecting files with ACLs, 143
protecting files with cryptographic mechanisms, 250
protecting files with UNIX permissions, 137
protecting system hardware, 74
securing logins and passwords, 60
task maps (Continued)
securing systems, 59-60
Solaris Secure Shell, 325
system access, 59-60
troubleshooting Solaris auditing, 612-624
Using BART task map, 108-109
using device allocation, 93
using RBAC, 173
using roles, 188
using Solaris Secure Shell, 331-332
using the cryptographic framework, 249
Using the Key Management Framework (Task Map), 281-282
TCP
addresses, 650
Solaris Secure Shell and, 329, 345
telnet command
Kerberos and, 526-529, 535
telnetd daemon, Kerberos and, 536
terminal ID, audit, 641
terminating, signal received during auditing shutdown, 636
terminology
authentication-specific, 537-538
Kerberos, 536-541
Kerberos-specific, 536
test manifests, 107
text audit token, format, 653
TGS, getting credential for, 542-543
TGT, in Kerberos, 361-362
third-party password algorithms, adding, 70-71
ticket file, See credential cache
ticket-granting service, See TGS
ticket-granting ticket, See TGT
tickets
creating, 517-518
creating with kinit, 518
definition, 360
definition in Kerberos, 537
destroying, 520
-F option or -f option, 527
file
See credential cache
forwardable, 360, 518, 529, 538
tickets (Continued)
initial, 538
invalid, 538
-k option, 528
klist command, 519-520
lifetime, 539-540
maximum renewable lifetime, 540
obtaining, 517-518
or credentials, 361
postdatable, 538
postdated, 360
proxiable, 539
proxy, 539
renewable, 539
requesting for specific realm, 528
types of, 538-541
viewing, 519-520
warning about expiration, 431
TIMEOUT in Solaris Secure Shell, 351
timestamps, audit files, 643
/tmp/krb5cc_uid file, description, 534
/tmp/ovsec_adm.xxxxx file, description, 534
tmpfile string, audit_warn script, 636
TMPFS file system, security, 131
token, definition in cryptographic framework, 245
trail audit policy
and trailer token, 570
description, 570
trailer audit token
format, 654
order in audit record, 654
praudit display, 654
transparency, definition in Kerberos, 360
Trojan horse, 48
troubleshooting
allocating a device, 94
audit classes
customized, 584, 614
auditing, 612-624
becoming superuser, 188
computer break-in attempts, 63
encrypt command, 263
finding files with setuid permissions, 149
Kerberos, 475

Index
troubleshooting (Continued)
lack of privilege, 224-225
list_devices command, 90
mounting a device, 96
praudit command, 610
preventing programs from using executable stacks, 150-151
privilege requirements, 224-225
remote superuser access, 74
rights profiles, 199
role capabilities, 179
root as a role, 188
terminal where su command originated, 72
user running privileged commands, 231-233
truss command, for privilege debugging, 224
trusted hosts, 56
types of tickets, 538-541
TZ in Solaris Secure Shell, 351

U
-U option
allocate command, 99
list_devices command, 99
uauth audit token, 564
UDP
addresses, 650
port forwarding and, 329
Solaris Secure Shell and, 329
using for remote audit logs, 558
umask value
and file creation, 131
typical settings, 131
unmount command, with security attributes, 89
uninstalling, cryptographic providers, 270
UNIX file permissions, See files, permissions
unmounting, allocated devices, 97
update_drv command
description, 98
using, 85-86
updating, audit service, 596-597
upriv audit token, 654
URL for online help, Graphical Kerberos Tool, 385
use_authid option, SASL and, 319
UseLogin keyword, sshd_config file, 350
UseOpenSSLEngine keyword
ssh_config file, 350
sshd_config file, 350
user accounts
See also users
displaying login status, 60-61, 61
User Accounts tool, description, 200-201
user ACL entries
default entries for directories, 135
description, 134-135
setting, 144-146
user administration audit class, 638
user_attr database
defaultpriv keyword, 238
description, 211, 213
limitpriv keyword, 238
RBAC relationships, 211-212
user audit fields, audit_user database, 634-635
user classes of files, 128
user database (RBAC), See user_attr database
user ID
audit ID and, 551-553, 641
in NFS services, 415-416
User keyword, ssh_config file, 350
user principal, description, 365
user procedures
adding plugins to KMF, 287-288
allocating devices, 93
assuming a role, 188
chkey command, 300
computing digest of a file, 257-258
computing MAC of a file, 258-260
creating self-signed certificate, 282-283
decrypting files, 260-263
encrypting files, 250
encrypting NIS user's private key, 300
exporting certificates, 285-286
generating a symmetric key
using the dd command, 250-252
using the pktool command, 252-257
generating passphrase for keystore, 286-287
importing certificates, 283-285
protecting files, 137-143
user procedures (Continued)
  using ACLs, 143-149
  using an assigned role, 188
  using pktool command, 281-282
  using Solaris Secure Shell, 331-332
useradd command
  adding local user, 185
  description, 219
userdel command, description, 219
UserKnownHostsFile keyword, ssh_config file, 350
UserKnownHostsFile2 keyword, See UserKnownHostsFile keyword
usermod command
  changing user's RBAC properties, 200
  description, 219
  using to assign role, 183-184
users
  adding local user, 185
  allocating devices, 93-94
  assigning allocate authorization to, 89
  assigning privileges to, 226-227
  assigning RBAC defaults, 217-218
  auditing all of their commands, 617-618
  basic privilege set, 169
  changing properties from command line, 201
  computing digest of files, 257-258
  computing MAC of files, 258-260
  creating local user, 185
  deallocating devices, 96-97
  determining directly assigned privileges, 230-231
  determining own privileged commands, 231-233
  disabling login, 62-63
  displaying login status, 60-61
  encrypting files, 260-263
  exceptions to Solaris Secure Shell defaults, 330-331
  generating a symmetric key, 252-257
  having no passwords, 61-62
  initial inheritable privileges, 168
  modifying audit preselection mask of, 581-583
  modifying properties (RBAC), 200-201
  mounting allocated devices, 94-96
  restricting basic privileges, 228
  troubleshooting running privileged commands, 231-233
users (Continued)
  unmounatng allocated devices, 97
using
  ACLs, 144-146
  allocate command, 93-94
  BART, 109
  cryptoadm command, 264
  cryptographic framework task map, 249
  dd command, 250-252
  deallocate command, 96-97
  device allocation, 93
  digest command, 257-258
  encrypt command, 260-263
  file permissions, 137
  mac command, 258-260
  mount command, 95
  new password algorithm, 68
  pktool command, 252-257
  ppriv command, 222
  privileges, 230
  privileges task map, 229
  RBAC task map, 173
  roles, 188
  roles task map, 188
  smrole command, 227
  Solaris Secure Shell task map, 331-332
  ssh-add command, 336-337
  ssh-agent daemon, 336-337
  truss command, 224
  umount command, 97
  usermod command, 226
Using the Key Management Framework (Task Map), 281-282
/usr/bin/ftp command, Kerberos and, 535
/usr/bin/kdestroy command, Kerberos and, 535
/usr/bin/kinit command, Kerberos and, 535
/usr/bin/klist command, Kerberos and, 535
/usr/bin/kpasswd command, Kerberos and, 535
/usr/bin/ktutil command, Kerberos and, 535
/usr/bin/rcp command, Kerberos and, 535
/usr/bin/rdist command, Kerberos and, 535
/usr/bin/rlogin command, Kerberos and, 535
/usr/bin/rsh command, Kerberos and, 535
/usr/bin/telnet command, Kerberos and, 535
V

v1 protocol, Solaris Secure Shell, 321

-v option
  audit command, 577
  digest command, 257
  mac command, 259
  ppri command, 222

v2 protocol, Solaris Secure Shell, 321

/var/adm/auditlog file, text audit records, 580
/var/adm/loginlog file, saving failed login attempts, 63
/var/adm/messages file
  executable stack messages, 136
  troubleshooting auditing, 615
/var/adm/sulog file, monitoring contents of, 72
/var/krb5-/k5.REALM file, description, 534
/var/krb5/kadmin.log file, description, 534
/var/krb5/kdcd.log file, description, 534
/var/krb5/principal file, description, 534
/var/krb5/principal.kadm5 file, description, 534
/var/krb5/principal.kadm5.lock file, description, 534
/var/krb5/principal.ok file, description, 534
/var/krb5/principal.ulo file, description, 534
/var/krb5/slave_datatrans file, description, 534
/var/krb5/slave_datatrans_slave file, description, 534
/var/log/authlog file, failed logins, 64-65
/var/log/sulog file, troubleshooting auditing, 615
/var/run/sshd.pid file, description, 353

variables
  adding to audit record, 569, 647-648
  auditing those associated with a command, 647
  for proxy servers and ports, 340
  KEYBOARD_ABORT, 75-76
  login and Solaris Secure Shell, 350-351
  noexec_user_stack, 136
  noexec_user_stack_log, 136
  rstchown, 139
  setting in Solaris Secure Shell, 351

verifiers
description, 294
  returned to NFS client, 295
  window, 294

VerifyReverseMapping keyword, ssh_config file, 350

viewing
  ACL entries, 148-149
  audit record formats, 602-604
  available cryptographic mechanisms, 266, 271
  binary audit files, 608-610
  contents of rights profiles, 209-210
  cryptographic mechanisms
    available, 266, 271
    existing, 265, 266, 271
  device allocation information, 90
  device policy, 84-85
  digest of a file, 257
  directly assigned privileges, 230-231
  existing cryptographic mechanisms, 266, 271
  file permissions, 137-138
  keylist buffer with list command, 513, 514
  list of policies, 497-499
  list of principals, 485-487
viewing (Continued)
- MAC of a file, 259
- policy attributes, 499-501
- principal's attributes, 487-489
- privileges in a shell, 223, 231
- privileges on a process, 222
- tickets, 519-520
- user's login status, 60-61
- users with no passwords, 61-62
- XML audit records, 609, 630

virus scanning
- configuring, 78-82
- described, 78
- engines, 77-78
- files, 77-78

viruses
- denial of service attack, 50
- Trojan horse, 48

vnode audit token, format, 646

vold daemon, turned off by device allocation, 89

W
- warn.conf file, description, 534
- warning about ticket expiration, 431
- wildcard characters
  - for hosts in Solaris Secure Shell, 340
  - in RBAC authorizations, 210
- window verifier, 294
- write permissions, symbolic mode, 132

X
- -x option
  - Kerberized commands, 528
  - praudit command, 630
- Xauth command, X11 forwarding, 350
- XAuthLocation keyword, Solaris Secure Shell port forwarding, 350
- XML format, audit records, 610
- XML option, praudit command, 630
- Xylogics tape drive device-clean script, 102

Z
- zone.max-locked-memory resource control, 167-168
- zonename audit policy
  - description, 570
  - using, 565, 637
- zonename audit token, 654
- zones
  - auditing and, 560-561, 637
  - configuring auditing in global zone, 591
  - cryptographic framework and, 248
  - cryptographic services and, 276-277
  - devices and, 46
  - perzone audit policy, 560-561, 565, 637
  - planning auditing in, 564-565
  - zonename audit policy, 565, 637