Oracle® Solaris Administration: Security Services

Part No: 821–1456–10
March 2012

120305@25097

Contents

Preface 23

Part I **Security Overview** 27

1 **Security Services (Overview)** 29

System Security 29

Cryptographic Services 30

Authentication Services 31

Authentication With Encryption 32

Auditing 32

Security Policy 32

Part II **System, File, and Device Security** 35

2 **Managing Machine Security (Overview)** 37

Controlling Access to a Computer System 37

- Maintaining Physical Security 38
- Maintaining Login Control 38

Controlling Access to Devices 42

- Device Policy (Overview) 43
- Device Allocation (Overview) 44

Controlling Access to Machine Resources 45

- Limiting and Monitoring Superuser 45
- Configuring Role-Based Access Control to Replace Superuser 45
- Preventing Unintentional Misuse of System Resources 45
- Restricting `setuid` Executable Files 47
- Using the Secure by Default Configuration 47

Using Resource Management Features 47
Using Oracle Solaris Zones 47
Monitoring Use of Machine Resources 48
Monitoring File Integrity 48
Controlling Access to Files 48
Protecting Files With Encryption 49
Using Access Control Lists 49
Sharing Files Across Machines 49
Restricting `root` Access to Shared Files 50
Controlling Network Access 50
Network Security Mechanisms 50
Authentication and Authorization for Remote Access 51
Firewall Systems 52
Encryption and Firewall Systems 53
Reporting Security Problems 54

3 Controlling Access to Systems (Tasks) 55
Controlling System Access (Task Map) 55
Securing Logins and Passwords (Tasks) 56
Securing Logins and Passwords (Task Map) 56
▼ How to Change the `root` Password 56
▼ How to Display a User's Login Status 57
▼ How to Display Users Without Passwords 58
▼ How to Temporarily Disable User Logins 58
▼ How to Monitor Failed Login Attempts 59
▼ How to Monitor All Failed Login Attempts 60
Changing the Default Algorithm for Password Encryption (Tasks) 61
▼ How to Specify an Algorithm for Password Encryption 61
▼ How to Specify a New Password Algorithm for an NIS Domain 62
▼ How to Specify a New Password Algorithm for an LDAP Domain 63
Monitoring and Restricting Superuser (Tasks) 64
▼ How to Monitor Who Is Using the `su` Command 64
▼ How to Restrict and Monitor Superuser Logins 65
Controlling Access to System Hardware (Tasks) 66
▼ How to Require a Password for Hardware Access 66

▼ How to Disable a System's Abort Sequence 67

4 Virus Scanning Service (Tasks) 69
About Virus Scanning 69
About the Vscan Service 70
Using the Vscan Service (Tasks) 70
▼ How to Enable Virus Scanning on a File System 71
▼ How to Enable the Vscan Service 72
▼ How to Add a Scan Engine 72
▼ How to View Vscan Properties 72
▼ How to Change Vscan Properties 73
▼ How to Exclude Files From Virus Scans 73

5 Controlling Access to Devices (Tasks) 75
Configuring Devices (Task Map) 75
Configuring Device Policy (Tasks) 76
Configuring Device Policy (Task Map) 76
▼ How to View Device Policy 76
▼ How to Change the Device Policy on an Existing Device 77
▼ How to Audit Changes in Device Policy 78
▼ How to Retrieve IP MIB-II Information From a `/dev/*` Device 78
Managing Device Allocation (Tasks) 79
Managing Device Allocation (Task Map) 79
▼ How to Enable Device Allocation 80
▼ How to Authorize Users to Allocate a Device 80
▼ How to View Allocation Information About a Device 81
▼ Forcibly Allocating a Device 82
▼ Forcibly Deallocating a Device 82
▼ How to Change Which Devices Can Be Allocated 82
▼ How to Audit Device Allocation 83
Allocating Devices (Tasks) 84
▼ How to Allocate a Device 84
▼ How to Mount an Allocated Device 85
▼ How to Deallocate a Device 87
Device Protection (Reference) 87

Device Policy Commands 88
Device Allocation 88

6 Using the Basic Audit Reporting Tool (Tasks) 95
Basic Audit Reporting Tool (Overview) 95
BART Features 95
BART Components 96
Using BART (Tasks) 98
BART Security Considerations 98
Using BART (Task Map) 99
▼ How to Create a Manifest 99
▼ How to Customize a Manifest 101
▼ How to Compare Manifests for the Same System Over Time 102
▼ How to Compare Manifests From Different Systems 104
▼ How to Customize a BART Report by Specifying File Attributes 106
▼ How to Customize a BART Report by Using a Rules File 107
BART Manifests, Rules Files, and Reports (Reference) 108
BART Manifest File Format 108
BART Rules File Format 109
BART Reporting 111

7 Controlling Access to Files (Tasks) 113
Using UNIX Permissions to Protect Files 113
Commands for Viewing and Securing Files 113
File and Directory Ownership 114
UNIX File Permissions 114
Special File Permissions (`setuid`, `setgid` and Sticky Bit) 115
Default `umask` Value 117
File Permission Modes 117
Using Access Control Lists to Protect UFS Files 119
Protecting Executable Files From Compromising Security 121
Protecting Files (Tasks) 121
Protecting Files With UNIX Permissions (Task Map) 122
▼ How to Display File Information 122
▼ How to Change the Owner of a File 123

▼ How to Change Group Ownership of a File 124
▼ How to Change File Permissions in Symbolic Mode 124
▼ How to Change File Permissions in Absolute Mode 125
▼ How to Change Special File Permissions in Absolute Mode 126
Protecting Against Programs With Security Risk (Task Map) 127
▼ How to Find Files With Special File Permissions 128
▼ How to Disable Programs From Using Executable Stacks 129

Part III Roles, Rights Profiles, and Privileges 131

8 Using Roles and Privileges (Overview) 133
Role-Based Access Control (Overview) 133
RBAC: An Alternative to the Superuser Model 133
RBAC Elements and Basic Concepts 137
Privilege Escalation 139
RBAC Authorizations 140
Authorizations and Privileges 141
Privileged Applications and RBAC 141
RBAC Rights Profiles 142
RBAC Roles 143
Profile Shells and RBAC 144
Name Service Scope and RBAC 144
Security Considerations When Directly Assigning Security Attributes 144
Usability Considerations When Directly Assigning Security Attributes 145
Privileges (Overview) 145
Privileges Protect Kernel Processes 145
Privilege Descriptions 146
Administrative Differences on a System With Privileges 147
Privileges and System Resources 148
How Privileges Are Implemented 149
How Processes Get Privileges 150
Assigning Privileges 151
Privileges and Devices 152
Privileges and Debugging 153

9 Using Role-Based Access Control (Tasks) 155

Using RBAC (Tasks) 155

Viewing and Using RBAC Defaults (Tasks) 156

Viewing and Using RBAC Defaults (Task Map) 156

▼ How to View All Defined Security Attributes 156

▼ How to View Your Assigned Rights 157

▼ How to Assume a Role 159

▼ How to Obtain Administrative Rights 160

Customizing RBAC for Your Site (Tasks) 163

Initially Configuring RBAC (Task Map) 163

▼ How to Plan Your RBAC Implementation 163

▼ How to Create a Role 165

▼ How to Assign a Role 167

▼ How to Audit Roles 169

▼ How to Create or Change a Rights Profile 170

▼ How to Add RBAC Properties to Legacy Applications 171

▼ How to Troubleshoot RBAC and Privilege Assignment 173

Managing RBAC (Tasks) 176

Managing RBAC (Task Map) 176

▼ How to Change the Password of a Role 177

▼ How to Change the Security Attributes of a Role 178

▼ How to Change the RBAC Properties of a User 179

▼ How to Restrict a User to Desktop Applications 181

▼ How to Restrict an Administrator to Explicitly Assigned Rights 182

▼ How to Enable a User to Use Own Password to Assume a Role 183

▼ How to Change the `root` Role Into a User 184

Using Privileges (Tasks) 186

Determining Your Privileges (Task Map) 186

▼ How to List the Privileges on the System 187

▼ How to Determine the Privileges That You Have Been Directly Assigned 188

▼ How to Determine the Privileged Commands That You Can Run 189

Managing Privileges (Task Map) 191

▼ How to Determine the Privileges on a Process 191

▼ How to Determine Which Privileges a Program Requires 193

▼ How to Run a Shell Script With Privileged Commands 194

10 Security Attributes in Oracle Solaris (Reference) 197
Rights Profiles 197
Viewing the Contents of Rights Profiles 199
Order of Search for Assigned Security Attributes 199
Authorizations 200
Authorization Naming Conventions 200
Example of Authorization Granularity 200
Delegation Authority in Authorizations 201
RBAC Databases 201
RBAC Databases and the Naming Services 201
`user_attr` Database 202
`auth_attr` Database 202
`prof_attr` Database 203
`exec_attr` Database 203
`policy.conf` File 203
RBAC Commands 204
Commands That Manage RBAC 204
Selected Commands That Require Authorizations 205
Privileges 206
Administrative Commands for Handling Privileges 206
Files With Privilege Information 207
Privileges and Auditing 207
Prevention of Privilege Escalation 208
Legacy Applications and the Privilege Model 209

Part IV Cryptographic Services 211

11 Cryptographic Framework (Overview) 213
Introduction to the Cryptographic Framework 213
Terminology in the Cryptographic Framework 215
Scope of the Cryptographic Framework 217
Administrative Commands in the Cryptographic Framework 217
User-Level Commands in the Cryptographic Framework 217
Binary Signatures for Third-Party Software 218
Plugins to the Cryptographic Framework 218

Cryptographic Services and Zones 219

12 **Cryptographic Framework (Tasks)** 221

Using the Cryptographic Framework (Task Map) 221

Protecting Files With the Cryptographic Framework (Tasks) 221

Protecting Files With the Cryptographic Framework (Task Map) 222

▼ How to Generate a Symmetric Key by Using the `dd` Command 222

▼ How to Generate a Symmetric Key by Using the `pktool` Command 224

▼ How to Compute a Digest of a File 229

▼ How to Compute a MAC of a File 230

▼ How to Encrypt and Decrypt a File 232

Administering the Cryptographic Framework (Tasks) 235

Administering the Cryptographic Framework (Task Map) 235

▼ How to List Available Providers 236

▼ How to Add a Software Provider 240

▼ How to Prevent the Use of a User-Level Mechanism 241

▼ How to Prevent the Use of a Kernel Software Provider 243

▼ How to List Hardware Providers 245

▼ How to Disable Hardware Provider Mechanisms and Features 246

▼ How to Refresh or Restart All Cryptographic Services 248

13 **Key Management Framework** 249

Managing Public Key Technologies 249

Key Management Framework Utilities 250

KMF Policy Management 250

KMF Plugin Management 251

KMF Keystore Management 251

Using the Key Management Framework (Tasks) 252

Using the Key Management Framework (Task Map) 252

▼ How to Create a Certificate by Using the `pktool gencert` Command 253

▼ How to Import a Certificate Into Your Keystore 254

▼ How to Export a Certificate and Private Key in PKCS #12 Format 255

▼ How to Generate a Passphrase by Using the `pktool setpin` Command 256

▼ How to Generate a Key Pair by Using the `pktool genkeypair` Command 257

▼ How to Sign a Certificate Request by Using the `pktool signcsr` Command 261

▼ How to Manage Third-Party Plugins in KMF ... 262

Part V Authentication Services and Secure Communication ... 265

14 Network Services Authentication (Tasks) ... 267

Overview of Secure RPC ... 267

NFS Services and Secure RPC ... 267

DES Encryption With Secure NFS ... 268

Kerberos Authentication ... 268

Diffie-Hellman Authentication and Secure RPC ... 268

Administering Authentication With Secure RPC (Tasks) ... 272

Administering Secure RPC (Task Map) ... 272

▼ How to Restart the Secure RPC Keyserver ... 272

▼ How to Set Up a Diffie-Hellman Key for an NIS Host ... 272

▼ How to Set Up a Diffie-Hellman Key for an NIS User ... 273

▼ How to Share NFS Files With Diffie-Hellman Authentication ... 274

15 Using PAM ... 277

PAM (Overview) ... 277

Benefits of Using PAM ... 277

Introduction to the PAM Framework ... 278

Changes to PAM for This Release ... 279

PAM (Tasks) ... 279

PAM (Task Map) ... 280

Planning for Your PAM Implementation ... 280

▼ How to Add a PAM Module ... 281

▼ How to Prevent Rhost-Style Access From Remote Systems With PAM ... 282

▼ How to Log PAM Error Reports ... 282

PAM Configuration (Reference) ... 282

PAM Configuration File Syntax ... 283

How PAM Stacking Works ... 283

PAM Stacking Example ... 287

16 Using SASL 289
SASL (Overview) 289
SASL (Reference) 290
SASL Plug-ins 290
SASL Environment Variable 290
SASL Options 291

17 Using Secure Shell (Tasks) 293
Secure Shell (Overview) 293
Secure Shell Authentication 294
Secure Shell in the Enterprise 295
Secure Shell and the OpenSSH Project 295
Secure Shell and FIPS-140 Support 296
Secure Shell (Task Map) 297
Configuring Secure Shell (Tasks) 297
Configuring Secure Shell (Task Map) 297
▼ How to Set Up Host-Based Authentication for Secure Shell 297
▼ How to Configure Port Forwarding in Secure Shell 300
▼ How to Create User and Host Exceptions to SSH System Defaults 300
Using Secure Shell (Tasks) 301
Using Secure Shell (Task Map) 301
▼ How to Generate a Public/Private Key Pair for Use With Secure Shell 302
▼ How to Change the Passphrase for a Secure Shell Private Key 304
▼ How to Log In to a Remote Host With Secure Shell 304
▼ How to Reduce Password Prompts in Secure Shell 305
▼ How to Use Port Forwarding in Secure Shell 307
▼ How to Copy Files With Secure Shell 308
▼ How to Set Up Default Connections to Hosts Outside a Firewall 309

18 Secure Shell (Reference) 311
A Typical Secure Shell Session 311
Session Characteristics in Secure Shell 311
Authentication and Key Exchange in Secure Shell 312
Command Execution and Data Forwarding in Secure Shell 313
Client and Server Configuration in Secure Shell 313

Client Configuration in Secure Shell ... 313

Server Configuration in Secure Shell ... 314

Keywords in Secure Shell ... 314

Host-Specific Parameters in Secure Shell ... 318

Secure Shell and Login Environment Variables ... 318

Maintaining Known Hosts in Secure Shell ... 319

Secure Shell Files ... 320

Secure Shell Commands ... 322

Part VI Kerberos Service ... 325

19 Introduction to the Kerberos Service ... 327

What Is the Kerberos Service? ... 327

How the Kerberos Service Works ... 328

Initial Authentication: the Ticket-Granting Ticket ... 329

Subsequent Kerberos Authentications ... 330

Kerberos Remote Applications ... 332

Kerberos Principals ... 332

Kerberos Realms ... 333

Kerberos Servers ... 333

Kerberos Security Services ... 334

Components of Various Kerberos Releases ... 335

Kerberos Components ... 335

About Kerberos in The Oracle Solaris 11 Release ... 336

20 Planning for the Kerberos Service ... 339

Why Plan for Kerberos Deployments? ... 339

Planning Kerberos Realms ... 340

Realm Names ... 340

Number of Realms ... 340

Realm Hierarchy ... 341

Mapping Host Names Onto Realms ... 341

Client and Service Principal Names ... 341

Ports for the KDC and Admin Services ... 342

The Number of Slave KDCs ... 342
Mapping GSS Credentials to UNIX Credentials ... 343
Automatic User Migration to a Kerberos Realm ... 343
Which Database Propagation System to Use ... 344
Clock Synchronization Within a Realm ... 344
Client Configuration Options ... 344
Improving Client Login Security ... 345
KDC Configuration Options ... 345
Trusts of Services for Delegation ... 346
Kerberos Encryption Types ... 346
Online Help URL in the Graphical Kerberos Administration Tool ... 347

21 Configuring the Kerberos Service (Tasks) ... 349
Configuring the Kerberos Service (Task Map) ... 349
Configuring Additional Kerberos Services (Task Map) ... 350
Configuring KDC Servers ... 350
▼ How to Automatically Configure a Master KDC ... 351
▼ How to Interactively Configure a Master KDC ... 352
▼ How to Manually Configure a Master KDC ... 353
▼ How to Configure a KDC to Use an LDAP Data Server ... 358
▼ How to Automatically Configure a Slave KDC ... 364
▼ How to Interactively Configure a Slave KDC ... 365
▼ How to Manually Configure a Slave KDC ... 366
▼ How to Refresh the Ticket-Granting Service Keys on a Master Server ... 369
Configuring Cross-Realm Authentication ... 370
▼ How to Establish Hierarchical Cross-Realm Authentication ... 370
▼ How to Establish Direct Cross-Realm Authentication ... 371
Configuring Kerberos Network Application Servers ... 372
▼ How to Configure a Kerberos Network Application Server ... 373
▼ How to Use the Generic Security Service With Kerberos When Running FTP ... 374
Configuring Kerberos NFS Servers ... 375
▼ How to Configure Kerberos NFS Servers ... 376
▼ How to Create a Credential Table ... 378
▼ How to Add a Single Entry to the Credential Table ... 378
▼ How to Provide Credential Mapping Between Realms ... 379

▼ How to Set Up a Secure NFS Environment With Multiple Kerberos Security Modes 380
Configuring Kerberos Clients 381
Configuring Kerberos Clients (Task Map) 381
▼ How to Create a Kerberos Client Installation Profile 382
▼ How to Automatically Configure a Kerberos Client 383
▼ How to Interactively Configure a Kerberos Client 384
▼ How to Configure a Kerberos Client for an Active Directory Server 387
▼ How to Manually Configure a Kerberos Client 388
▼ How to Disable Verification of the Ticket-Granting Ticket 393
▼ How to Access a Kerberos Protected NFS File System as the `root` User 394
▼ How to Configure Automatic Migration of Users in a Kerberos Realm 395
▼ How to Configure Account Lockout 397
Synchronizing Clocks Between KDCs and Kerberos Clients 398
Swapping a Master KDC and a Slave KDC 399
▼ How to Configure a Swappable Slave KDC 399
▼ How to Swap a Master KDC and a Slave KDC 400
Administering the Kerberos Database 403
Backing Up and Propagating the Kerberos Database 403
▼ How to Back Up the Kerberos Database 405
▼ How to Restore the Kerberos Database 406
▼ How to Convert a Kerberos Database After a Server Upgrade 407
▼ How to Reconfigure a Master KDC to Use Incremental Propagation 407
▼ How to Reconfigure a Slave KDC to Use Incremental Propagation 409
▼ How to Configure a Slave KDC to Use Full Propagation 410
▼ How to Verify That the KDC Servers Are Synchronized 413
▼ How to Manually Propagate the Kerberos Database to the Slave KDCs 415
Setting Up Parallel Propagation 415
Configuration Steps for Setting Up Parallel Propagation 416
Administering the Stash File 417
▼ How to Remove a Stash File 417
▼ How to Employ a New Master Key 418
Managing a KDC on an LDAP Directory Server 419
▼ How to Mix Kerberos Principal Attributes in a Non-Kerberos Object Class Type 420
▼ How to Destroy a Realm on an LDAP Directory Server 421
Increasing Security on Kerberos Servers 421
▼ How to Enable Only Kerberized Applications 421

▼ How to Restrict Access to KDC Servers 422
▼ How to Use a Dictionary File to Increase Password Security 423

22 Kerberos Error Messages and Troubleshooting 425
Kerberos Error Messages 425
SEAM Tool Error Messages 425
Common Kerberos Error Messages (A-M) 426
Common Kerberos Error Messages (N-Z) 435
Kerberos Troubleshooting 439
▼ How to Identify Problems With Key Version Numbers 439
Problems With the Format of the `krb5.conf` File 440
Problems Propagating the Kerberos Database 440
Problems Mounting a Kerberized NFS File System 441
Problems Authenticating as the `root` User 441
Observing Mapping From GSS Credentials to UNIX Credentials 442
Using DTrace With the Kerberos Service 442

23 Administering Kerberos Principals and Policies (Tasks) 443
Ways to Administer Kerberos Principals and Policies 443
SEAM Tool 444
Command-Line Equivalents of the SEAM Tool 444
The Only File Modified by the SEAM Tool 445
Print and Online Help Features of the SEAM Tool 445
Working With Large Lists in the SEAM Tool 446
▼ How to Start the SEAM Tool 447
Administering Kerberos Principals 448
Administering Kerberos Principals (Task Map) 448
Automating the Creation of New Kerberos Principals 449
▼ How to View the List of Kerberos Principals 449
▼ How to View a Kerberos Principal's Attributes 451
▼ How to Create a New Kerberos Principal 453
▼ How to Duplicate a Kerberos Principal 456
▼ How to Modify a Kerberos Principal 456
▼ How to Delete a Kerberos Principal 457
▼ How to Set Up Defaults for Creating New Kerberos Principals 458

▼ How to Modify the Kerberos Administration Privileges 459
Administering Kerberos Policies 461
Administering Kerberos Policies (Task Map) 461
▼ How to View the List of Kerberos Policies 461
▼ How to View a Kerberos Policy's Attributes 463
▼ How to Create a New Kerberos Policy 465
▼ How to Duplicate a Kerberos Policy 467
▼ How to Modify a Kerberos Policy 467
▼ How to Delete a Kerberos Policy 468
SEAM Tool Reference 469
SEAM Tool Panel Descriptions 469
Using the SEAM Tool With Limited Kerberos Administration Privileges 472
Administering Keytab Files 473
Administering Keytab Files (Task Map) 474
▼ How to Add a Kerberos Service Principal to a Keytab File 474
▼ How to Remove a Service Principal From a Keytab File 476
▼ How to Display the Keylist (Principals) in a Keytab File 476
▼ How to Temporarily Disable Authentication for a Service on a Host 477

24 Using Kerberos Applications (Tasks) 481
Kerberos Ticket Management 481
Do You Need to Worry About Tickets? 481
Creating a Kerberos Ticket 482
Viewing Kerberos Tickets 483
Destroying Kerberos Tickets 484
Kerberos Password Management 485
Advice on Choosing a Password 485
Changing Your Password 486
Granting Access to Your Account 488
Kerberos User Commands 490
Overview of Kerberized Commands 490
Forwarding Kerberos Tickets 493
Using Kerberized Commands (Examples) 494

25 The Kerberos Service (Reference) 497
Kerberos Files 497
Kerberos Commands 499
Kerberos Daemons 500
Kerberos Terminology 500
  Kerberos-Specific Terminology 500
  Authentication-Specific Terminology 501
  Types of Tickets 502
How the Kerberos Authentication System Works 505
How the Kerberos Service Interacts With DNS and the `nsswitch` Service 506
Gaining Access to a Service Using Kerberos 506
  Obtaining a Credential for the Ticket-Granting Service 506
  Obtaining a Credential for a Server 507
  Obtaining Access to a Specific Service 508
Using Kerberos Encryption Types 509
Using the `gsscred` Table 511
Notable Differences Between Oracle Solaris Kerberos and MIT Kerberos 511

Part VII Auditing in Oracle Solaris 513

26 Auditing (Overview) 515
What Is Auditing? 515
Audit Terminology and Concepts 516
  Audit Events 518
  Audit Classes and Preselection 519
  Audit Records and Audit Tokens 520
  Audit Plugin Modules 520
  Audit Logs 521
  Storing and Managing the Audit Trail 523
  Ensuring Reliable Time Stamps 523
  Managing a Remote Repository 524
How Is Auditing Related to Security? 524
How Does Auditing Work? 524
How Is Auditing Configured? 525
Auditing on a System With Oracle Solaris Zones 526

About the Audit Service in This Release 527

27 Planning for Auditing 529

Planning Auditing (Tasks) 529

▼ How to Plan Auditing in Zones 530

▼ How to Plan Storage for Audit Records 531

▼ How to Plan Who and What to Audit 532

Understanding Audit Policy 534

Controlling Auditing Costs 537

Cost of Increased Processing Time of Audit Data 537

Cost of Analysis of Audit Data 537

Cost of Storage of Audit Data 537

Auditing Efficiently 538

28 Managing Auditing (Tasks) 541

Managing Auditing (Task Map) 541

Configuring the Audit Service (Tasks) 542

Configuring the Audit Service (Task Map) 542

▼ How to Display Audit Service Defaults 543

▼ How to Preselect Audit Classes 545

▼ How to Configure a User's Audit Characteristics 546

▼ How to Change Audit Policy 549

▼ How to Change Audit Queue Controls 551

▼ How to Configure the `audit_warn` Email Alias 553

▼ How to Add an Audit Class 554

▼ How to Change an Audit Event's Class Membership 555

Configuring Audit Logs (Tasks) 556

Configuring Audit Logs (Task Map) 556

▼ How to Create ZFS File Systems for Audit Files 556

▼ How to Assign Audit Space for the Audit Trail 559

▼ How to Send Audit Files to a Remote Repository 562

▼ How to Configure `syslog` Audit Logs 563

Configuring the Audit Service in Zones (Tasks) 565

▼ How to Configure All Zones Identically for Auditing 565

▼ How to Configure Per-Zone Auditing 567

Enabling and Disabling the Audit Service (Tasks) ... 568
- ▼ How to Refresh the Audit Service ... 568
- ▼ How to Disable the Audit Service ... 571
- ▼ How to Enable the Audit Service ... 571

Managing Audit Records on Local Systems (Tasks) ... 572
- Managing Audit Records on Local Systems (Task Map) ... 572
- ▼ How to Display Audit Record Definitions ... 573
- ▼ How to Merge Audit Files From the Audit Trail ... 574
- ▼ How to Select Audit Events From the Audit Trail ... 576
- ▼ How to View the Contents of Binary Audit Files ... 578
- ▼ How to Clean Up a `not_terminated` Audit File ... 580
- ▼ How to Prevent Audit Trail Overflow ... 581

Troubleshooting the Audit Service (Tasks) ... 582
- Troubleshooting the Audit Service (Task Map) ... 582
- ▼ How to Determine That Auditing Is Running ... 583
- ▼ How to Lessen the Volume of Audit Records That Are Produced ... 585
- ▼ How to Audit All Commands by Users ... 587
- ▼ How to Find Audit Records of Changes to Specific Files ... 589
- ▼ How to Update the Preselection Mask of Logged In Users ... 591
- ▼ How to Prevent the Auditing of Specific Events ... 592
- ▼ How to Limit the Size of Binary Audit Files ... 593
- ▼ How to Compress Audit Files on a Dedicated File System ... 593
- ▼ How to Audit Logins From Other Operating Systems ... 594
- ▼ How to Audit FTP and SFTP File Transfers ... 595

29 Auditing (Reference) ... 597

Audit Service ... 597

Audit Service Man Pages ... 598

Rights Profiles for Administering Auditing ... 600

Auditing and Oracle Solaris Zones ... 600

Audit Classes ... 601
- Audit Class Syntax ... 601

Audit Plugins ... 602

Audit Policy ... 603
- Audit Policies for Asynchronous and Synchronous Events ... 603

Process Audit Characteristics 604
Audit Trail 605
Conventions for Binary Audit File Names 605
Audit Record Structure 606
Audit Record Analysis 606
Audit Token Formats 607
acl Token 609
argument Token 609
attribute Token 609
cmd Token 609
exec_args Token 610
exec_env Token 610
file Token 610
fmri Token 611
group Token 611
header Token 611
ip address Token 612
ip port Token 612
ipc Token 612
IPC_perm Token 613
path Token 613
path_attr Token 613
privilege Token 614
process Token 614
return Token 614
sequence Token 614
socket Token 615
subject Token 615
text Token 615
trailer Token 616
use of authorization Token 616
use of privilege Token 616
user Token 616
xclient Token 617
zonename Token 617

Glossary 619

Index 631

Preface

System Administration Guide: Security Services is part of a multivolume set that covers a significant part of the Oracle Solaris operating system (Oracle Solaris OS) administration information. This book assumes that you have already installed the latest release, and you have set up any networking software that you plan to use. The Oracle Solaris OS is part of the Oracle Solaris product family, which includes many features, such as Secure Shell.

Note – This Oracle Solaris release supports systems that use the SPARC and x86 families of processor architectures. The supported systems appear in the *Oracle Solaris OS: Hardware Compatibility Lists*. This document cites any implementation differences between the platform types.

Who Should Use This Book

This book is intended for anyone who is responsible for administering one or more systems that run Oracle Solaris. 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 Guides Are Organized

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

Book Title	Topics
Booting and Shutting Down Oracle Solaris on SPARC Platforms	Booting and shutting down a system, managing boot services, modifying boot behavior, booting from ZFS, managing the boot archive, and troubleshooting booting on SPARC platforms
Booting and Shutting Down Oracle Solaris on x86 Platforms	Booting and shutting down a system, managing boot services, modifying boot behavior, booting from ZFS, managing the boot archive, and troubleshooting booting on x86 platforms

Book Title	Topics
Oracle Solaris Administration: Common Tasks	Using Oracle Solaris commands, booting and shutting down a system, managing user accounts and groups, managing services, hardware faults, system information, system resources, and system performance, managing software, printing, the console and terminals, and troubleshooting system and software problems
Oracle Solaris Administration: Devices and File Systems	Removable media, disks and devices, file systems, and backing up and restoring data
Oracle Solaris Administration: IP Services	TCP/IP network administration, IPv4 and IPv6 address administration, DHCP, IPsec, IKE, IP Filter, and IPQoS
Oracle Solaris Administration: Naming and Directory Services	DNS, NIS, and LDAP naming and directory services, including transitioning from NIS to LDAP
Oracle Solaris Administration: Network Interfaces and Network Virtualization	Automatic and manual IP interface configuration including WiFi wireless; administration of bridges, VLANs, aggregations, LLDP, and IPMP; virtual NICs and resource management.
Oracle Solaris Administration: Network Services	Web cache servers, time-related services, network file systems (NFS and autofs), mail, SLP, and PPP
Oracle Solaris Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management	Resource management features, which enable you to control how applications use available system resources; Oracle Solaris Zones software partitioning technology, which virtualizes operating system services to create an isolated environment for running applications; and Oracle Solaris 10 Zones, which host Oracle Solaris 10 environments running on the Oracle Solaris 11 kernel
Oracle Solaris Administration: Security Services	Auditing, device management, file security, BART, Kerberos services, PAM, Cryptographic Framework, Key Management, privileges, RBAC, SASL, Secure Shell, and virus scanning
Oracle Solaris Administration: SMB and Windows Interoperability	SMB service, which enables you to configure an Oracle Solaris system to make SMB shares available to SMB clients; SMB client, which enables you to access SMB shares; and native identity mapping services, which enables you to map user and group identities between Oracle Solaris systems and Windows systems
Oracle Solaris Administration: ZFS File Systems	ZFS storage pool and file system creation and management, snapshots, clones, backups, using access control lists (ACLs) to protect ZFS files, and using ZFS on an Oracle Solaris system with zones installed
Trusted Extensions Configuration and Administration	System installation, configuration, and administration that is specific to Trusted Extensions
Oracle Solaris 11 Security Guidelines	Securing an Oracle Solaris system, as well as usage scenarios for its security features, such as zones, ZFS, and Trusted Extensions

Book Title	Topics
Transitioning From Oracle Solaris 10 to Oracle Solaris 11	Provides system administration information and examples for transitioning from Oracle Solaris 10 to Oracle Solaris 11 in the areas of installation, device, disk, and file system management, software management, networking, system management, security, virtualization, desktop features, user account management, and user environments emulated volumes, and troubleshooting and data recovery

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Typographic Conventions

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

TABLE P–1 Typographic Conventions

Typeface	Description	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`

`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`**
		`Password:`

aabbcc123	Placeholder: replace with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* save the file. **Note:** Some emphasized items appear bold online.

Shell Prompts in Command Examples

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

TABLE P–2 Shell Prompts

Shell	Prompt
Bash shell, Korn shell, and Bourne shell	`$`
Bash shell, Korn shell, and Bourne shell for superuser	`#`
C shell	`machine_name%`
C shell for superuser	`machine_name#`

PART I

Security Overview

This book focuses on the features that enhance security in the Oracle Solaris OS. This book is intended for system administrators and users of these security features. Chapter 1, “Security Services (Overview),” introduces the topics in the book.

CHAPTER 1

Security Services (Overview)

To maintain the security of the Oracle Solaris OS, the software provides the following features:

- "System Security" on page 29 – 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
- "Cryptographic Services" on page 30 – 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 31 – 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 32 – The ability to ensure that authenticated parties can communicate without interception, modification, or spoofing
- "Auditing" on page 32 – 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 32 – The design and implementation of security guidelines for a system or network of systems

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. Oracle Solaris 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 56.
- **Hardware access –** Commands for limiting access to the PROM, and for restricting who can boot the system. See "Controlling Access to System Hardware (Tasks)" on page 66.

- **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 45.

  For the management of resources in Oracle Solaris Zones, see Part I, "Oracle Solaris Resource Management," in *Oracle Solaris Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management*.
- **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 133.
- **Privileges –** Discrete rights on processes to perform operations. These process rights are enforced in the kernel. See "Privileges (Overview)" on page 145.
- **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 42.
- **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)."

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 Framework –** A central framework of cryptographic services for kernel-level and user-level consumers that is based on the following standard: RSA Security Inc. PKCS #11 Cryptographic Token Interface (Cryptoki). Uses include passwords, IPsec, and third-party applications. The framework centralizes hardware and software sources for encryption. The PKCS #11 library provides an API for third-party developers to plug in the cryptographic requirements for their applications. See Chapter 11, "Cryptographic Framework (Overview)."

- **Encryption mechanisms per application –**
  - For the use of DES in Secure RPC, see "Overview of Secure RPC" on page 267.
  - For the use of DES, 3DES, AES, and ARCFOUR in the Kerberos service, see Chapter 19, "Introduction to the Kerberos Service."
  - For the use of RSA, DSA, and ciphers such as Blowfish in Secure Shell, see Chapter 17, "Using Secure Shell (Tasks)."
  - For the use of cryptographic algorithms in passwords, see "Changing the Default Algorithm for Password Encryption (Tasks)" on page 61.
- 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 13, "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 token 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 Oracle Solaris features for authentication include the following:

- **Secure RPC –** An authentication mechanism that uses the Diffie-Hellman protocol to protect NFS mounts and a naming service, such as NIS. See "Overview of Secure RPC" on page 267.
- **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 15, "Using PAM."
- **Simple Authentication and Security Layer (SASL) –** A framework that provides authentication and security services to network protocols. See Chapter 16, "Using SASL."
- **Secure Shell –** A secure remote login and transfer protocol that encrypts communications over an insecure network. See Chapter 17, "Using Secure Shell (Tasks)."
- **Kerberos service –** A client-server architecture that provides encryption with authentication. See Part VI, "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 Oracle Solaris features for secure communication include the following:

- **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 17, "Using Secure Shell (Tasks)."
- **Kerberos service –** A client-server architecture that provides authentication with encryption. See Part VI, "Kerberos Service."
- **Internet Protocol Security Architecture (IPsec) –** An architecture that provides IP datagram protection. Protections include confidentiality, strong integrity of the data, data authentication, and partial sequence integrity. See Part III, "IP Security," in *Oracle Solaris Administration: IP Services*.

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 Part VII, "Auditing in Oracle Solaris."

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 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, Oracle Solaris uses audit policy options to configure some aspects of audit policy. The following table points to glossary, man page, and information about features that use the word policy to describe specific aspects of their implementation.

TABLE 1–1 Use of the Word "Policy" in Oracle Solaris

"Policy" Term	Selected Man Pages	Further Information
audit policy	`auditconfig`(1M)	Chapter 26, "Auditing (Overview)"
policy in the Cryptographic Framework	`cryptoadm`(1M)	Chapter 11, "Cryptographic Framework (Overview)"
device policy	`getdevpolicy`(1M)	"Controlling Access to Devices" on page 42
Kerberos policy	`krb5.conf`(4)	Chapter 23, "Administering Kerberos Principals and Policies (Tasks)"
network policies	`ipfilter`(5), `ipadm`(1M), `ike.config`(4), `ipsecconf`(1M), `routeadm`(1M)	Part III, "IP Security," in *Oracle Solaris Administration: IP Services*
password policy	`passwd`(1), `crypt.conf`(4), `policy.conf`(4)	"Maintaining Login Control" on page 38
policy for public key technologies	`kmfcfg`(1)	Chapter 13, "Key Management Framework"
RBAC policy	`rbac`(5), `policy.conf`(4)	"`policy.conf` File" on page 203

PART II

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.

- Chapter 2, “Managing Machine Security (Overview)”
- Chapter 3, “Controlling Access to Systems (Tasks)”
- Chapter 4, “Virus Scanning Service (Tasks)”
- Chapter 5, “Controlling Access to Devices (Tasks)”
- Chapter 6, “Using the Basic Audit Reporting Tool (Tasks)”
- Chapter 7, “Controlling Access to Files (Tasks)”

CHAPTER 2

Managing Machine Security (Overview)

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 37
- "Controlling Access to Devices" on page 42
- "Controlling Access to Machine Resources" on page 45
- "Controlling Access to Files" on page 48
- "Controlling Network Access" on page 50
- "Reporting Security Problems" on page 54

Controlling Access to a Computer System

In the workplace, all computers 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. You also need to ensure the integrity of the data on the computers within the network.

At the file level, Oracle Solaris 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 38
- "Maintaining Login Control" on page 38
- "Controlling Access to Devices" on page 42
- "Controlling Access to Machine Resources" on page 45
- "Controlling Access to Files" on page 48
- "Controlling Network Access" on page 50

- “Reporting Security Problems” on page 54

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 must 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 66. To protect x86 hardware, consult the vendor documentation.

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 must 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 naming service or directory service database according to the information in the name switch service, `svc:/system/name-service/switch`. The following databases can affect login:

- `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
- `dns` – Designates the domain name service on the network

For a description of the naming service, see the `nscd`(1M) man page. For information about naming services and directory services, see the *Oracle Solaris Administration: Naming and Directory Services*.

The `login` command verifies the user name and password that were supplied by the user. If the user name is not in the password database, 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 15, “Using PAM.”

Sophisticated authentication and authorization mechanisms are available on Oracle Solaris systems. For a discussion of authentication and authorization mechanisms at the network level, see "Authentication and Authorization for Remote Access" on page 51.

Managing Password Information

When users log in to a system, they must supply both a user name and a password. Although logins are publicly known, passwords must be kept secret. Passwords should be known only to each user. Users must choose their passwords carefully and change them 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 2, "Managing User Accounts and Groups (Overview)," in *Oracle Solaris Administration: Common Tasks* 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 `/etc/passwd` file. 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 can read the `/etc/shadow` file. You can use the `passwd` command to change a user's password on a local system.

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.

LDAP Passwords

The Oracle Solaris LDAP naming service stores password information and shadow information in the `ou=people` container of the LDAP directory tree. On the Oracle 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.

Password policy is enforced on the Oracle Directory Server Enterprise Edition. Specifically, the client's `pam_ldap` module follows the password policy controls that are enforced on the Oracle Directory Server Enterprise Edition. For more information, see "LDAP Naming Services Security Model" in *Oracle Solaris Administration: Naming and Directory Services*.

Password Encryption

Strong password encryption provides an early barrier against attack. Oracle Solaris software provides six password encryption algorithms. The Blowfish, MD5, and SHA algorithms 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. For the identifier-algorithm mapping, see the /etc/security/crypt.conf file.

TABLE 2–1 Password Encryption Algorithms

Identifier	Description	Algorithm Man Page
1	The MD5 algorithm that is compatible with MD5 algorithms on BSD and Linux systems.	crypt_bsdmd5(5)
2a	The Blowfish algorithm that is compatible with the Blowfish algorithm on BSD systems.	crypt_bsdbf(5)
md5	The Sun MD5 algorithm, which is considered stronger than the BSD and Linux version of MD5.	crypt_sunmd5(5)
5	The SHA256 algorithm. SHA stands for Secure Hash Algorithm. This algorithm is a member of the SHA-2 family. SHA256 supports 255-character passwords.	crypt_sha256(5)
6	The SHA512 algorithm.	crypt_sha512(5)
__unix__	The traditional UNIX encryption algorithm.	crypt_unix(5)

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
# be used for new passwords.  This is enforced only in crypt_gensalt(3c).
#
CRYPT_ALGORITHMS_ALLOW=1,2a,md5,5,6

# To deprecate use of the traditional unix algorithm, uncomment below
# and change CRYPT_DEFAULT= to another algorithm.  For example,
# CRYPT_DEFAULT=1 for BSD/Linux MD5.
#
#CRYPT_ALGORITHMS_DEPRECATE=__unix__
```

```
# The Oracle Solaris default is a SHA256 based algorithm.  To revert to
# the policy present in Solaris releases set CRYPT_DEFAULT=__unix__,
# which is not listed in crypt.conf(4) since it is internal to libc.
#
CRYPT_DEFAULT=5
...
```

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 existing 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.

Identifier = Password Algorithm		
Initial Password	**Changed Password**	**Explanation**
1 = crypt_bsdmd5	Uses same algorithm	The 1 identifier is also the value of CRYPT_DEFAULT. The user's password continues to be encrypted with the crypt_bsdmd5 algorithm.
2a = crypt_bsdbf	Uses same algorithm	The 2a identifier is in the CRYPT_ALGORITHMS_ALLOW list. Therefore, the new password is encrypted with the crypt_bsbdf algorithm.
md5 = crypt_md5	Uses same algorithm	The md5 identifier is in the CRYPT_ALGORITHMS_ALLOW list. Therefore, the new password is encrypted with the crypt_md5 algorithm.
5 = crypt_sha256	Uses same algorithm	The 5 identifier is in the CRYPT_ALGORITHMS_ALLOW list. Therefore, the new password is encrypted with the crypt_sha256 algorithm.
6 = crypt_sha512	Uses same algorithm	The 6 identifier is in the CRYPT_ALGORITHMS_ALLOW list. Therefore, the new password is encrypted with the crypt_sha512 algorithm.
__unix__ = crypt_unix	Uses crypt_bsdmd5 algorithm	The __unix__ 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.

For more information about configuring the algorithm choices, see the policy.conf(4) man page. To specify password encryption algorithms, see "Changing the Default Algorithm for Password Encryption (Tasks)" on page 61.

Special System Accounts

The root account is one of several special *system* accounts. Of these accounts, only the root account is assigned a password and can log in. The nuucp account can log in for file transfers. The other system accounts either protect files or run administrative processes without using the full powers of root.

Caution – Never change the password setting of a system account. System accounts from Oracle Solaris are delivered in a safe and secure state.

The following table lists some system accounts and their uses. The system accounts perform special functions. Each account has a UID that is less than 100.

TABLE 2–2 System Accounts and Their Uses

System Account	UID	Use
root	0	Has almost no restrictions. Can override other protections and permissions. The root account has access to the entire system. The password for the root login should be very carefully protected. The root account, owns most of the Oracle Solaris commands.
daemon	1	Controls background processing.
bin	2	Owns some Oracle Solaris commands.
sys	3	Owns many system files.
adm	4	Owns some administrative files.
lp	71	Owns the object data files and spooled data files for the printer.
uucp	5	Owns the object data files and spooled data files for UUCP, the UNIX-to-UNIX copy program.
nuucp	9	Is used by remote systems to log in to the system and start file transfers.

Remote Logins

Remote logins offer a tempting avenue for intruders. Oracle Solaris provides several commands to monitor, limit, and disable remote logins. For procedures, see "Securing Logins and Passwords (Task Map)" on page 56.

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.

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 bge0 are considered integral devices.

Oracle 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 `bge` 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.

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

- **Set device policy –** In Oracle Solaris 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, Oracle Solaris software configures device policy. Third-party drivers can be configured with device policy during installation. After installation, you, as the 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 *Oracle Solaris Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management*. For a more general discussion of devices and zones, see "Configured Devices in Zones" in *Oracle Solaris Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management*.

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. Oracle Solaris provides default device policy. For example, network interfaces such as `bge0` 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 145.

In earlier 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`. Now, 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/ip` 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_MODDEVPLCY` audit event records changes in device policy.

For more information about device policy, see the following:

- "Configuring Device Policy (Task Map)" on page 76
- "Device Policy Commands" on page 88
- "Privileges and Devices" on page 152

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 94.

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

For more information about device allocation, see the following:

- "Managing Device Allocation (Task Map)" on page 79
- "Device Allocation" on page 88
- "Device Allocation Commands" on page 90

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 systems 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 (Tasks)" on page 64.

Configuring Role-Based Access Control to Replace Superuser

Role-based access control (RBAC), a feature of Oracle Solaris, is designed to distribute the capabilities of superuser to administrative roles. 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 133.

Preventing Unintentional Misuse of System 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 your initialization files, such as .bashrc and /etc/profile. 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 the root account 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 a 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 Oracle Solaris 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` 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 Oracle 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 must restrict their use. Secure administration requires few `setuid` programs.

For more information, see "Protecting Executable Files From Compromising Security" on page 121. For procedures, see "Protecting Against Programs With Security Risk (Task Map)" on page 127.

Using the Secure by Default Configuration

By default, when Oracle Solaris is installed, a large set of network services are disabled. This configuration is called "Secure by Default" (SBD). 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 Service Management Facility (SMF) feature of Oracle Solaris. For more information, see the `netservices`(1M) and `smf`(5) man pages.

Using Resource Management Features

Oracle 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 Oracle 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, "Oracle Solaris Resource Management," in *Oracle Solaris Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management*.

Using Oracle Solaris Zones

Oracle Solaris zones provide an application execution environment in which processes are isolated from the rest of the system within a single instance of the Oracle 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.

Oracle Solaris zones are ideal for environments that place several applications on a single server. For more information, see Part II, "Oracle Solaris Zones," in *Oracle Solaris Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management*.

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 about the audit service, see Chapter 26, "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

Oracle Solaris is a multiuser environment. In a multiuser environment, all the users who are logged in to 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 (Tasks)" on page 121.

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 Cryptographic Framework provides digest, mac, and encrypt commands to protect files. For more information, see Chapter 11, “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 fine-grained file permissions, including the following:

- 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 UFS Files” on page 119. To protect ZFS files with access control lists (ACLs), see Chapter 8, “Using ACLs and Attributes to Protect Oracle Solaris ZFS Files,” in *Oracle Solaris Administration: ZFS File Systems*.

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.

When you create an NFS share of a ZFS file system, the file system is permanently shared until you remove the share. SMF automatically manages the share when the system is rebooted. For more information, see Chapter 3, “Oracle Solaris ZFS and Traditional File System Differences,” in *Oracle Solaris Administration: ZFS File Systems*.

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 `root` access to 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 *Oracle Solaris Administration: Network Services*.

Controlling Network Access

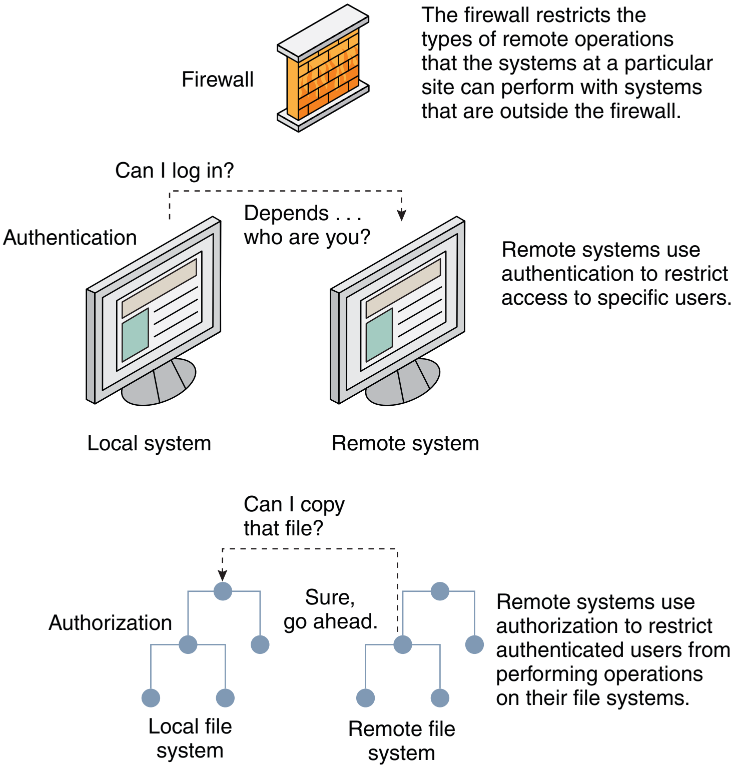
Computers are often part of a *network* of computers. A network allows connected computers to exchange information. Networked computers can access data and other resources from other computers on the network. Computer networks create a powerful and sophisticated computing environment. However, networks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

For example, within a network of computers, individual systems 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.

FIGURE 2–1 Security Restrictions for Remote Operations


Firewall
The firewall restricts the types of remote operations that the systems at a particular site can perform with systems that are outside the firewall.
Can I log in?
Authentication
Depends . . . who are you?
Remote systems use authentication to restrict access to specific users.
Local system
Remote system
Can I copy that file?
Authorization
Sure, go ahead.
Remote systems use authorization to restrict authenticated users from performing operations on their file systems.
Local file system
Remote file system

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 2–3 Authentication Services for Remote Access

Service	Description	For More Information
IPsec	IPsec provides host-based and certificate-based authentication and network traffic encryption.	Chapter 14, "IP Security Architecture (Overview)," in *Oracle Solaris Administration: IP Services*
Kerberos	Kerberos uses encryption to authenticate and authorize a user who is logging in to the system.	For an example, see "How the Kerberos Service Works" on page 328.

TABLE 2–3 Authentication Services for Remote Access *(Continued)*

Service	Description	For More Information
LDAP	The LDAP directory service can provide both authentication and authorization at the network level.	*Oracle Solaris Administration: Naming and Directory Services*
Remote login commands	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 "trusted host," authentication is automatic. Otherwise, you are asked to authenticate yourself.	Chapter 29, "Accessing Remote Systems (Tasks)," in *Oracle Solaris Administration: Network Services*
SASL	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.	"SASL (Overview)" on page 289
Secure RPC	Secure RPC improves the security of network environments by authenticating users who make requests on remote machines. You can use either a UNIX, DES, or Kerberos authentication system for Secure RPC.	"Overview of Secure RPC" on page 267
	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.	"NFS Services and Secure RPC" on page 267
Secure Shell	Secure Shell encrypts network traffic over an unsecured network. Secure Shell provides authentication by the use of passwords, public keys, or both. Secure Shell uses RSA and DSA authentication for public keys.	"Secure Shell (Overview)" on page 293

A possible substitute for Secure RPC is the Oracle 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 Oracle 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 appears similar to the following:

```
"Weak Authentication
NFS request from unprivileged port"
```

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.

IPsec and the IP Filter feature of Oracle Solaris can provide firewall protection. For more information about protecting network traffic, see Part III, "IP Security," in *Oracle Solaris Administration: IP Services*.

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 17, "Using Secure Shell (Tasks)."
- To encrypt and authenticate data across a network, see Chapter 19, "Introduction to the Kerberos Service."
- To encrypt IP datagrams, see Chapter 14, "IP Security Architecture (Overview)," in *Oracle Solaris Administration: IP Services*.

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. For current contact information, consult the CERT/CC (`http://www.cert.org/contact_cert/`) web site.

CHAPTER 3

Controlling Access to Systems (Tasks)

This chapter describes the procedures for controlling who can access Oracle Solaris systems.

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

- "Controlling System Access (Task Map)" on page 55
- "Securing Logins and Passwords (Tasks)" on page 56
- "Changing the Default Algorithm for Password Encryption (Tasks)" on page 61
- "Monitoring and Restricting Superuser (Tasks)" on page 64
- "Controlling Access to System Hardware (Tasks)" on page 66

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 must monitor and secure.

Task	Description	For Instructions
Monitor, permit, and deny user login.	Monitors unusual login activity. Prevents logins temporarily.	"Securing Logins and Passwords (Task Map)" on page 56
Provide strong password encryption.	Specifies algorithms to encrypt user passwords. Installs additional algorithms.	"Changing the Default Algorithm for Password Encryption (Tasks)" on page 61
Monitor and restrict superuser activities.	Regularly monitors superuser activity. Prevents remote login by a `root` user.	"Monitoring and Restricting Superuser (Tasks)" on page 64
Prevent access to hardware settings.	Keeps regular users away from the PROM.	"Controlling Access to System Hardware (Tasks)" on page 66

Securing Logins and Passwords (Tasks)

You can limit remote logins, require users to have passwords, and require the root account to have a complex password. You can also monitor failed access attempts and disable logins temporarily.

Securing Logins and Passwords (Task Map)

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

Task	Description	For Instructions
Change the root password.	Ensures that the root account complies with password requirements.	"How to Change the root Password" on page 56
Display a user's login status.	Lists extensive information about a user's login account, such as full name and password aging information.	"How to Display a User's Login Status" on page 57
Find users who do not have passwords.	Finds only those users whose accounts do not require a password.	"How to Display Users Without Passwords" on page 58
Disable logins temporarily.	Denies user logins to a machine as part of system shutdown or routine maintenance.	"How to Temporarily Disable User Logins" on page 58
Save failed login attempts.	Creates a log of users who failed to provide the correct password after five attempts.	"How to Monitor Failed Login Attempts" on page 59
Save all failed login attempts.	Creates a log of failed attempts to log in.	"How to Monitor All Failed Login Attempts" on page 60

▼ How to Change the root Password

When you change the root password, you must comply with the password requirements that apply to all users of the system.

Before You Begin You must be in the root role.

- **Change your password.**

```
# passwd root
New Password:
Re-enter new Password:
passwd: password successfully changed for root
```

A message prints to the screen if your password does not conform to requirements. The messages are informative. After three attempts, you must run the command again to change the password.

```
passwd: Password too short - must be at least 6 characters.
passwd: The password must contain at least 2 alphabetic character(s).
passwd: The password must contain at least 1 numeric or special character(s).
```

▼ How to Display a User's Login Status

Before You Begin You must be in the `root` role.

● **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 are separated by commas.

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 naming 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 `jdoe` is displayed.

```
# logins -x -l jdoe
jdoe       500     staff           10   Jaylee Jaye Doe
                   /home/jdoe
                   /bin/bash
                   PS 010103 10 7 -1
```

`jdoe` 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).

Jaylee Jaye Doe Identifies the comment.

`/home/jdoe` Identifies the user's home directory.

`/bin/bash` 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

Before You Begin You must be in the root role.

- **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 passwd database from the local system unless a distributed naming service is specified in the nsswitch.conf file.

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.

Before You Begin You must be in the root role.

1. **Create the /etc/nologin file in a text editor.**

```
# vi /etc/nologin
```

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

3. **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 about bringing the system to single-user mode, see Chapter 3, "Shutting Down a System (Tasks)," in *Booting and Shutting Down Oracle Solaris on x86 Platforms*.

▼ How to Monitor Failed Login Attempts

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

Before You Begin You must be in the root role.

1. **Create the loginlog file in the /var/adm directory.**

```
# touch /var/adm/loginlog
```

2. **Set read-write permissions for root user on the loginlog file.**

```
# chmod 600 /var/adm/loginlog
```

3. **Change group membership to sys on the loginlog file.**

```
# chgrp sys /var/adm/loginlog
```

4. **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 2010
jdoe:/dev/pts/2:Tue Nov  4 10:21:21 2010
jdoe:/dev/pts/2:Tue Nov  4 10:21:30 2010
jdoe:/dev/pts/2:Tue Nov  4 10:21:40 2010
jdoe:/dev/pts/2:Tue Nov  4 10:21:49 2010
#
```

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.

Before You Begin You must be in the `root` role.

1 **Set up the `/etc/default/login` file with the desired values for `SYSLOG` and `SYSLOG_FAILED_LOGINS`**

Edit the `/etc/default/login` file to change the entry. Make sure that **`SYSLOG=YES`** is uncommented.

```
# grep SYSLOG /etc/default/login
# SYSLOG determines whether the syslog(3) LOG_AUTH facility should be used
SYSLOG=YES
# The SYSLOG_FAILED_LOGINS variable is used to determine how many failed
#SYSLOG_FAILED_LOGINS=5
SYSLOG_FAILED_LOGINS=0
#
```

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

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

```
# touch /var/adm/authlog
```

b. **Set read-and-write permissions for `root` user on the `authlog` file.**

```
# chmod 600 /var/adm/authlog
```

c. **Change group membership to `sys` on the `authlog` file.**

```
# chgrp sys /var/adm/authlog
```

3 **Edit the `syslog.conf` file to log failed password attempts.**

Send the failures 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.

```
auth.notice     <Press Tab>   /var/adm/authlog
```

b. **Refresh the `system-log` service.**

```
# svcadm refresh system/system-log
```

4 **Verify that the log works.**

For example, as an regular user, log in to the system with the wrong password. Then, 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
#
```

5 **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.

Changing the Default Algorithm for Password Encryption (Tasks)

By default, user passwords are encrypted with the crypt_sha256 algorithm. You can use a different encryption algorithm, 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 systems that run the Oracle Solaris, BSD, and Linux versions of UNIX. For a list of password encryption algorithms and algorithm identifiers, see Table 2–1.

Before You Begin You must be in the root role.

● **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 (5) that works with Linux and BSD systems.
# Passwords previously encrypted with SHA256 (1) will be encrypted
# with MD5 when users change their passwords.
#
#
#CRYPT_DEFAULT=5
CRYPT_DEFAULT=1
```

In this example, the algorithms configuration ensures that the sha256 algorithm is not used to encrypt a password. Users whose passwords were encrypted with the sha256 module get a crypt_bsdmd5-encrypted password when they change their passwords.

For more information about configuring the algorithm choices, see the policy.conf(4) man page.

Example 3–6 Constraining Password Encryption Algorithms in a Heterogeneous Environment

In this example, the administrator on a network that includes BSD and Linux systems configures passwords to be usable on all systems. Because some network applications cannot handle SHA512 encryption, the administrator does not include its identifier in the list of allowed algorithms. The administrator retains the SHA256 algorithm, 5, as the value for the CRYPT_DEFAULT variable. The CRYPT_ALGORITHMS_ALLOW variable contains the MD5 identifier, which is compatible with BSD and Linux systems, and the Blowfish identifier, which is compatible with BSD systems. Because 5 is the CRYPT_DEFAULT algorithm, it does not need to be listed in the CRYPT_ALGORITHMS_ALLOW list. However, for maintenance purposes, the administrator places 5 in the CRYPT_ALGORITHMS_ALLOW list and the unused identifiers in the CRYPT_ALGORITHMS_DEPRECATE list.

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

▼ 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 system encrypts the password.

Before You Begin You must be in the root role.

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 system 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 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.

Before You Begin You must be in the `root` role.

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 system 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 `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 system 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 Oracle Directory Server Enterprise Edition's cryptographic mechanism. For the procedure, see Chapter 11, "Setting Up Oracle Directory Server Enterprise Edition With LDAP Clients (Tasks)," in *Oracle Solaris Administration: Naming and Directory Services*.

Monitoring and Restricting Superuser (Tasks)

An alternative to using the superuser account is to set up role-based access control (RBAC). For overview information about RBAC, see "Role-Based Access Control (Overview)" on page 133. 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.

Before You Begin You must be in the `root` role.

● **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 jdoe-root
SU 01/12 14:57 + pts/0 jdoe-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 `ttynam` 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`. A Trusted Extensions user might have switched to the `oracle` role by using a GUI.

▼ How to Restrict and Monitor Superuser Logins

This method immediately detects `root` attempts to access the local system.

Before You Begin You must be in the `root` role.

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 `root`.

```
mach2 % ssh -l root mach1
Password:       <Type root password of mach1>
Password:
Password:
Permission denied (gssapi-keyex,gssapi-with-mic,publickey,keyboard-interactive).
```

In the default configuration, `root` is a role, and roles cannot log in. Also, in the default configuration the `ssh` protocol prevents `root` user login.

3 **Monitor attempts to become `root`.**

By default, attempts to become `root` 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 then can 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.

Controlling Access to System Hardware (Tasks)

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

To protect the BIOS, consult the vendor documentation.

▼ How to Require a Password for Hardware Access

Before You Begin You must be assigned the Device Security, Maintenance and Repair, or System Administrator rights profile.

1 **Become an administrator with the required security attributes.**

For more information, see "How to Obtain Administrative Rights" on page 160.

2 **In a terminal window, type the PROM security mode.**

```
# 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 eeprom(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:**

```
# 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

Note – 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.

Before You Begin You must be in the root role.

1 **Change the value of KEYBOARD_ABORT to disable.**

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

```
# 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.
...
```

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

2 **Update the keyboard defaults.**

```
# kbd -i
```

CHAPTER 4

Virus Scanning Service (Tasks)

This chapter provides information about using antivirus software, and covers the following topics:

- "About Virus Scanning" on page 69
- "About the Vscan Service" on page 70
- "Using the Vscan Service (Tasks)" on page 70

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 4–1 Antivirus Scan Engine Software

Antivirus Software	ICAP Support
Symantec Antivirus Scan Engine 4.3	Is supported
Symantec Antivirus Scan Engine 5.1	Is supported

TABLE 4–1 Antivirus Scan Engine Software *(Continued)*

Antivirus Software	ICAP Support
Computer Associates eTrust AntiVirus 7.1 Computer Associates Integrated Threat Management 8.1	Is not supported[1]
Trend Micro Interscan Web Security Suite (IWSS) 2.5	Is supported
McAfee Secure Internet Gateway 4.5	Is supported

[1] Requires installation of the Sun StorageTek 5000 NAS ICAP Server for Computer Associates Antivirus Scan Engine. Get the package from the Sun Download Center: (`http://www.oracle.com/technetwork/indexes/downloads/index.html`).

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 (Tasks)

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 vscan service.

Task	Description	For Instructions
Install a scan engine.	Installs and configures one or more of the supported third-party products listed in Table 4–1.	See the product documentation.
Enable the file system to allow virus scans.	Enables virus scans on a ZFS file system. By default, scans are disabled.	"How to Enable Virus Scanning on a File System" on page 71
Enable the vscan service.	Starts the scan service.	"How to Enable the Vscan Service" on page 72
Add a scan engine to the vscan service.	Includes specific scan engines in the vscan service.	"How to Add a Scan Engine" on page 72
Configure the vscan service.	Views and changes vscan properties.	"How to View Vscan Properties" on page 72 "How to Change Vscan Properties" on page 73
Configure the vscan service for specific file types.	Specifies the file types to include and exclude in a scan.	"How to Exclude Files From Virus Scans" on page 73

▼ 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.

Before You Begin You must be assigned the ZFS File System Management or the ZFS Storage Management rights profile. The ZFS file system allows some administrative tasks to be delegated to specific users. For more information about delegated administration, see Chapter 9, "Oracle Solaris ZFS Delegated Administration," in *Oracle Solaris Administration: ZFS File Systems*.

1 **Become an administrator with the required security attributes.**

For more information, see "How to Obtain Administrative Rights" on page 160.

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

Before You Begin You must be assigned the VSCAN Management rights profile.

1 **Become an administrator with the required security attributes.**

For more information, see "How to Obtain Administrative Rights" on page 160.

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

```
# svcadm enable vscan
```

▼ How to Add a Scan Engine

Before You Begin You must be assigned the VSCAN Management rights profile.

1 **Become an administrator with the required security attributes.**

For more information, see "How to Obtain Administrative Rights" on page 160.

2 **To add a scan engine to the vscan service with default properties, type:**

```
#vscanadm add-engine engine_ID
```

See the man page for the `vscanadm`(1M) command for a description of the command.

▼ How to View Vscan Properties

Before You Begin You must be assigned the VSCAN Management rights profile.

1 **Become an administrator with the required security attributes.**

For more information, see "How to Obtain Administrative Rights" on page 160.

2 **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 man page 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.

Before You Begin You must be assigned the VSCAN Management rights profile.

1. **Become an administrator with the required security attributes.**

   For more information, see "How to Obtain Administrative Rights" on page 160.

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 man page 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.

Before You Begin You must be assigned the VSCAN Management rights profile.

1. **Become an administrator with the required security attributes.**

   For more information, see "How to Obtain Administrative Rights" on page 160.

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,-*
  ```

For more information, see the vscanadm(1M) man page.

CHAPTER 5

Controlling Access to Devices (Tasks)

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 75
- "Configuring Device Policy (Tasks)" on page 76
- "Managing Device Allocation (Tasks)" on page 79
- "Allocating Devices (Tasks)" on page 84
- "Device Protection (Reference)" on page 87

For overview information about device protection, see "Controlling Access to Devices" on page 42.

Configuring Devices (Task Map)

The following task map points to the tasks to perform to manage access to devices.

Task	For Instructions
Manage device policy.	"Configuring Device Policy (Task Map)" on page 76
Manage device allocation.	"Managing Device Allocation (Task Map)" on page 79
Use device allocation.	"Allocating Devices (Tasks)" on page 84

Configuring Device Policy (Tasks)

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

Configuring Device Policy (Task Map)

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

Task	Description	For Instructions
View the device policy for the devices on your system.	Lists the devices and their device policy.	"How to View Device Policy" on page 76
Require privilege for device use.	Uses privileges to protect a device.	"How to Change the Device Policy on an Existing Device" on page 77
Remove privilege requirements from a device.	Removes or lessens the privileges that are required to access a device.	Example 5–3
Audit changes in device policy.	Records changes in device policy in the audit trail.	"How to Audit Changes in Device Policy" on page 78
Access /dev/arp.	Gets Oracle Solaris IP MIB-II information.	"How to Retrieve IP MIB-II Information From a /dev/* Device" on page 78

▼ 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/bge
/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/bge
read_priv_set=net_rawaccess
write_priv_set=net_rawaccess
```

▼ How to Change the Device Policy on an Existing Device

Before You Begin You must be assigned the Device Security rights profile.

1 **Become an administrator with the required security attributes.**

For more information, see "How to Obtain Administrative Rights" on page 160.

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.

Before You Begin You must be assigned the Audit Configuration rights profile.

1 **Become an administrator with the required security attributes.**

For more information, see "How to Obtain Administrative Rights" on page 160.

2 **Preselect the audit class that includes AUE_MODDEVPLCY audit event.**

```
# auditconfig -getflags
current-flags
# auditconfig -setflags current-flags,as
```

For detailed instructions, see "How to Preselect Audit Classes" on page 545.

▼ How to Retrieve IP MIB-II Information From a /dev/* Device

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

1 **Determine the device policy on /dev/ip and /dev/arp.**

```
% 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 (Tasks)

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.

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 (Tasks)" on page 84 for instructions on allocating devices.

Task	Description	For Instructions
Make a device allocatable. Disable device allocation.	Enables a device to be allocated to one user at a time. Removes allocation restrictions from all devices.	"How to Enable Device Allocation" on page 80
Authorize users to allocate a device.	Assigns device allocation authorizations to users.	"How to Authorize Users to Allocate a Device" on page 80
View the allocatable devices on your system.	Lists the devices that are allocatable, and the state of the device.	"How to View Allocation Information About a Device" on page 81
Forcibly allocate a device.	Allocates a device to a user who has an immediate need.	"Forcibly Allocating a Device" on page 82
Forcibly deallocate a device.	Deallocates a device that is currently allocated to a user.	"Forcibly Deallocating a Device" on page 82
Change the allocation properties of a device.	Changes the requirements for allocating a device.	"How to Change Which Devices Can Be Allocated" on page 82
Create a device-clean script.	Purges data from a physical device.	"Writing New Device-Clean Scripts" on page 94
Audit device allocation	Records device allocation in the audit trail	"How to Audit Device Allocation" on page 83

▼ How to Enable Device Allocation

Before You Begin You must be assigned the Device Security rights profile.

1. **Become an administrator with the required security attributes.**

   For more information, see "How to Obtain Administrative Rights" on page 160.

2. **Enable the device allocation service and verify that the service is enabled.**

   ```
   # svcadm enable svc:/system/device/allocate
   # svcs -x allocate
   svc:/system/device/allocate:default (device allocation)
    State: online since September 10, 2011 01:10:11 PM PDT
      See: allocate(1)
      See: deallocate(1)
      See: list_devices(1)
      See: device_allocate(1M)
      See: mkdevalloc(1M)
      See: mkdevmaps(1M)
      See: dminfo(1M)
      See: device_maps(4)
      See: /var/svc/log/system-device-allocate:default.log
   Impact: None.
   ```

   To disable the device allocation service, use the `disable` subcommand.

   ```
   # svcadm disable device/allocate
   ```

▼ How to Authorize Users to Allocate a Device

Before You Begin You must be assigned the User Security rights profile.

1. **Become an administrator with the required security attributes.**

   For more information, see "How to Obtain Administrative Rights" on page 160.

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 170. Give the rights profile appropriate properties, such as the following:

   - Rights profile name: `Device Allocation`
   - Granted authorizations: `solaris.device.allocate`
   - Commands with security attributes: In the `exec_attr` database, `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 a Role" on page 165. 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 the first in 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 84.

▼ How to View Allocation Information About a Device

Before You Begin You have completed "How to Enable Device Allocation" on page 80.

You must be assigned the Device Security rights profile.

1 **Become an administrator with the required security attributes.**

For more information, see "How to Obtain Administrative Rights" on page 160.

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.
- `rmdisk`[*n*] – Is a removable media device.
- `sr`[*n*] – Is a CD-ROM 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 You must be assigned 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 You must be assigned 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

Before You Begin Device allocation must be enabled for this procedure to succeed. To enable device allocation, see "How to Enable Device Allocation" on page 80. You must be superuser.

- **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 (@).

```
# 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 5–5 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 (*).

```
# vi /etc/security/device_allocate
audio;audio;reserved;reserved;*;/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
...
```

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 (*).

```
# 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.

Before You Begin You must be assigned the Audit Configuration rights profile.

1 **Become an administrator with the required security attributes.**

For more information, see "How to Obtain Administrative Rights" on page 160.

2 **Preselect the `ot` audit class.**

```
# auditconfig -getflags
current-flags
# auditconfig -setflags current-flags,ot
```

For detailed instructions, see "How to Preselect Audit Classes" on page 545.

Allocating Devices (Tasks)

Device allocation reserves the use of a device to one user at a time. Devices that require a mount point must be mounted. The following procedures show users how to allocate devices.

▼ How to Allocate a Device

Before You Begin Device allocation must be enabled, as described in "How to Enable Device Allocation" on page 80. 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`.

```
% 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 82.

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

Devices mount automatically if you are granted the appropriate privileges. Follow this procedure if the device fails to mount.

Before You Begin You have allocated the device. You are assigned 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 80.

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 that 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 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:

- Verify 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. For the list of available profile shells, see the `pfexec(1)`.
- Verify that you own the specified mount point. You must 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 5–11 Deallocating a Microphone

In this example, the user `jdoe` deallocates the microphone, `audio`.

```
% whoami
jdoe
% deallocate audio0
```

Example 5–12 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 Oracle Solaris 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. Users who are assigned the Device Management and Device Security rights profiles can manage devices.

The following table lists the device management commands.

TABLE 5–1 Device Management Commands

Man Page for Command	Purpose
devfsadm(1M)	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.
getdevpolicy(1M)	Displays the policy associated with one or more devices. This command can be run by any user.
add_drv(1M)	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.
update_drv(1M)	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.
rem_drv(1M)	Removes a device or device driver.

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. 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 svc:/system/device/allocate service. For more information, see the smf(5) man page and the man pages for the device allocation commands.
- The allocate, deallocate, dminfo, and list_devices commands. For more information, see "Device Allocation Commands" on page 90.
- The Device Management and Device Security rights profiles. For more information, see "Device Allocation Rights Profiles" on page 89.

- 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 Oracle Solaris.

Device Allocation Service

The svc:/system/device/allocate service controls device allocation. This service is off by default. To enable the service, run the svcadm enable svc:/system/device/allocate command.

Device Allocation Rights Profiles

The Device Management and Device Security rights profiles are required to manage devices and device allocation.

These rights profiles include the following authorizations:

- solaris.device.allocate – Required to allocate a device
- solaris.device.cdrw – Required to read and write a CD-ROM
- solaris.device.config – Required to configure the attributes of a device
- solaris.device.grant – Required to delegate to another user the device authorizations that are assigned to you
- solaris.device.mount.alloptions.fixed – Required to specify mount options when mounting a fixed device
- solaris.device.mount.alloptions.removable – Required to specify mount options when mounting a removable device
- solaris.device.mount.fixed – Required to mount a fixed device
- solaris.device.mount.removable – Required to mount a removable device
- solaris.device.revoke – Required to revoke or reclaim a device

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 5–2 Device Allocation Commands

Man Page for Command	Purpose
dminfo(1M)	Searches for an allocatable device by device type, by device name, and by full path name.
list_devices(1)	Lists the status of allocatable devices. Lists all the device-special files that are associated with any device that is listed in the device_maps file. With the -U option, lists the devices that are allocatable or allocated to the specified user ID. This option allows you to check which devices are allocatable or allocated to another user. You must have the solaris.device.revoke authorization.
allocate(1)	Reserves an allocatable device for use by one user. By default, a user must have the solaris.device.allocate 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.
deallocate(1)	Removes the allocation reservation from a device.

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 80.

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 "Selected Commands That Require Authorizations" on page 205.

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 a user or 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. The /etc/security/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 5–13 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/fd0 /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 81.

device-type Specifies the generic device type. The generic name is the name for the class of devices, such as st, fd, rmdisk, 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

You can modify the /etc/security/device_allocate file to change devices from allocatable to nonallocatable, 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 81.

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.

reserved — Sun reserves the two fields that are marked reserved for future use.

auths — 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 solaris.device.allocate in the *auths* field indicates that the solaris.device.allocate authorization is required to allocate the device. An at sign (@) in this file indicates that the device is allocatable by any user.

device-exec — 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 *device-exec* script is run any time that the device is acted on by the deallocate command.

For example, the following entry for the sr0 device indicates that the CD-ROM drive is allocatable by a user with the solaris.device.allocate authorization:

```
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, removable media devices, 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 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. Oracle Solaris 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-clean scripts are provided for diskettes and CD-ROM drives:

- **`fd_clean` script** – Is a device-clean script for diskettes.
- **`sr_clean` script** – Is a device-clean script 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-clean scripts. The `deallocate` command passes a parameter to the device-clean scripts. 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-clean scripts 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.

CHAPTER 6

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:

- "Basic Audit Reporting Tool (Overview)" on page 95
- "Using BART (Tasks)" on page 98
- "BART Manifests, Rules Files, and Reports (Reference)" on page 108

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 Oracle 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 of little value.

For more information about BART manifests, see “BART Manifest File Format” on page 108.

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 111.

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 lists 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 user errors are also 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 other purposes.

For more information about the BART rules file, see “BART Rules File Format” on page 109 and the `bart_rules(4)` man page.

Using BART (Tasks)

You can run the `bart` command as a regular user, superuser, or a user who has assumed a role. If you run the `bart` command as a regular user, you are only able to catalog and monitor files that you have permission to access, such as files in your home directory. The advantage of becoming superuser when you run the `bart` command is that the manifests you create 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. For more information about using role-based access control, see “Role-Based Access Control (Overview)” on page 133.

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.

Using BART (Task Map)

Task	Description	For Instructions
Create a BART manifest.	Generates a list of information about every file that is installed on a system.	"How to Create a Manifest" on page 99
Create a custom BART manifest.	Generates a list of information about specific files that are installed on a system.	"How to Customize a Manifest" on page 101
Compare BART manifests.	Generates a report that compares changes to a system over time. Or, generates a report that compares one or several systems to control system.	"How to Compare Manifests for the Same System Over Time" on page 102 "How to Compare Manifests From Different Systems" on page 104
(Optional) Customize a BART report.	Generates a custom BART report in one of the following ways: ■ By specifying attributes. ■ By using a rules file.	"How to Customize a BART Report by Specifying File Attributes" on page 106 "How to Customize a BART Report by Using a Rules File" on page 107

▼ How to Create a Manifest

You can create a manifest of a system immediately after an initial Oracle Solaris software installation. This type of manifest provides 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.

Before You Begin To create a system manifest, you must be in the `root` role.

1. **After installing the Oracle 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 are interpreted relative to this directory. All paths reported in the manifest are 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, reads 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.

2 **Examine the contents of the manifest.**

3 **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 is 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 1.1
! HASH SHA256
! Wednesday, September 07, 2011 (22:22:27)
# 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 1024 40755 user::rwx,group::r-x,mask:r-x,other:r-x
3ebc418eb5be3729ffe7e54053be2d33ee884205502c81ae9689cd8cca5b0090 0 0
.
.
.
/zone D 512 40755 user::rwx group::r-x,mask:r-x,other:r-x 3f81e892
154de3e7bdfd0d57a074c9fae0896a9e2e04bebfe5e872d273b063319e57f334 0 0
.
.
.
```

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 file name. 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 hash line indicates the hash mechanism that was used. 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.

Before You Begin You must be in the `root` role.

1 Determine which files you want to catalog and monitor.

2 After installing the Oracle 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
```

3 **Examine the contents of the manifest.**

4 **Save the manifest for future use.**

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.

Before You Begin To create and compare manifests that include public objects, you must be in the `root` role.

1 **After installing the Oracle Solaris software, create a control manifest of the files that you want to monitor on the system.**

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

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

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

3 **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 read from standard input.

`-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.

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

Example 6–2 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.090711
! Version 1.1
! HASH SHA256
! Wednesday, September 07, 2011 (11:11:17)
# 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
/.cpr_config F 2236 100644 owner@:read_data/write_data/append_data/read_xattr/wr
ite_xattr/read_attributes/write_attributes/read_acl/write_acl/write_owner/synchr
onize:allow,group@:read_data/read_xattr/read_attributes/read_acl/synchronize:all
ow,everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
4e271c59 0 0 3ebc418eb5be3729ffe7e54053be2d33ee884205502c81ae9689cd8cca5b0090
/.login F 1429 100644 owner@:read_data/write_data/append_data/read_xattr/write_x
attr/read_attributes/write_attributes/read_acl/write_acl/write_owner/synchronize
:allow,group@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow,ev
eryone@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
4bf9d6d7 0 3 ff6251a473a53de68ce8b4036d0f569838cff107caf1dd9fd04701c48f09242e
.
.
.
```

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

```
# bart create -R /etc > system1.test.101011
Version 1.1
! HASH SHA256
! Monday, October 10, 2011 (10:10:17)
# 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
/.cpr_config F 2236 100644 owner@:read_data/write_data/append_data/read_xattr/wr
ite_xattr/read_attributes/write_attributes/read_acl/write_acl/write_owner/synchr
onize:allow,group@:read_data/read_xattr/read_attributes/read_acl/synchronize:all
ow,everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
4e271c59 0 0 3ebc418eb5be3729ffe7e54053be2d33ee884205502c81ae9689cd8cca5b0090
.
.
.
```

- Compare the control manifest with the test manifest.

```
# bart compare system1.control.090711 system1.test.101011
/security/audit_class
mtime  4f272f59
```

The preceding output indicates that the modification time on the `audit_class` file has 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 Different Systems

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 Oracle 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 lists any discrepancies in the file contents for each test system that you compare with the control system.

Before You Begin To compare system manifests, you must be in the `root` role.

1 **After installing the Oracle Solaris software, create a control manifest.**

```
# bart create options > control-manifest
```

2 **Save the control manifest.**

3 **On the test system, use the same `bart` options to create a manifest, and redirect the output to a file.**

```
# bart create options > test1-manifest
```

Choose a distinct and meaningful name for the test manifest.

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

5 **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:

```
# 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.

6 **Compare the control manifest with the test manifest and redirect the output to a file.**

```
# bart compare control-manifest test1-manifest > test1.report
```

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

8 **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.

Example 6–3 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.

```
# bart create -R /usr/bin > control-manifest.090711
! Version 1.1
! HASH SHA256
! Wednesday, September 07, 2011 (11:11:17)
# 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
/2to3 F 105 100555 owner@:read_data/read_xattr/write_xattr/execute/read_attribut
es/write_attributes/read_acl/write_acl/write_owner/synchronize:allow,group@:read
_data/read_xattr/execute/read_attributes/read_acl/synchronize:allow,everyone@:re
ad_data/read_xattr/execute/read_attributes/read_acl/synchronize:allow 4bf9d261 0
 2 154de3e7bdfd0d57a074c9fae0896a9e2e04bebfe5e872d273b063319e57f334
/7z F 509220 100555 owner@:read_data/read_xattr/write_xattr/execute/read_attribu
tes/write_attributes/read_acl/write_acl/write_owner/synchronize:allow,group@:rea
d_data/read_xattr/execute/read_attributes/read_acl/synchronize:allow,everyone@:r
ead_data/read_xattr/execute/read_attributes/read_acl/synchronize:allow 4dadc48a 0
 2 3ecd418eb5be3729ffe7e54053be2d33ee884205502c81ae9689cd8cca5b0090
...
```

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

```
# bart create -R /usr/bin > system2-manifest.101011
! Version 1.1
! HASH SHA256
! Monday, October 10, 2011 (10:10:22)
# 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
```

```
/2to3 F 105 100555 owner@:read_data/read_xattr/write_xattr/execute/read_attribut
es/write_attributes/read_acl/write_acl/write_owner/synchronize:allow,group@:read
_data/read_xattr/execute/read_attributes/read_acl/synchronize:allow,everyone@:re
ad_data/read_xattr/execute/read_attributes/read_acl/synchronize:allow 4bf9d261 0
 2 154de3e7bdfd0d57a074c9fae0896a9e2e04bebfe5e872d273b063319e57f334
...
```

- 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
  /su:
    gid  control:3  test:1
  /ypcat:
    mtime  control:3fd72511  test:3fd9eb23
  ```

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.

Before You Begin You must be in the root role.

1 **Determine which file attributes you want to monitor.**

2 **After installing the Oracle Solaris software, create a control manifest.**

3 **Create a test manifest when you want to monitor changes.**

Prepare the test manifest identically to the control manifest.

4 **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.

5 **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.

Before You Begin You must be in the `root` role.

1 **Determine which files and file attributes you want to monitor.**

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

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

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

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

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

5 **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
```

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

Example 6–4 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.

```
# 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 Manifests, Rules Files, and Reports (Reference)

This section describes the format of files that BART uses and creates.

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 109.

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 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 6–5 Rules File Format

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

<subtree2> [pattern2..]
```

EXAMPLE 6–5 Rules File Format *(Continued)*

```
<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`
- `mode`
- `mtime`
- `size`
- `type`
- `uid`

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, checks 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:

0 Success

1 Nonfatal error when processing files, such as permission problems

>1 Fatal error, such as an invalid command-line option

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:daca28ae0de97afd7a6b91fde8d57afa
  test:84b2b32c4165887355317207b48a6ec7
  ```

- 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 110.

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 Oracle Solaris. The chapter also describes how to protect the system from files whose permissions could compromise the system.

Note – To protect ZFS files with access control lists (ACLs), see Chapter 8, "Using ACLs and Attributes to Protect Oracle Solaris ZFS Files," in *Oracle Solaris Administration: ZFS File Systems*.

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

- "Using UNIX Permissions to Protect Files" on page 113
- "Protecting Executable Files From Compromising Security" on page 121
- "Protecting Files With UNIX Permissions (Task Map)" on page 122
- "Protecting Against Programs With Security Risk (Task Map)" on page 127

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 7–1 Commands for Securing Files and Directories

Command	Description	Man Page
`ls`	Lists the files in a directory and information about the files.	`ls(1)`

TABLE 7–1 Commands for Securing Files and Directories *(Continued)*

Command	Description	Man Page
`chown`	Changes the ownership of a file.	`chown(1)`
`chgrp`	Changes the group ownership of a file.	`chgrp(1)`
`chmod`	Changes permissions on a file. You can use either symbolic mode, which uses letters and symbols, or absolute mode, which uses octal numbers, to change permissions on a file.	`chmod(1)`

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, the `root` account 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

Symbol	Permission	Object	Description
`r`	Read	File	Designated users can open and read the contents of a file.
		Directory	Designated users can list files in the directory.
`w`	Write	File	Designated users can modify the contents of the file or delete the file.
		Directory	Designated users can add files or add links in the directory. They can also remove files or remove links in the directory.
`x`	Execute	File	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.
		Directory	Designated users can open files or execute files in the directory. They also can make the directory and the directories beneath it current.
`-`	Denied	File and Directory	Designated users cannot read, write, or execute the file.

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 128.

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-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 128.

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 is an example of a privileged user. The sticky bit prevents a user from deleting other users' files from public directories such as /tmp:

```
drwxrwxrwt 7  root  sys   400 Sep  3 13:37 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 shell initialization files, such as .bashrc and .kshrc.user. An administrator can also set defaults in the /etc/profile 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 values and their effect on an executable file.

TABLE 7–3 umask Settings for Different Security Levels

Level of Security	umask Setting	Permissions Disallowed
Permissive (744)	022	w for group and others
Moderate (740)	027	w for group, rwx for others
Moderate (741)	026	w for group, rw for others
Severe (700)	077	rwx for group and others

For more information about 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

Octal Value	File Permissions Set	Permissions Description
0	---	No permissions
1	--x	Execute permission only
2	-w-	Write permission only
3	-wx	Write and execute permissions
4	r--	Read permission only
5	r-x	Read and execute permissions
6	rw-	Read and write permissions
7	rwx	Read, write, and execute permissions

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

Symbol	Function	Description
u	*who*	User (owner)
g	*who*	Group
o	*who*	Others
a	*who*	All
=	*operator*	Assign

TABLE 7–5 Setting File Permissions in Symbolic Mode *(Continued)*

Symbol	Function	Description
`+`	*operator*	Add
`-`	*operator*	Remove
`r`	*permissions*	Read
`w`	*permissions*	Write
`x`	*permissions*	Execute
`l`	*permissions*	Mandatory locking, `setgid` bit is on, group execution bit is off
`s`	*permissions*	`setuid` or `setgid` bit is on
`t`	*permissions*	Sticky bit is on, execution bit for others is on

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

Octal Value	Special File Permissions
`1`	Sticky bit
`2`	`setgid`
`4`	`setuid`

Using Access Control Lists to Protect UFS Files

Traditional UNIX file protection provides read, write, and execute permissions for the three user classes: file owner, file group, and other. In a UFS file system, 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

Note – For ACLs in the ZFS file system and ACLs on NFSv4 files, see Chapter 8, "Using ACLs and Attributes to Protect Oracle Solaris ZFS Files," in *Oracle Solaris Administration: ZFS File Systems*.

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.

On a UFS file system, ACL entries are set on a file through the `setfacl` command. UFS 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).

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.

For more information about ACLs on UFS file systems, see *System Administration Guide: Security Services* for the Oracle Solaris 10 release.

Protecting Executable Files From Compromising Security

Programs read and write data on the stack. Typically, they execute from read-only portions of memory that are specifically designated for code. Some attacks that cause buffers on the stack to overflow try to insert new code on the stack and cause the program to execute it. Removing execute permission from the stack memory prevents these attacks from succeeding. That is, most programs can function correctly without using executable stacks.

64-bit processes always have non-executable stacks. The noexec_user_stack variable enables you to specify whether the stacks of 32-bit processes are executable To comply with the 32-bit SPARC ABI, the default value is zero, which specifies that the stack is 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 the noexec_user_stack variable. If you do not want any messages logged, then set the log variable, noexec_user_stack_log, 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. You can also compile your program with -M /usr/lib/ld/map.noexstk to make the stack non-executable regardless of the system-wide setting.

Protecting Files (Tasks)

The following procedures protect files with UNIX permissions, locate files with security risks, and protect the system from compromise by these files.

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.

Task	For Instructions
Display file information.	"How to Display File Information" on page 122
Change local file ownership.	"How to Change the Owner of a File" on page 123 "How to Change Group Ownership of a File" on page 124
Change local file permissions.	"How to Change File Permissions in Symbolic Mode" on page 124 "How to Change File Permissions in Absolute Mode" on page 125 "How to Change Special File Permissions in Absolute Mode" on page 126

▼ 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.**

```
% 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.

```
% cd /sbin
% ls -la
total 4960
drwxr-xr-x   2 root     sys          64 Dec  8 11:57 ./
drwxr-xr-x  39 root     root         41 Dec  8 15:20 ../
-r-xr-xr-x   1 root     bin       21492 Dec  1 20:55 autopush*
-r-xr-xr-x   1 root     bin       33680 Oct  1 11:36 beadm*
-r-xr-xr-x   1 root     bin      184360 Dec  1 20:55 bootadm*
lrwxrwxrwx   1 root     root         21 Jun  7  2010 bpgetfile -> ...
-r-xr-xr-x   1 root     bin       86048 Dec  1 20:55 cryptoadm*
-r-xr-xr-x   1 root     bin       12828 Dec  1 20:55 devprop*
-r-xr-xr-x   1 root     bin      130132 Dec  1 20:55 dhcpagent*
-r-xr-xr-x   1 root     bin       13076 Dec  1 20:55 dhcpinfo*
```

```
.
.
.
```

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 114.
- Permissions – For example, r-xr-xr-x. For description, see "File and Directory Ownership" on page 114.
- 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, 21308.
- Date the file was created or the last date that the file was changed – For example, Dec 9 15:55.
- Name of the file – For example, dhcpinfo.

▼ How to Change the Owner of a File

Before You Begin If you are not the owner of the file or directory, you must be assigned the Object Access Management rights profile. To change a file that is a public object, you must be superuser.

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 Become an administrator with the required security attributes.

For more information, see "How to Obtain Administrative Rights" on page 160.

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
```

NFS-mounted file systems have further restrictions on changing ownership and groups. For more information, see Chapter 6, "Accessing Network File Systems (Reference)," in *Oracle Solaris Administration: Network Services*.

Example 7–2 Enabling Users to Change the Ownership of Their Own Files

Security Consideration – You need a good reason to change the setting of the `rstchown` variable to zero. The default setting prevents users from listing their files as belonging to others so as to bypass space quotas.

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.

▼ How to Change Group Ownership of a File

Before You Begin If you are not the owner of the file or directory, you must be assigned the Object Access Management rights profile. To change a file that is a public object, you must be superuser.

1 **Become an administrator with the required security attributes.**

For more information, see "How to Obtain Administrative Rights" on page 160.

2 **Change the group ownership of a file.**

```
$ chgrp scifi example-file
```

For information about setting up groups, see Chapter 2, "Managing User Accounts and Groups (Overview)," in *Oracle Solaris Administration: Common Tasks*.

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

In the following procedure, a user changes permissions on a file that the user owns.

1 **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.

2 **Verify that the permissions of the file have changed.**

```
% ls -l filename
```

Note – If you are not the owner of the file or directory, you must be assigned the Object Access Management rights profile. To change a file that is a public object, you must be superuser.

Example 7–3 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 to a local file for user, group, and others.

```
$ chmod a+rx example-file2
```

In the following example, read, write, and execute permissions for group are assigned to a local file.

```
$ chmod g=rwx example-file3
```

▼ How to Change File Permissions in Absolute Mode

In the following procedure, a user changes permissions on a file that the user owns.

1 **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.

---

2 **Verify that the permissions of the file have changed.**

```
% ls -l filename
```

Note – If you are not the owner of the file or directory, you must be assigned the Object Access Management rights profile. To change a file that is a public object, you must be superuser.

Example 7–4 Changing Permissions in Absolute Mode

In the following example, the permissions of a directory that is open to the public 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 jdoe   staff    6023 Aug  5 12:06 public_dir
# chmod 755 public_dir
# ls -ld public_dir
drwxr-xr-x  1 jdoe   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 jdoe   staff    6023 Aug  5 12:06 my_script
% chmod 700 my_script
% ls -l my_script
-rwx------ 1 jdoe   staff    6023 Aug  5 12:06 my_script
```

▼ How to Change Special File Permissions in Absolute Mode

Before You Begin If you are not the owner of the file or directory, you must be assigned the Object Access Management rights profile. To change a file that is a public object, you must be superuser.

1 **Become an administrator with the required security attributes.**

For more information, see "How to Obtain Administrative Rights" on page 160.

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 jdoe   staff          512 May 15 15:27 public_dir
```

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.

Task	Description	For Instructions
Find files with special permissions.	Locates files with the setuid bit set, but that are not owned by the root user.	"How to Find Files With Special File Permissions" on page 128
Prevent executable stack from overflowing.	Prevents programs from exploiting an executable stack.	"How to Disable Programs From Using Executable Stacks" on page 129
Prevent logging of executable stack messages.	Turns off logging of executable stack messages.	Example 7–7

▼ How to Find Files With Special File Permissions

This procedure locates potentially unauthorized use of the setuid and setgid permissions on programs. A suspicious executable file grants ownership to a user rather than to root or bin.

Before You Begin You must be in the root role.

1 Find files with setuid permissions by using the find command.

```
# 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.

2 Display the results in /tmp/*filename*.

```
# more /tmp/filename
```

For background information about setuid permissions, see "setuid Permission" on page 116.

Example 7–6 Finding Files With setuid Permissions

The output from the following example shows that a user in a group called 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 /var/tmp/chkprm directory to an archive.

```
# 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-sr-x 1 root bin 33208 Aug 10 15:55 /usr/lib/lpadmin
-r-sr-sr-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
-rwsr-xr-x 1 root sys 21780 Aug 11 01:27 /usr/bin/newgrp
-r-sr-sr-x 1 root sys 23000 Aug 11 01:27 /usr/bin/passwd
-r-sr-xr-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 32–bit executable stacks, see "Protecting Executable Files From Compromising Security" on page 121.

Before You Begin You must be in the root role.

1. **Edit the /etc/system file, and add the following line:**

```
set noexec_user_stack=1
```

2. **Reboot the system.**

```
# reboot
```

Example 7–7 Disabling the Logging of Executable Stack Messages

In this example, the logging of executable stack messages is disabled, and then the system is rebooted.

```
# cat /etc/system
set noexec_user_stack=1
set noexec_user_stack_log=0
# reboot
```

See Also For more information, read the following:

- http://blogs.oracle.com/gbrunett/entry/solaris_non_executable_stack_overview
- http://blogs.oracle.com/gbrunett/entry/solaris_non_executable_stack_continued

- `http://blogs.oracle.com/gbrunett/entry/solaris_non_executable_stack_concluded`

PART III

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.

- Chapter 8, "Using Roles and Privileges (Overview)"
- Chapter 9, "Using Role-Based Access Control (Tasks)"
- Chapter 10, "Security Attributes in Oracle Solaris (Reference)"

◆ ◆ ◆ CHAPTER 8

Using Roles and Privileges (Overview)

The role-based access control (RBAC) feature of Oracle Solaris and the privileges feature of Oracle Solaris 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 133
- "Privileges (Overview)" on page 145

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 the `root` role. 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 145.
- For information about RBAC tasks, see Chapter 9, "Using Role-Based Access Control (Tasks)."
- For reference information, see Chapter 10, "Security Attributes in Oracle Solaris (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.

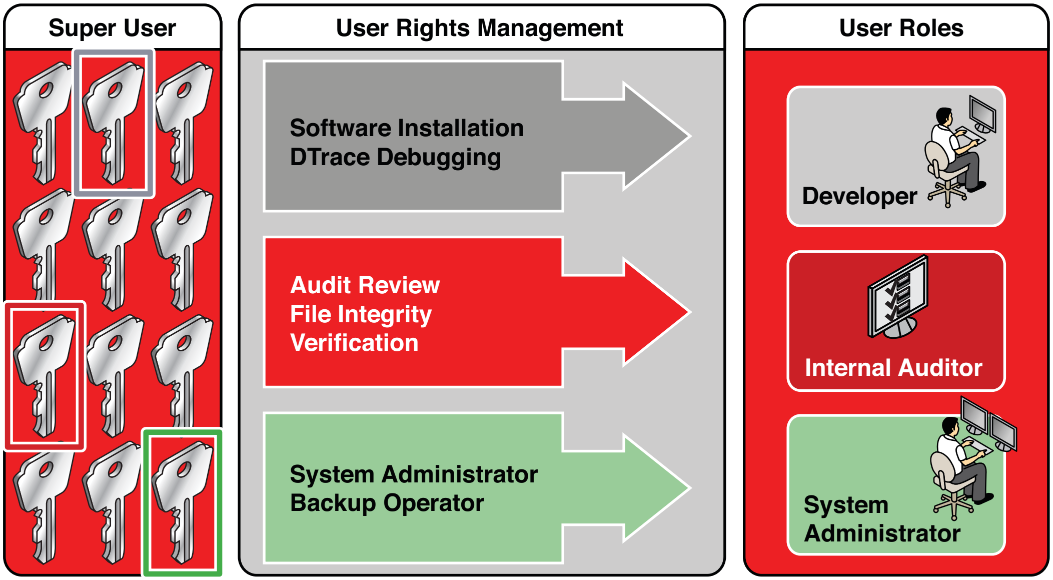
Role-based access control (RBAC) provides a more secure alternative to the all-or-nothing superuser model. With RBAC, you can enforce security policy at a more fine-grained level. RBAC uses the security principle of *least privilege*. Least privilege means that a user has precisely the amount of privilege that is necessary to perform a job. Regular users have enough privilege to use their applications, check the status of their jobs, print files, create new files, and so on. Capabilities beyond regular 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 Oracle Solaris software. You create the roles and assign the profiles.

Rights profiles can provide broad capabilities. For example, the System Administrator rights profile enables an account to perform tasks that are not related to security, such as printer management and cron jobs. Rights profiles can also be narrowly defined. For example, the Cron Management rights profile manages `at` and `cron` jobs. When you create roles, the roles can be assigned broad capabilities or narrow capabilities or both.

The following figure illustrates how RBAC can distribute rights to trusted parties.

FIGURE 8–1 RBAC Distribution of Rights



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 few roles are shipped with Oracle Solaris, a variety of roles can easily be configured. You can base most roles on rights profiles of the same name:

- **root** – A powerful role that is equivalent to the `root` user. However, this `root` cannot log in. A regular user must log in, then assume the assigned `root` role. This role is configured by default.
- **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.

Note – The Media Backup rights profile provides access to the entire root file system. Therefore, while the Media Backup and Operator rights profiles are designed for a junior administrator, you must ensure that the user can be trusted.

You might also want to configure one or more security roles. Three rights profiles and their supplementary profiles handle security: Information Security, User Security, and Zone Security. Network security is a supplementary profile in the Information Security rights profile.

These roles do not have to be implemented. Roles are a function of an organization's security needs. One strategy is to set up roles 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 regular 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 Contrasted With the RBAC With Privileges Model

User Capabilities on a System	Superuser Model	RBAC Model
Can become superuser with full superuser capability	Can	Can
Can log in as a user with full user capabilities	Can	Can
Can become superuser with limited capabilities	Cannot	Can
Can log in as a user, and have superuser capabilities, sporadically	Can, with `setuid` programs only	Can, with `setuid` programs and with RBAC
Can log in as a user with administrative capabilities, but without full superuser capability	Cannot	Can, with RBAC and with directly-assigned privileges and authorizations
Can log in as a user with fewer capabilities than a regular user	Cannot	Can, with RBAC and with removed privileges
Can track superuser actions	Can, by auditing the `su` command	Can, by auditing calls to `pfexec()` Also, the name of the user who has assumed the `root` role is in the audit trail

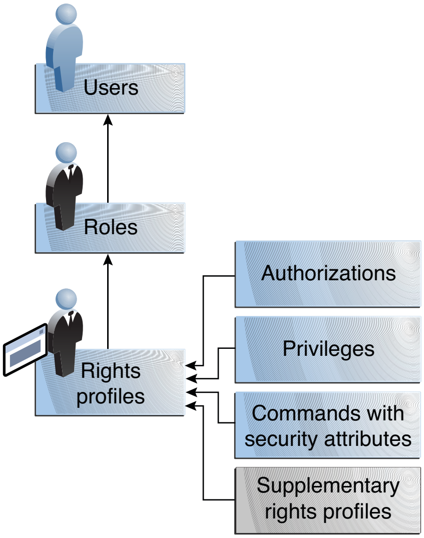
RBAC Elements and Basic Concepts

The RBAC model in Oracle Solaris introduces the following elements:

- **Authorization –** A permission that enables a user or role to perform a class of actions that require additional rights. For example, security policy at installation gives regular 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/auth_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()`. Regular users have basic privileges. To see your basic privileges, run the `ppriv -vl 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 regular users. For example, `setuid` and `setgid` programs have security attributes. In the RBAC model, authorizations and privileges are security attributes in addition to `setuid` and `setgid` programs. These attributes can be assigned to a user. For example, a user with the `solaris.device.allocate` authorization can allocate a device for exclusive use. Privileges can be placed on a process. For example, a process with the `file_flag_set` privilege can set immutable, no-unlink, or append-only file attributes.
- **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 141.
- **Rights profile –** A collection of security attributes that can be assigned to a role or to a user. A rights profiles can include authorizations, directly assigned privileges, commands with security attributes, and other rights profiles. Profiles that are within another profile are called supplementary 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, including the `root` role, 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.

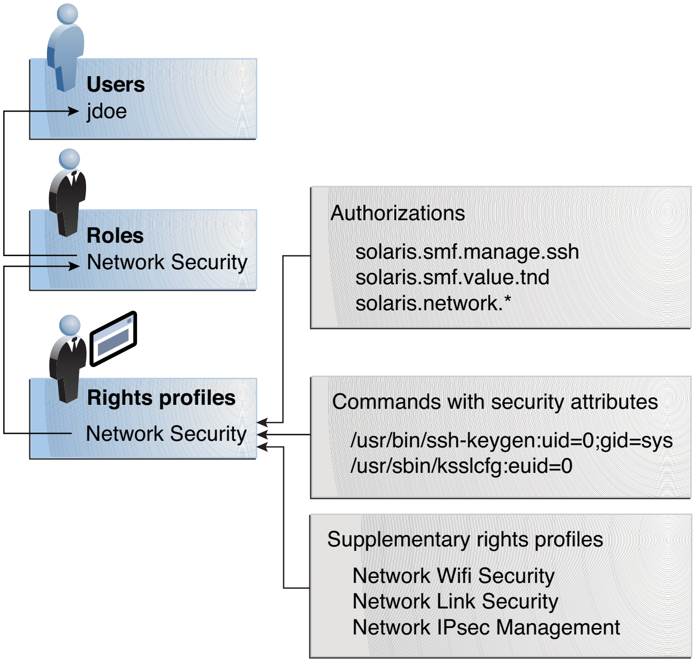
The following figure shows how the RBAC elements work together.

FIGURE 8–2 RBAC Element Relationships


Users
Roles
Rights profiles
Authorizations
Privileges
Commands with security attributes
Supplementary rights profiles

The following figure uses the Network Security role and the Network Security rights profile to demonstrate RBAC relationships.

FIGURE 8–3 Example of RBAC Element Relationships



The Network Security role is used to manage IPsec, wifi, and network links. 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 administrator can customize the role to accept the user password rather than the role password.

In Figure 8–3, the Network Security rights profile is assigned to the Network Security role. The Network Security rights profile contains supplementary profiles that are evaluated in order, Network Wifi Security, Network Link Security, and Network IPsec Management. These supplementary profiles fill out the role's primary tasks.

The Network Security rights profile has three directly assigned authorizations, no directly assigned privileges, and two commands with security attributes. The supplementary rights profiles have directly assigned authorizations, and two of them have commands with security attributes. In the Network Security role, `jdoe` has all assigned authorizations in these profiles, and can run all the commands with security attributes in these profiles. `jdoe` can administer network security

Privilege Escalation

Oracle Solaris provides administrators with a great deal of flexibility when configuring security. As installed, the software does not allow for privilege escalation. Privilege escalation occurs when a user or process gains more administrative rights than were intended to be granted. In this sense, privilege means any security attribute, not just privileges.

Oracle Solaris software includes security attributes that are assigned to the `root` role only. With other security protections in place, an administrator might assign attributes that are designed for the `root` role to other accounts, but such assignment must be made with care.

The following rights profile and set of authorizations can escalate the privileges of a non-root account.

- **Media Restore rights profile –** This profile exists, but is not part of any other rights profile. Because `Media Restore` provides access to the entire root file system, its use is a possible escalation of privilege. Deliberately altered files or substitute media could be restored. By default, the `root` role includes this rights profile.
- **solaris.*.assign authorizations –** These authorizations exist, but are not assigned to any rights profile or account. An account with a `solaris.*.assign` authorization can assign security attributes to others that the account itself is not assigned. For example, a role with the `solaris.profile.assign` authorization can assign rights profiles to other accounts that the role itself is not assigned. By default, only the `root` role has `solaris.*.assign` authorizations.

  Best practice is to assign `solaris.*.delegate` authorizations, not `solaris.*.assign` authorizations. A `solaris.*.delegate` authorization enables the delegater to assign other accounts only those security attributes that the delegater possesses. For example, a role that is assigned the `solaris.profile.delegate` authorization can assign rights profiles that the role itself is assigned to other users and roles.

For escalations that affect the privilege security attribute, see "Prevention of Privilege Escalation" on page 208.

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.

While authorizations can be assigned directly to a role or to a user, best practice is to include authorizations in a rights profile. The rights profile is then added to a role, and the role is assigned to a user. For an example, see Figure 8–3.

Authorizations that include the words `delegate` or `assign` enable the user or role to assign security attributes to others.

To prevent escalation of privilege, do not assign an account an assign authorization.

- A delegate authorization enables the delegater to assign others only those security attributes that the delegater possesses. For example, a role that is assigned the solaris.profile.delegate authorization can assign to others rights profiles that the role itself is assigned.
- An assign authorization enables the assigner to assign others security attributes that the account does not possess. For example, a role with the solaris.profile.assign authorization can assign to others any rights profile.

The solaris.*.assign authorizations are delivered, but are not included in any profile. By default, only the root role has the solaris.*.assign authorizations.

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 about authorizations, see the following sections:

- "Authorizations" on page 200
- "auth_attr Database" on page 202
- "Selected Commands That Require Authorizations" on page 205

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 145.

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 assigned 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 `reboot` command requires a real rather than an effective UID. If an effective ID is not sufficient to run a command, you need to assign the real ID to the command.

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 that require security attributes. Then, you can isolate the command with assigned 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 `ipadm`, `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 170 and the `profiles(1)` man page. To determine which commands check for privileges in a particular profile, see "How to View All Defined Security Attributes" on page 156.

Applications That Check Authorizations

Oracle Solaris 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:

- 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 9–16. To write a program that requires authorizations, see "About Authorizations" in *Developer's Guide to Oracle Solaris 11 Security*.

RBAC Rights Profiles

A *rights profile* is a collection of security attributes that can be assigned to a role or user to perform tasks that require administrative rights. A rights profile can include authorizations, privileges, commands with assigned security attributes, and other rights profiles. Privileges that are assigned in a rights profile are in effect for all commands. Rights profiles also contain entries to reduce or extend the initial inheritable set, and to reduce the limit set of privileges.

For more information about rights profiles, see the following sections:

- "Rights Profiles" on page 197
- "`prof_attr` Database" on page 203
- "`exec_attr` Database" on page 203

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. The actions of roles can be audited. For detailed information about setting up roles, see the following sections:

- "How to Plan Your RBAC Implementation" on page 163
- "How to Create a Role" on page 165
- "How to Change the Security Attributes of a Role" on page 178

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 a password. By default, users authenticate to a role by supplying the *role's* password. The administrator can configure the system to enable a user to authenticate by supplying the *user's* password. For the procedure, see "How to Enable a User to Use Own Password to Assume a Role" on page 183.

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.

The fact that `root` is a role in Oracle Solaris prevents anonymous `root` login. 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 169.

The rights profiles that ship with the software are designed to map to roles. For example, the System Administrator rights profile can be used to create the System Administrator role. To configure a role, see "How to Create a Role" on page 165.

Profile Shells and RBAC

Users and roles can run privileged applications from a profile shell. A *profile shell* is a special shell that recognizes the security attributes that are included in a rights profile. Administrators can assign a profile shell to a specific user as a login shell, or the profile shell is started when that user runs the su command to assume a role. In Oracle Solaris every shell has a profile shell counterpart. For example, the profile shell counterparts to the Bourne shell (sh), bash shell (csh), and Korn shell (ksh) are the pfsh, pfbash, and pfksh shells, respectively. For the list of profile shells, see the pfexec(1) man page.

Users who have been directly assigned a rights profile and whose login shell is not a profile shell 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 144.

All commands that are executed in a profile shell can be audited. For more information, see "How to Audit Roles" on page 169.

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 naming service such as LDAP. The name service scope for a system is specified in the name switch service, svc:/system/name-service/switch. 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, privileges, 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 users and roles.

However, direct assignment of privileges 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. A more secure practice is to assign the privilege as a security attribute of a command in a rights profile. Then, that privilege is available only for that command by someone who has that rights profile.

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 assign audit flags.

Usability Considerations When Directly Assigning Security Attributes

Direct assignment of rights profiles and security attributes can affect usability:

- Directly assigned privileges and authorizations, and the commands and authorizations in a directly assigned rights profile, must be interpreted by a profile shell to be effective. By default, users are not assigned a profile shell.

  The user must remember to open a profile shell and execute the commands in that shell.
- Singly assigning authorizations is not scalable. And, directly assigned authorizations might not be sufficient to perform a task. The task might require privileged commands.

  Rights profiles are designed to bundle authorizations and privileged commands together. They are also scalable.

Privileges (Overview)

Process rights management enables processes to be restricted at the command, user, role, or system level. Oracle Solaris 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 about RBAC, see "Role-Based Access Control (Overview)" on page 133.
- For information about how to administer privileges, see "Using Privileges (Tasks)" on page 186.
- For reference information about privileges, see "Privileges" on page 206.

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 *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 manipulates files might require the `file_dac_write` and `file_flag_set` privileges. This capability eliminates the need to run the 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 regular 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 can access 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.

Other logical groups include CONTRACT, CPC, DTRACE, GRAPHICS, VIRT, WIN, and XVM.

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

Feature	No Privileges	Privileges
Daemons	Daemons run as root.	Daemons run as the user daemon. For example, the following daemons have been assigned appropriate privileges and run as daemon: lockd, nfsd, and rpcbind.

TABLE 8–2 Visible Differences Between a System With Privileges and a System Without Privileges *(Continued)*

Feature	No Privileges	Privileges
Log File Ownership	Log files are owned by root.	Log files are now owned by daemon, who created the log file. The root user does not own the file.
Error Messages	Error messages refer to superuser. For example, chroot: not superuser.	Error messages reflect the use of privileges. For example, the equivalent error message for chroot failure is chroot: exec failed.
setuid Programs	Programs use setuid to complete tasks that regular users are not allowed to perform.	Many setuid programs have been changed to run with privileges. For example, the following commands use privileges: audit, ikeadm, ipadm, ipsecconf, ping, traceroute, and newtask.
File Permissions	Device permissions are controlled by DAC. For example, members of the group sys can open /dev/ip.	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.
Audit Events	Auditing the use of the su command covers many administrative functions.	Auditing the use of privileges covers most administrative functions. The pm, ps, ex, ua and as audit classes include audit events that monitor device policy and use of privilege.
Processes	Processes are protected by who owns the process.	Processes are protected by privileges. Process privileges and process flags are visible as a new entry in the /proc/<pid> directory, priv.
Debugging	No reference to privileges in core dumps.	The ELF note section of core dumps includes information about process privileges and flags in the NT_PRPRIV and NT_PRPRIVINFO notes. The ppriv command and other commands show the proper number of properly sized sets. The commands correctly map the bits in the bit sets to privilege names.

Privileges and System Resources

In the Oracle Solaris release, 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 *Oracle Solaris Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management* and Chapter 16, “Non-Global Zone Configuration (Overview),” in *Oracle Solaris Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management*.

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 about privilege-aware programs, see Chapter 2, “Developing Privileged Applications,” in *Developer’s Guide to Oracle Solaris 11 Security*.
- **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. While you cannot modify the basic set, you can modify which privileges a user inherits from 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 is equivalent to the default limit set for the zone, global or non-global. 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 144.

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 *Developer's Guide to Oracle Solaris 11 Security*. To assign privileges to a program that requires privileges, see Example 9–14.

Assigning Privileges

You, in your capacity as security administrator, are responsible for assigning privileges. Best practice is to assign the privilege to a command in a rights profile. The rights profile is then assigned to a role or to a user.

Privileges can also be assigned directly to a user, a role, or a rights profile. 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. 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.

You can expand the privileges that are available in two ways.

- The initial inheritable set can be expanded for users, roles, and systems.
- A privilege that is not in the inheritable set can also be explicitly 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 assigning the user a role. The role would be assigned rights profiles that include commands with added privileges. When the user assumes the role, the user gets the role's profile shell. When commands from the rights profile are typed in the role's shell, the commands execute with the added privileges.

You can also assign a rights profile to the user rather than to a role that the user assumes. When the user opens a profile shell, such as `pfksh`, the user can execute the commands in the user's rights 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 responsiblities.

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 regular users.

Note – The limit set cannot be expanded, because the limit set is initially all privileges.

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 rights profile executes the script in a profile shell. If the script contains commands that require privileges, the commands with added privileges must also be in an assigned rights 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, in the superuser model, are protected by file permissions alone. 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, in Oracle Solaris, 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 are used to modify device policy entries and driver-specific privileges. You must be running a process that has 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

Oracle Solaris 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 193. You can also use the dtrace command. For more information, see the dtrace(1M) man page.

CHAPTER 9

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 (Tasks)" on page 155
- "Using Privileges (Tasks)" on page 186

For an overview of RBAC, see "Role-Based Access Control (Overview)" on page 133. For reference information, see Chapter 10, "Security Attributes in Oracle Solaris (Reference)." To use privileges, see "Using Privileges (Tasks)" on page 186.

Using RBAC (Tasks)

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, including the use of privilege.

Task	Description	For Instructions
Use the default RBAC configuration.	Views and uses RBAC without modifying the initial installation.	"Viewing and Using RBAC Defaults (Task Map)" on page 156
Plan, configure, and use RBAC.	Configures RBAC for your site.	"Initially Configuring RBAC (Task Map)" on page 163
Administer RBAC.	Updates your site's RBAC configuration.	"Managing RBAC (Task Map)" on page 176
Manage and use privileges.	Adds and removes privileges from users, roles, systems, and processes. Uses privileges. Views and debugs use of privilege.	"Using Privileges (Tasks)" on page 186

Viewing and Using RBAC Defaults (Tasks)

Users are assigned rights by default. Rights for all users of a system are assigned in the `/etc/security/policy.conf` file.

Viewing and Using RBAC Defaults (Task Map)

At Oracle Solaris installation, your system is configured with user rights and process rights. With no further configuration, use the following task map to view and use RBAC.

Task	Description	For Instructions
View the contents of the security attributes databases.	List all the authorizations, rights profiles, and commands with security attributes on the system.	"How to View All Defined Security Attributes" on page 156
View your rights.	Involves listing your rights profiles, authorizations, privileges, and assigned roles.	"How to View Your Assigned Rights" on page 157
Assume the `root` role.	The initial user gains administrative rights.	"How to Assume a Role" on page 159
Become an administrator.	Several methods are available to users who are assigned administrative rights to use those rights.	"How to Obtain Administrative Rights" on page 160

▼ How to View All Defined Security Attributes

Use the following commands to list all authorizations, rights profiles, and commands with security attributes on the system. To list all defined privileges, see "How to List the Privileges on the System" on page 187.

1 List all authorizations.

```
% getent auth_attr | more
solaris.:::All Solaris Authorizations::help=AllSolAuthsHeader.html
solaris.account.:::Account Management::help=AccountHeader.html
...
solaris.zone.login:::Zone Login::help=ZoneLogin.html
solaris.zone.manage:::Zone Deployment::help=ZoneManage.html
```

2 List all rights profiles.

```
% getent prof_attr | more
All:::Execute any command as the user or role:help=RtAll.html
Audit Configuration:::Configure Solaris Audit:auths=solaris.smf.value.audit;
help=RtAuditCfg.html
...
Zone Management:::Zones Virtual Application Environment Administration:
help=RtZoneMngmnt.html
Zone Security:::Zones Virtual Application Environment Security:auths=solaris.zone.*,
solaris.auth.delegate;help=RtZoneSecurity.html ...
```

3 **List all commands with security attributes.**

```
% getent exec_attr | more
All:solaris:cmd:::*:
Audit Configuration:solaris:cmd:::/usr/sbin/auditconfig:privs=sys_audit
...
Zone Security:solaris:cmd:::/usr/sbin/txzonemgr:uid=0
Zone Security:solaris:cmd:::/usr/sbin/zonecfg:uid=0 ...
```

▼ How to View Your Assigned Rights

Use the following commands to view your RBAC assignments. To view all rights that can be assigned, see "How to View All Defined Security Attributes" on page 156.

1 **List your authorizations.**

```
% auths
solaris.device.cdrw,solaris.device.mount.removable,solaris.mail.mailq
```

These authorizations are assigned to all users by default.

2 **List your rights profiles.**

```
% profiles
Basic Solaris User
All
```

These rights profiles are assigned to all users by default.

3 **List your assigned roles.**

```
% roles
root
```

This role is assigned to the initial user by default. `No roles` indicates that you are not assigned a role.

4 **List the privileges in your default shell.**

```
% ppriv $$
1234:    /bin/csh
flags = <none>
    E: basic
    I: basic
    P: basic
    L: all
```

Every user is assigned the basic privilege set by default. The limit set is all privileges.

```
% ppriv -vl basic
file_link_any
        Allows a process to create hardlinks to files owned by a uid
        different from the process' effective uid.
file_read
        Allows a process to read objects in the filesystem.
```

```
file_write
        Allows a process to modify objects in the filesystem.
net_access
        Allows a process to open a TCP, UDP, SDP or SCTP network endpoint.
proc_exec
        Allows a process to call execve().
proc_fork
        Allows a process to call fork1()/forkall()/vfork()
proc_info
        Allows a process to examine the status of processes other
        than those it can send signals to.  Processes which cannot
        be examined cannot be seen in /proc and appear not to exist.
proc_session
        Allows a process to send signals or trace processes outside its session.
```

5 **List the privileges on commands in your rights profiles.**

```
% profiles -l
  Basic Solaris User
   /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.bin     privs=file_dac_read,sys_devices,net_privaddr
  All
   *
```

A user's rights profiles can include commands that run with particular privileges. The Basic Solaris User profile includes commands that enable users to read and write to CD-ROMs.

Example 9–1 Listing a User's Authorizations

```
% auths username
solaris.device.cdrw,solaris.device.mount.removable,solaris.mail.mailq
```

Example 9–2 Listing a User or Role's Rights Profiles

The following command lists the rights profiles of a specific user.

```
% profiles jdoe
jdoe:
          Basic Solaris User
          All
```

The following command lists the rights profiles of the `cryptomgt` role.

```
% profiles cryptomgt
cryptomgt:
          Crypto Management
          Basic Solaris User
          All
```

The following command lists the rights profiles of the `root` role:

```
% profiles root
root:
          All
          Console User
          Network Wifi Info
          Desktop Removable Media User
          Suspend To RAM
          Suspend To Disk
          Brightness
          CPU Power Management
          Network Autoconf User
          Basic Solaris User
```

Example 9–3 Listing a User's Assigned Roles

The following command lists the assigned roles of a specific user.

```
% roles jdoe
root
```

Example 9–4 Listing a User's Privileges on Specific Commands

The following command lists the privileged commands in a regular user's rights profiles.

```
% profiles -l jdoe
jdoe:
  Basic Solaris User
   /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.bin     privs=file_dac_read,sys_devices,net_privaddr
  All
   *
```

▼ How to Assume a Role

Before You Begin The role must already be assigned to you. The naming 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, such as authorizations, privileges, and set ID bits.

3 **(Optional) Verify that you are now in a role.**

```
$ /usr/bin/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 View Your Assigned Rights" on page 157.

Example 9–5 Assuming the root Role

In the following example, the initial user assumes the `root` role and lists the privileges in the role's shell.

```
% roles
root
% su - root
Password:       <Type root password>
#        Prompt changes to root prompt
# ppriv $$
1200:   pfksh
flags = <none>
        E: all
        I: basic
        P: all
        L: all
```

For information about privileges, see "Privileges (Overview)" on page 145.

▼ How to Obtain Administrative Rights

Administrative rights are in effect when you are running a profile shell. By default, a role account is assigned a profile shell. Roles are special accounts that are assigned specific administrative rights, typically to a related set of administrative activities, such as reviewing audit files

In the `root` role, the initial user has all administrative rights, that is, the initial user is superuser. The `root` role can create other roles.

Before You Begin To administer the system, you must have rights that regular users are not assigned. If you are not superuser, you must be assigned a role, an administrative rights profile, or specific privileges or authorizations.

- **Choose one of the following methods to run administrative commands.**

  Open a terminal window.

  - **Become `root`.**

    ```
    % su -
    Password:       Type the root password
    #
    ```

    Note – This method works whether `root` is a user or a role. The pound sign (#) prompt indicates that you are now superuser.

  - **Assume a role that you have been assigned.**

    In the following example, you assume a network management role. This role includes the Network Management rights profile.

    ```
    % su - networkadmin
    Password:       Type the networkadmin password
    $
    ```

    You are now in a profile shell. In this shell, you can run `snoop`, `route`, `dladm`, and other commands. For more about profile shells, see "Profile Shells and RBAC" on page 144.

    Tip – Use the steps in "How to View Your Assigned Rights" on page 157 to view the capabilities of your role.

  - **Use the `pfbash` command to create a shell that runs with administrative rights.**

    For example, the following set of commands enables you to examine network packets in the `pfbash` shell:

    ```
    % pfbash
    $ anoop
    ```

    If you are not assigned the `net_observability` privilege, the `snoop` command fails with an error message similar to the following: `snoop: cannot open "net0": Permission denied`. If you are assigned the privilege directly, or through a rights profile or a role, this command will succeed. Also, you can run additional privileged commands in this shell.

  - **Use the `pfexec` command to create a process that runs with administrative rights.**

    Run the `pfexec` command with the name of a privileged command from your rights profile. For example, the following command enables you to examine network packets:

    ```
    % pfexec snoop
    ```

    The same privilege limitations apply to `pfexec` as to `pfbash`. However, to run another privileged command, you must type `pfexec` again before you type the privileged command.

Example 9–6 Caching Authentication for Ease of Role Use

In this example, the administrator configures a role to manage the network, but provides ease of use by caching the user's authentication. First, the administrator creates and assigns the role.

```
# roleadd -K roleauth=user -P "Network Management" netmgt
# usermod -R +netmgt jdoe
```

When jdoe uses the -c option when switching to the role, a password is required before the snoop output is displayed:

```
% su - netmgt -c snoop options
Password:

    snoop output
```

If authentication is not being cached, and jdoe runs the command again immediately, a password prompt appears.

The administrator configures the pam.conf file to cache authentication, so that a password is initially required, but not thereafter until a certain amount of time has passed. The administrator places all pam.conf customized stacks at the end of the file.

```
# vi /etc/pam.conf
...
#
## Cache authentication for switched user
#
su      auth required           pam_unix_cred.so.1
su      auth sufficient         pam_tty_tickets.so.1
su      auth requisite          pam_authtok_get.so.1
su      auth required           pam_dhkeys.so.1
su      auth required           pam_unix_auth.so.1
```

After creating the entries, the administrator checks the entries for typos, omissions, or repetitions.

The entire su stack is required. The pam_tty_tickets.so.1 module provides the cache. For more about PAM, see the pam.conf(4) man page and Chapter 15, "Using PAM."

After the su PAM stack is added to the pam.conf file, the netmgt role is prompted only once for a password when running a series of commands.

```
% su - netmgt -c snoop options
Password:

    snoop output
% su - netmgt -c snoop options
    snoop output
...
```

Customizing RBAC for Your Site (Tasks)

Initial configuration of RBAC includes creating users who can assume specific roles, creating the roles, and assigning them to the appropriate users.

Initially Configuring RBAC (Task Map)

Use the following task map to plan and initially implement RBAC at your site. Some tasks are ordered.

Task	Description	For Instructions
1. Plan for RBAC.	Involves examining your site's security needs, and deciding how to use RBAC at your site.	"How to Plan Your RBAC Implementation" on page 163
2. Configure users who can assume a role.	Ensures that users who can assume an administrative role exist.	"Setting Up and Administering User Accounts (Task Map)" in *Oracle Solaris Administration: Common Tasks*
3. Create roles.	Creates roles and assigns the roles to users	"How to Create a Role" on page 165 "How to Assign a Role" on page 167
(Recommended) Audit role actions.	Preselect an audit class that includes an audit event that records role actions.	"How to Audit Roles" on page 169
Create or change rights profiles.	Creates a rights profile. Or modifies the security attributes or supplementary rights profiles in a rights profile. Adds privileges to a command.	"How to Create or Change a Rights Profile" on page 170 Example 9–14
Secure legacy applications	Turns on the set ID permissions for legacy applications. Scripts can contain commands with set IDs. Legacy applications can check for authorizations, if appropriate.	"How to Add RBAC Properties to Legacy Applications" on page 171 Example 9–16
Troubleshoot security attribute assignment.	Debugs why assigned security attributes might not be available to users, roles, or processes.	"How to Troubleshoot RBAC and Privilege Assignment" on page 173

▼ 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.

Note – Default rights are assigned in the `/etc/security/policy.conf` file.

1 **Learn the basic RBAC concepts.**

Read "Role-Based Access Control (Overview)" on page 133. Using RBAC to administer a system is very different from using conventional UNIX administrative practices. To be familiar with RBAC concepts before you start your implementation, see Chapter 10, "Security Attributes in Oracle Solaris (Reference)."

2 **Examine your security policy.**

Your organization's security policy details the potential threats to your system, measures the risk of each threat, and provides strategies to counter these threats. Isolating the security-relevant tasks through RBAC can be a part of the strategy. Although you can use the installed RBAC configurations as is, you might need to customize it 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:

- **Root as a role –** This method is provided by default. It prevents any user from logging in as `root`. Instead, a user must log in by using their assigned login prior to assuming the `root` role.
- **Discrete roles –** This method creates roles that are based on provided rights profiles. The roles can be assigned according to level of responsibility, scope of task, and type of task. For example, the System Administrator role can perform many tasks that superuser can perform, while the Network IPsec Management role can manage IPsec.

  You can also separate security responsibilities from other responsibilities, The User Management role can create users, while the User Security role can assign security attributes, such as roles and rights profiles. However, the User Security role cannot create a user, and the User Management role cannot assign a rights profile to a user.
- **No root role –** This method requires you to change the default configuration of the system. In this configuration, any user who knows the password for `root` can log in and modify the system. You cannot tell which user was acting as superuser.

4 **Decide which roles are appropriate for your organization.**

Review the capabilities of the recommended roles and their default rights profiles. Default rights profiles enable administrators to configure a recommended role by using a single profile.

To further examine rights profiles, do one of the following:

- For the available rights profiles on your system, use the `getent prof_attr` command.
- In this guide, refer to "Rights Profiles" on page 197 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.

Note – The Media Backup and Media Restore rights profiles provide access to the entire root file system. Therefore, these rights profiles are appropriately assigned to trusted users only. Alternatively, you can choose to not assign these rights profiles. By default, only the `root` role is trusted to back up and restore.

---

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 role's original rights profiles are appropriate for users who are assigned to this role. Order the new rights profile so that commands execute with their required privileges. For information about ordering, see "Order of Search for Assigned Security Attributes" on page 199.

6 **Decide which users should be assigned to which roles.**

According to the principle of least privilege, you assign users to roles that are appropriate to the user's level of trust. When you prevent users from performing tasks that the users do not need to perform, you reduce potential problems.

▼ How to Create a Role

Roles can be created locally and in an LDAP repository.

Before You Begin To create a role and assign its initial password, you must be assigned the User Management rights profile. To assign security attributes to the role, you must be assigned the User Security rights profile.

1 **Become an administrator with the required security attributes.**

For more information, see "How to Obtain Administrative Rights" on page 160.

2 To create a role, use the `roleadd` command.

The RBAC arguments to the command are the following:

```
# roleadd [-e expire] [-f inactive] [-s shell] [-m] [-S repository] \
[-A authorization-list] -K key=value] rolename
```

`-e` *expire*
Is the date that a role expires. Use this option to create temporary roles.

`-f` *inactive*
Is the maximum number of days that is allowed between uses of a role. When the *inactive* value is exceeded, the role cannot be used. The default value is `0`, no expiration date.

`-m`
Creates a home directory for *rolename* at the default location.

`-s` *shell*
Is the login shell for *rolename*. This shell must be a profile shell. For a list of profile shells, see the `pfexec`(1) man page.

Tip – You can also list the profile shells from the `/usr/bin` directory on your system, as in `ls /usr/bin/pf*sh`.

`-S` *repository*
Is one of `files` or `ldap`. The default is local files.

`-A` *authorization-list*
Is one or more authorizations separated by commas. For the list of authorizations, see the `/etc/security/auth_attr` file.

`-K` *key=value*
Is a *key=value* pair. This option can be repeated. The following keys are available: `audit_flags`, `auths`, `profiles`, `project`, `defaultpriv`, `limitpriv`, `lock_after_retries`, and `roleauth`. For information about the keys, their values, and the authorizations that are required to set the values, see the `user_attr`(4) man page.

rolename
Is the name of the new role. For restrictions on acceptable strings, see the `roleadd`(1M) man page.

Tip – When the name of the role reflects the name of a rights profile, you can easily understand the purpose of the role. For example, assign the Audit Review rights profile to the `auditreview` role to enable the role to read, filter, and archive audit records.

For example, the following command creates a local User Administrator role and a home directory:

```
# roleadd -c "User Administrator role, local" -s /usr/bin/pfbash \
-m -K profiles="User Security,User Management"  useradm
80 blocks
# ls /export/home/useradm
local.cshrc     local.login     local.profile
```

3 **Create the initial password for the role.**

```
# passwd -r files useradmPassword:      <Type useradm password>
Confirm Password:      <Retype useradm password>
#
```

Note – Typically, a role account is assigned to more than one user. Therefore, an administrator typically creates a role password and provides the users with the role password out of band.

4 **To assign the role to a user, run the `usermod` command.**

For the procedure, see "How to Assign a Role" on page 167 and Example 9–10.

Example 9–7 Creating a User Administrator Role in the LDAP Repository

In this example, the administrator's site uses an LDAP repository. By running the following command, the administrator creates a User Administrator role in LDAP.

```
# roleadd -c "User Administrator role, LDAP" -s /usr/bin/pfbash \
-m -S ldap -K profiles="User Security,User Management"  useradm
```

Example 9–8 Creating Roles for Separation of Duty

In this example, the administrator's site uses an LDAP repository. By running the following commands, the administrator creates two roles. The `usermgt` role can create users, give them home directories, assign an initial password, and perform other non-security tasks. The `usersec` role cannot create users, but can change user passwords and change other RBAC properties.

```
# roleadd -c "User Management role, LDAP" -s /usr/bin/pfbash \
-m -S ldap -K profiles="User Management"  usermgt
# roleadd -c "User Security role, LDAP" -s /usr/bin/pfbash \
-m -S ldap -K profiles="User Security"  usersec
```

Example 9–9 Creating a Device and File Security Role

In this example, the administrator creates a Device and File Security role for this system:

```
# roleadd -c "Device and File System Security admin, local" -s /usr/bin/pfbash \
-m -K profiles="Device Security,File System Security"  devflsec
```

▼ How to Assign a Role

This procedure assigns a role to a user, restarts the name cache daemon, and then shows how the user can assume the role.

Before You Begin You have added a role and assigned the role a password, as described in "How to Create a Role" on page 165.

To modify most security attributes of a user, you must be assigned the User Security rights profile. To modify a user's audit flags, you must be superuer. To modify other attributes, you must be assigned the User Management rights profile.

1. **Become an administrator with the required security attributes.**

   For more information, see "How to Obtain Administrative Rights" on page 160.

2. **Assign the role to a user.**

   `usermod [-S` *repository***`]`** **`[`***RBAC-arguments***`]`** *login*

   For example, assign the role to a local user:

   ```
   # usermod -R +useradm jdoe-local
   ```

   For the options to the `usermod` command, see the `usermod`(1M) man page or the description in Step 2 in "How to Create a Role" on page 165.

3. **To put the changes into effect, restart the name service cache daemon.**

   ```
   # svcadm restart system/name-service-cache
   ```

Example 9–10 Creating and Assigning a Role to Administer Crypto

In this example, the administrator on an LDAP network creates a role to administer the Cryptographic Framework, and assigns the role to UID `1111`. The administrator restarts the `nscd` daemon to put the assignment into effect.

```
# roleadd -c "Cryptographic Services manager" \
-g 14 -m -u 104 -s /usr/bin/pfksh \
-S ldap -K profiles="Crypto Management" cryptmgt
# passwd cryptmgt
New Password:       <Type cryptmgt password>
Confirm password:       <Retype cryptmgt password>
# usermod -u 1111 -R +cryptmgt
# svcadm restart system/name-service-cache
```

The user with UID `1111` logs in, then assumes the role and displays the assigned security attributes.

```
% su - cryptmgt
Password:       <Type cryptmgt password>
Confirm Password:       <Retype cryptmgt password>
$ profiles -l
      Crypto Management
          /usr/bin/kmfcfg            euid=0
```

```
          /usr/sbin/cryptoadm         euid=0
          /usr/sfw/bin/CA.pl          euid=0
          /usr/sfw/bin/openssl        euid=0
$
```

For information about the Cryptographic Framework, see Chapter 11, "Cryptographic Framework (Overview)." To administer the framework, see "Administering the Cryptographic Framework (Task Map)" on page 235.

▼ 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 rolename, and the action that the role performed. The `116:AUE_PFEXEC:execve(2) with pfexec enabled:ps,ex,ua,as` audit event captures role actions,. By preselecting one of the `as`, `ex`, `ps`, or `ua` classes, role actions are audited.

Before You Begin To configure auditing, you must be assigned the Audit Configuration rights profile. To enable or refresh the audit service, you must be assigned the Audit Control rights profile.

1. **Include the auditing of roles in your audit plan.**

   For planning information, see Chapter 27, "Planning for Auditing."

2. **Become an administrator with the required security attributes.**

   For more information, see "How to Obtain Administrative Rights" on page 160.

3. **Preselect one of the `as`, `ex`, `ps`, or `ua` classes.**

   - **If the audit service is enabled, review the preselected classes.**

     ```
     # auditconfig -getflags
     ```

     If one of the `as`, `ex`, `ps`, or `ua` classes is preselected, role actions are being audited. If not, add one of these classes to the existing classes.

     ```
     # auditconfig -setflags existing preselections,as
     ```

   - **If auditing is not yet enabled, preselect a class that audits role actions.**

     ```
     # auditconfig -setflags as
     ```

     In this example, the administrator chooses the `as` class. This class includes other audit events. To view the audit events that are included in a class, use the `auditrecord` command, as shown in Example 28–25.

4 **Enable or refresh the audit service.**

```
# audit -s
```

▼ How to Create or Change a Rights Profile

You can create or change a rights profile when the provided rights profiles do not contain the collection security attributes that you need. To learn more about rights profiles, see "RBAC Rights Profiles" on page 142.

The easiest way to create a new rights profile is to copy and modify an existing rights profile.

Before You Begin To create or change a rights profile, you must be assigned the File Security rights profile.

1 **Become an administrator with the required security attributes.**

For more information, see "How to Obtain Administrative Rights" on page 160.

2 **Create a new rights profile from an existing profile.**

```
# profiles [-S repository] existing-profile-name
```

You are prompted for a new name. The content of the existing rights profile is duplicated in the new profile.

3 **Continue to modify the new rights profile.**

Add or remove supplementary rights profiles, authorizations, and other security attributes, as shown in the following examples.

Example 9–11 Creating a New Rights Profile From an Existing Profile

In this example, the administrator customizes the Console User rights profile in the LDAP repository.

```
# profiles -S ldap Console User
New name: ExampleCo Console User
ExampleCo Console User >
Description > Manage MyCompany Systems as the Console User
Help > ExCoConsUser.html
```

The administrator sets the `roleauth` attribute for this rights profile.

```
roleauth=yes
```

Example 9–12 Removing a Basic Privilege From a Rights Profile

In the following example, after thorough testing, the security administrator removes a basic privilege from all users who are assigned the SunRayUser rights profile. They are prevented from using the proc_session privilege. That is, these users cannot examine any processes outside the user's current session.

```
$ profiles -K defaultpriv=basic,!proc_session SunRayUser
```

Example 9–13 Removing Privileges From the Limit Set of a Rights Profile

In the following example, after thorough testing, the security administrator removes a limit privilege from all users who are assigned the SunRayUser rights profile. This removal prevents these users from viewing other users' processes.

```
$ profiles -K limitpriv=all,!proc_session SunRayUser
```

Example 9–14 Adding Privileges to a Command

In this example, the security administrator adds privileges to an application in a rights profile. The application is privilege-aware.

```
# profiles -p SiteApp
profiles:SiteApp> set desc="Site application"
profiles:SiteApp> add cmd=/opt/site-app/bin/site-cmd
profiles:SiteApp:site-cmd> add privs=proc_fork,proc_taskid
profiles:SiteApp:site-cmd> end
profiles:SiteApp> exit
```

To verify, the administrator selects the site-cmd.

```
# profiles -p SiteApp "select cmd=/opt/site-app/bin/site-cmd; info;end"
Found profile in files repository.
  id=/opt/site-app/bin/site-cmd
  privs=proc_fork,proc_taskid
```

See Also To troubleshoot security attribute assignment, see "How to Troubleshoot RBAC and Privilege Assignment" on page 173. For background, see "Order of Search for Assigned Security Attributes" on page 199.

▼ 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.

Before You Begin To create the rights profile, you must be assigned the Information Security or Rights Management rights profile. To assign the rights profile, you must be assigned the User Security rights profile.

1. **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 170.

   a. **Create a new rights profile for your legacy application.**

      For the steps, see "How to Create or Change a Rights Profile" on page 170.

   b. **Add the commands with their required security attributes.**

      For an example, see Example 9–14.

2. **Include the rights profile in a role's list of profiles.**

   To assign a rights profile to a role, see Example 9–10.

Example 9–15 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.

Example 9–16 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:

```
if [ ‘/usr/bin/auths|/usr/xpg4/bin/grep $1‘ ]; then
        echo Auth granted
else
        echo Auth denied
fi
```

To be more complete, the test must include logic that checks for other authorizations that use wildcards. For example, to test if the user has the `solaris.system.date` authorization, you would need to check for the following strings:

- `solaris.system.date`
- `solaris.system.*`
- `solaris.*`

If you are writing a program, use the function `getauthattr()` to test for the authorization.

▼ How to Troubleshoot RBAC and Privilege Assignment

Several factors can affect why a user or role's processes do not run with assigned security attributes.

- The security attribute is misspelled. Misspelled authorizations fail silently.
- The user or role is not using the naming service that includes the assignments.
- The assignment that you expect is not the first assignment of that attribute.

  The order in which a user or role's security attributes are searched for and then assigned at authentication determines which assignments are successful. The exception is authorizations. During search, authorizations that are assigned to the user or role accumulate. In contrast, privilege assignment, and the assignment of security attributes in rights profiles is search-dependent. First assignment wins, later assignments are ignored.
- The command is not being run in a profile shell.

Before You Begin You must be in the `root` role.

1 Verify and restart the naming service.

a. Verify that the security assignments for the user or role are in the naming service that is enabled on the system.

b. Restart the name service cache, `svc:/system/name-service/cache`.

The `nscd` daemon can have a lengthy time-to-live interval. By restarting the daemon, you update the naming service with current data.

2 Determine where a security attribute is assigned.

Use the security attribute as the value to the `userattr -v` command. For example, the following commands indicate which security attributes are assigned and where the assignment was made for the user `jdoe`:

```
# userattr -v audit_flags jdoe      Modifications to the system defaults
user_attr: fw:no
# userattr -v auths jdoe      Assigned authorizations
solaris.admin.wusb.read,solaris.device.cdrw,solaris.device.mount.removable,
solaris.mail.mailq,solaris.profmgr.read,solaris.smf.manage.audit,
solaris.smf.value.audit
# userattr -v audit_flags jdoe      Modifications to audit preselection mask
# userattr -v auths jdoe      Assigned authorizations
# userattr -v defaultpriv jdoe      Modifications to basic user privileges
# userattr -v limitpriv jdoe      Modifications to limit privileges
# userattr -v lock_after_retries jdoe      Automatic lockout attribute
# userattr -v profiles jdoe      Assigned rights profiles
user_attr: Audit Review,Stop
# userattr roles jdoe      Assigned roles
user_attr : cryptomgt,infosec
```

3 For rights profiles that you have created, verify that you have assigned the appropriate security attributes to the command.

For example, some commands require `uid=0` rather than `euid=0` to succeed. Aspects of some commands can require authorizations.

4 Check the following if security attributes are not available to a user.

a. Check if the security attributes are directly assigned to the user.

Use the `userattr` command.

b. If the security attributes are not directly assigned, check the rights profiles that are directly assigned to the user.

i. In order, check for the security attribute assignment in the list of rights profiles.

The value of the attribute in the earliest rights profile in the list is the value that the user can use. If this value is incorrect, either change the value in that rights profile, or reorder the list of profiles.

For privileged commands, check if a privilege is assigned in the `defaultpriv` keyword. This assignment is in addition to privileges on a particular command.

ii. If no attribute assignment is listed, check the roles that the user is assigned.

If the attribute is assigned to a role, the user must assume the role to obtain the security attributes. If the attribute is assigned to more than one role, the assignment in the earliest role in the list is effective. If this value is incorrect, either assign the correct value to the first role in the list, or reorder the role assignment.

5. **If you assigned a privilege directly to a user or role, check if a failed command requires authorizations to succeed.**

   Note – Aspects of some commands can require authorization. Best practice is to assign a rights profile that includes the administrative command, rather than assign a privilege directly.

   Review the rights profiles that include the administrative command. If a rights profile exists that includes authorizations, assign the rights profile to the user, not simply the privileges. Order the rights profile before any other rights profile that includes the command.

6. **Check the following if a command continues to fail for a user.**

   a. **Verify that the user is executing the command in a profile shell.**

      Administrative commands must be executed in a profile shell. To mitigate user error, you can assign a profile shell as the user's login shell. Or, you can remind the user to run administrative commands in a profile shell.

   b. **Check if any security attributes that are directly assigned to the user prevent the command from succeeding.**

      In particular, check the values of the user's `defaultpriv` and `limitpriv` attributes.

   c. **Determine which rights profile or role includes the command.**

      i. **In order, check for the command with security attributes in the list of rights profiles.**

         The earliest value in the list of rights profiles is the value that the user can use. If this value is incorrect, either change the value in that rights profile, or reorder the list of profiles.

         In particular, check the values of the profile's `defaultpriv` and `limitpriv` attributes.

      ii. **If no attribute assignment is listed, check the roles that the user is assigned.**

         If the command is assigned to a role, the user must assume the role to obtain the security attributes. If the attribute is assigned to more than one role, the assignment in the earliest role in the list is effective. If this value is incorrect, either assign the correct value to the first role in the list, or reorder the role assignment.

7. **Check the following if a command fails for a role.**

   Administrative commands require privileges to succeed. Aspects of some commands can require authorization. Best practice is to assign a rights profile that includes the administrative command.

   a. **Check if any security attributes that are directly assigned to the role prevent the command from succeeding.**

      In particular, check the values of the role's `defaultpriv`and `limitpriv` attributes.

   b. **In order, check for the command with security attributes in the list of rights profiles.**

      The earliest value in the list of rights profiles is the value that the user can use. If this value is incorrect, either change the value in that rights profile, or reorder the list of profiles.

Managing RBAC (Tasks)

After you have configured and are using RBAC, use the following procedures to maintain and modify RBAC on your systems.

Managing RBAC (Task Map)

The following task map points to procedures for maintaining role-based access control (RBAC) after RBAC has been initially implemented.

Task	Description	For Instructions
Change the role password.	An authorized user or role changes the password of another role.	"How to Change the Password of a Role" on page 177
Modify the assigned rights of a role.	Modifies the security attributes of a role.	"How to Change the Security Attributes of a Role" on page 178 Example 9–19
Modify the rights of a user.	Adds security attributes to a regular user or removes them.	"How to Change the RBAC Properties of a User" on page 179 Example 9–24 Example 9–12
Modify a user's rights in a rights profile.	Assigns security attribute values in a rights profile, such as audit flags, default privileges.	Example 9–21 Example 9–13

Task	Description	For Instructions
Create a restricted profile shell.	Prevents users or roles from full access to all commands in the software.	"How to Restrict an Administrator to Explicitly Assigned Rights" on page 182
Remove default rights from a system.	Creates a system for special uses.	Example 9–25
Restrict a user's privileges.	Limits the user's basic or limit set of privileges.	Example 9–21
Enable a user to supply the user's password to assume a role.	Modifies a user's security attributes to make the user's password authenticate the user to a role. This behavior is similar to Linux role behavior.	"How to Enable a User to Use Own Password to Assume a Role" on page 183
Change `root` into a user.	Prior to decommissioning a system, change `root` role into a user.	"How to Change the `root` Role Into a User" on page 184

These procedures manage security attributes on users, roles, and rights profiles. For basic user management procedures, refer to Chapter 2, "Managing User Accounts and Groups (Overview)," in *Oracle Solaris Administration: Common Tasks*.

▼ How to Change the Password of a Role

Before You Begin You must be in the `root` role.

● **Run the `passwd` command.**

```
# passwd  [-r naming-service] target-rolename
```

`-r` *naming-service* Applies the password change to the `files` or `ldap` repository. The default repository is `files`. If you do not specify a repository, the password is changed in all repositories.

target-rolename Is the name of an existing role that you want to modify.

For more command options, see the `passwd(1)` man page.

Example 9–17 Changing a Role's Password

In this example, the `root` role changes the password of the local `devmgt` role.

```
# passwd -r files  devmgt
New password:      Type new password
Confirm password:      Retype new password
```

In this example, the `root` role changes the password of the `devmgt` role in the LDAP directory service.

```
# passwd -r ldap devmgt
New password:       Type new password
Confirm password:       Retype new password
```

In this example, the `root` role changes the password of the `devmgt` role in file and LDAP.

```
# passwd devmgt
New password:       Type new password
Confirm password:       Retype new password
```

▼ How to Change the Security Attributes of a Role

Before You Begin You must be assigned the User Security rights profile to change the security attributes of a role, except for the role's password and audit flags. Role properties include rights profiles and authorizations. To assign audit flags or change a role's password, you must be in the `root` role.

Note – To change the password, see "How to Change the Password of a Role" on page 177.

---

1 Become an administrator with the required security attributes.

For more information, see "How to Obtain Administrative Rights" on page 160.

2 Use the `rolemod` command.

This command modifies the attributes of a role that is defined in the local naming service or in LDAP. The values of the `-A`, `-P`, and `-R` options can be modified by `-` or `++`. `-` indicates to subtract the value from the currently assigned values. `++` indicates to add the value to the currently assigned values.

For more information about the `rolemod` command, see the following:

- For a short description, see the description of the `roleadd` command in "How to Create a Role" on page 165.
- For all arguments to this command, see the `rolemod`(1M) man page.
- For the list of key values for the `-K` option, see the `user_attr`(4) man page.

The following command replaces the `devmgt` role's assigned rights profiles in the LDAP repository:

```
$ rolemod  -P "Device Management,File Management" -S ldap devadmin
```

Example 9–18 Changing a Local Role's Security Attributes

In this example, the security administrator modifies the `prtmgt` role to include the VSCAN Management rights profile.

```
$ rolemod -c "Handles printers and virus scanning" \
-P "Printer Management,VSCAN Management,All" prtmgt
```

These rights profiles are added to the profiles that are granted through the policy.conf file.

Example 9–19 Assigning Privileges Directly to a Role

In this example, the security administrator entrusts the systime role with a very specific privilege that affects system time.

```
$ rolemod -K priv=proc_clock_highres systime
```

The values for the priv keyword are in the list of privileges in the role's processes at all times.

▼ How to Change the RBAC Properties of a User

User properties include login shell, rights profiles, and roles. 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 144.

Before You Begin You must be assigned the User Security rights profile to change the security attributes of a user, except for the user's password and audit flags. To assign audit flags or change a role's password, you must be in the root role. To change other user attributes, you must be assigned the User Management rights profile.

1 **Become an administrator with the required security attributes.**

For more information, see "How to Obtain Administrative Rights" on page 160.

2 **Use the usermod command.**

This command modifies the attributes of a user that is defined in the local naming service or the LDAP naming service. The RBAC arguments to this command are similar to the arguments to the useradd command, as described on the user_attr(4) man page, and shown in Example 9–23.

In the following example, an LDAP user is assigned the devmgt role. This role replaces any prior role assignments. The devmgt role must exist in the LDAP naming service.

```
$ usermod -R devmgt -S ldap jdoe-ldap
```

In the following example, this role is added to any prior role assignments.

```
$ usermod -R +devmgt -S ldap jdoe-ldap
```

Example 9–20 Assigning a Role to a Local User

In this example, the user jdoe can now assume the role of System Administrator, sysadmin.

```
$ userattr roles jdoe
secdevice
$ usermod -R secdevice,sysadmin jdoe
$ userattr roles jdoe
secdevice,sysadmin
```

Example 9–21 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
$ userattr limitpriv jdoe
all,!sys_linkdir
```

Example 9–22 Creating a User Who Can Manage DHCP

In this example, the security administrator creates a user in LDAP. At login, the jdoe-dhcp user is able to manage DHCP.

```
# useradd -P "DHCP Management" -s /usr/bin/pfbash -S ldap  jdoe-dhcp
```

Because the user is assigned pfbash as the login shell, the security attributes in the DHCP Management rights profile are available to the user in the user's default shell.

Example 9–23 Assigning Authorizations Directly to a User

In this example, the security administrator creates a local user who can control screen brightness.

```
# useradd -c "Screened JDoe, local" -s /usr/bin/pfbash \
-A solaris.system.power.brightness  jdoe-scr
```

This authorization is added to the user's existing authorization assignments.

Example 9–24 Assigning Privileges Directly to a User

In this example, the security administrator trusts the user jdoe with a very specific privilege that affects system time.

```
$ usermod -K defaultpriv=basic,proc_clock_highres jdoe
```

The values for the `defaultpriv` keyword replace the existing values. Therefore, for the user to retain the `basic` privileges, the value `basic` is specified. In the default configuration, all users have basic privileges.

▼ How to Restrict a User to Desktop Applications

You can restrict an Oracle Solaris user to desktop access only.

Before You Begin You must be in the `root` role.

1 **Assign the user a profile shell as the login shell.**

For example, you could assign the `pfbash` shell to the user.

```
# usermod -s /usr/bin/pfbash username
```

All user processes are now under the control of RBAC.

2 **Create a rights profile that enables a user to run the basic applets on the Oracle desktop.**

The following command creates the rights profile. The `end` command indicates that the added command does not require security attributes. To create the rights profile in your LDAP repository, use the `-S ldap` option.

```
# profiles -p "Desktop Applets"
profiles:Desktop Applets> set desc="Can use basic desktop applications"
profiles:Desktop Applets> add cmd=/usr/bin/nautilus;end
profiles:Desktop Applets> add cmd=/usr/bin/dbus-launch;end
profiles:Desktop Applets> add cmd=/usr/lib/dbus-daemon;end
profiles:Desktop Applets> add cmd=/usr/lib/clock-applet;end
profiles:Desktop Applets> add cmd=/usr/lib/gconfd-2;end
profiles:Desktop Applets> add cmd=/usr/lib/gvfsd;end
profiles:Desktop Applets> add cmd=/usr/lib/gvfsd-metadata;end
profiles:Desktop Applets> add cmd=/usr/lib/gvfsd-trash;end
profiles:Desktop Applets> add cmd=/usr/lib/gvfs-hal-volume-monitor;end
profiles:Desktop Applets> add cmd=/usr/lib/gnome-pty-helper;end
profiles:Desktop Applets> add cmd=/usr/lib/utmp_update;end
profiles:Desktop Applets> add cmd=/usr/bin/sh;end
profiles:Desktop Applets> add cmd=/usr/bin/bash;end
profiles:Desktop Applets> add cmd=/usr/bin/csh;end
profiles:Desktop Applets> add cmd=/usr/bin/ksh;end
profiles:Desktop Applets> commit
profiles:Desktop Applets> exit
```

3 **Verify that the rights profile contains the correct entries.**

Review the entries for errors, such as typos, omissions, or repetition.

```
# profiles -p "Desktop Applets" info
Found profile in files repository.
name=Desktop Applets
desc=Can use basic desktop applications
cmd=/usr/bin/nautilus
```

```
cmd=/usr/bin/dbus-launch
cmd=/usr/lib/dbus-daemon
cmd=/usr/lib/clock-applet
cmd=/usr/lib/gconfd-2
cmd=/usr/lib/gvfsd
cmd=/usr/lib/gvfsd-metadata
cmd=/usr/lib/gvfsd-trash
cmd=/usr/lib/gvfs-hal-volume-monitor
cmd=/usr/lib/gnome-pty-helper
cmd=/usr/lib/utmp_update
cmd=/usr/bin/sh
cmd=/usr/bin/bash
cmd=/usr/bin/csh
cmd=/usr/bin/ksh
```

Tip – You can create a rights profile for an application or a class of applications that have desktop icons. Then, add Desktop Applets as a supplementary rights profile to this new rights profile. Together, these rights profiles enable the user to use the appropriate desktop applications.

4. **Assign the Desktop Applets rights profile and the Stop rights profile to the user.**

```
# usermod -P "Desktop Applets,Stop" username
```

This user does not have the Basic Solaris User rights profile or the Console User rights profile. Therefore, no commands other than the commands in the Desktop Applets rights profile can be run by this user. For example, the user does not have access to a terminal window.

For more information, see "Rights Profiles" on page 197, "Order of Search for Assigned Security Attributes" on page 199, and "How to Limit a User to Desktop Applications" in *Trusted Extensions Configuration and Administration*.

The `usermod` command modifies the user attributes that are defined in the local naming service or in LDAP. For arguments to this command, see the `usermod`(1M) man page.

▼ How to Restrict an Administrator to Explicitly Assigned Rights

You can restrict a role or user to a limited number of administrative actions in two ways.

- You can use the Stop rights profile.

  The Stop rights profile is the simplest way to create a restricted shell. The authorizations and rights profiles that are assigned in the `policy.conf` file are not consulted. In the default configuration, the role or user is not assigned the Basic Solaris User rights profile, the Console User rights profile, or the `solaris.device.cdrw` authorization.
- You can modify the `policy.conf` file on a system, and require the role or user to use that system for administrative tasks.

Before You Begin You must be in the root role.

- **Add the Stop rights profile as the last profile in the list of profiles that you assign.**

  For example, you could limit the auditrev role to performing only audit reviews.

  ```
  # rolemod -P "Audit Review,Stop" auditrev
  ```

  Because the auditrev role does not have the Console User rights profile, the auditor cannot shut down the system. Because this role does not have the solaris.device.cdrw authorization, the auditor cannot read from or write to the CD-ROM drive. Because this role does not have the Basic Solaris User rights profile, no commands other than the commands in the Audit Review rights profile can be run in this role. For example, the ls command will not run. The role uses the File Browser to view the audit files.

  For more information, see "Rights Profiles" on page 197 and "Order of Search for Assigned Security Attributes" on page 199.

  The rolemod command modifies the attributes of a role that is defined in the local naming service or in LDAP. For arguments to this command, see the rolemod(1M) man page. The list of RBAC arguments is similar to the list for the roleadd command, as described in "How to Create a Role" on page 165

Example 9–25 Modifying a System to Limit the Rights Available to Its Users

In this example, the administrator creates a system that is useful only to administer the network. The administrator removes the Basic Solaris User rights profile and the solaris.device.cdrw authorization from the policy.conf file. The Console User rights profile is not removed. The affected lines in the resulting policy.conf file are the following:

```
...
#AUTHS_GRANTED=solaris.device.cdrw
#PROFS_GRANTED=Basic Solaris User
CONSOLE_USER=Console User
...
```

Only a user who has been explicitly assigned authorizations, commands, or rights profiles is able to use this system. After login, the authorized user can perform administrative duties. If the authorized user is sitting at the system, the user has the rights of the Console User.

▼ How to Enable a User to Use Own Password to Assume a Role

By default, users must type the role's password to assume a role. Perform this procedure to make assuming a role in Oracle Solaris similar to assuming a role in a Linux environment.

Before You Begin You must have assumed a role that includes the User Security rights profile. This role cannot be the role whose `roleauth` value you want to change.

- **Enable a user password to authenticate a role.**

```
$ rolemod -K roleauth=user rolename
```

To assume this role, the assigned users can now use their own password, not the password that was created specifically for the role.

Example 9–26 Enabling a Role to Use the Assigned User's Password When Using a Rights Profile

In this example, the `root` role changes the value of `roleauth` for the role `secadmin` on the local system.

```
# profiles -K roleauth=user "System Administrator"
```

When a user who is assigned the Security Administrator rights profile wants to assume the role, the user is prompted for a password. In the following sequence, the role name is `secadmin`:

```
% su - secadmin
Password:       Type user password
$       /** You are now in a profile shell with administrative rights**/
```

If the user has been assigned other roles, they use their own password to authenticate to those roles, too.

Example 9–27 Changing the Value of roleauth for a Role in the LDAP Repository

In this example, the `root` role enables all users who can assume the role `secadmin` to use their own password when assuming a role. This capability is granted to these users for all systems that are managed by the LDAP server.

```
# rolemod -S ldap -K roleauth=user secadmin
# profiles -S ldap -K roleauth=user "Security Administrator"
```

Troubleshooting If `roleauth=user` is set for the role, the user password enables the authenticated role to access all rights that are assigned to that role. This keyword is search-dependent. For more information, see "Order of Search for Assigned Security Attributes" on page 199.

▼ How to Change the root Role Into a User

An administrator might change `root` to a user when decommissioning a system that has been removed from the network. In this instance, logging in to the system as `root` simplifies the cleanup.

Before You Begin You must become an administrator who is assigned the User Management and User Security rights profiles.

1. **Remove the `root` role assignment from local users.**

   For example, remove the role assignment from two users.

```
% su - root
Password: a!2@3#4$5%6^7
# roles jdoe
root
# roles kdoe
root
# roles ldoe
secadmin
# usermod -R "" jdoe
# usermod -R "" kdoe
#
```

2. **Change the `root` role into a user.**

```
# rolemod -K type=normal root
```

   Users who are currently in the `root` role remain so, Other users who have root access can `su` to `root` or log in to the system as the `root` user.

3. **Verify the change.**

   You can use one of the following commands.

```
# getent user_attr root
root::::auths=solaris.*;profiles=All;audit_flags=lo\:no;lock_after_retries=no;
min_label=admin_low;clearance=admin_high
```

   If the `type` keyword is missing in the output or is equal to `normal`, the account is not a role.

```
# userattr type root
```

   If the output is empty or lists `normal`, the account is not a role.

Example 9–28 Preventing the root Role From Being Used to Configure a System

In this example, site security policy requires that the `root` account be prevented from maintaining the system. The administrator has created and tested the roles which maintain the system. These roles include every security profile and the System Administrator rights profile. A trusted user has been assigned a role that can restore a backup. No role can change the audit flags for the system, a user, or a rights profile.

To prevent the `root` account from being used to maintain the system, the security administrator removes the root `role` assignment. Because the `root` account must be able to log in to the system in single-user mode, the account retains a password.

```
# rolemod -K roles= jdoe
# userattr roles jdoe
```

Example 9–29 Changing the root User Into the root Role

In this example, the root user turns the root user back into a role.

First, root changes the root account into a role and verifies the change.

```
# rolemod -K type=role root
# getent user_attr root
root::::type=role;auths=solaris.*;profiles=All;audit_flags=lo\:no;
lock_after_retries=no;min_label=admin_low;clearance=admin_high
```

Then, root assigns the root role to a local user.

```
# usermod -R root jdoe
```

Troubleshooting 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 password, and assign the root role to the new account. Then, log in as the new user and assume the root role.

Using Privileges (Tasks)

The following task maps point to step-by-step instructions for managing privileges and using privileges on your system.

Task	Description	For Instructions
Use privileges when you run a command.	Involves listing the privileges that have been assigned to you and the privileges that are available on the system.	"Determining Your Privileges (Task Map)" on page 186
Use privileges at your site.	Involves assigning, removing, adding, and debugging the use of privileges.	"Managing Privileges (Task Map)" on page 191

Determining Your Privileges (Task Map)

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.

The following task map points to procedures for viewing the privileges that have been assigned to you.

Task	Description	For Instructions
View the defined privileges.	List the Oracle Solaris privileges and their definitions.	"How to List the Privileges on the System" on page 187
View your privileges as a user in any shell.	Shows your directly assigned privileges. All of your processes run with these privileges.	"How to Determine the Privileges That You Have Been Directly Assigned" on page 188
View your privileged commands in a profile shell.	Shows the privileged commands that you can run through an assigned rights profile.	"How to Determine the Privileged Commands That You Can Run" on page 189
View your privileges as a role in any shell.	Shows the privileged commands that your role can run through an assigned rights profile.	"How to Determine the Privileged Commands That You Can Run" on page 189

▼ How to List the Privileges on the System

The following procedure shows how to view the privilege names and definitions.

● **In a terminal window, you can view privileges online.**

- **List the privileges by viewing the privileges(5) man page.**

```
% man privileges
Standards, Environments, and Macros                 privileges(5)

NAME
     privileges - process privilege model
...
     The defined privileges are:

     PRIV_CONTRACT_EVENT

         Allow a process to request reliable delivery  of  events
         to an event endpoint.

         Allow a process to include events in the critical  event
         set  term  of  a  template  which  could be generated in
         volume by the user.
...
```

  This privilege format is used by developers.

- **List the privileges by using the ppriv command.**

```
% ppriv -lv | more
contract_event
    Allows a process to request critical events without limitation.
    Allows a process to request reliable delivery of all events on
    any event queue.
...
```

```
win_upgrade_sl
        Allows a process to set the sensitivity label of a window
        resource to a sensitivity label that dominates the existing
        sensitivity label.
        This privilege is interpreted only if the system is configured
        with Trusted Extensions.
```

This privilege format is used to assign privileges to users and roles with the `useradd`, `roleadd`, `usermod`, and `rolemod` commands, and to rights profiles with the `profiles` command.

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 144.

1. **List the privileges that your processes can use.**

   See "How to Determine the Privileges on a Process" on page 191 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 9–30 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.

```
% /usr/bin/whoami
jdoe
% ppriv -v $$
1800:   pfksh
flags = <none>
        E: file_link_any,...,proc_clock_highres,proc_session
        I: file_link_any,...,proc_clock_highres,proc_session
        P: file_link_any,...,proc_clock_highres,proc_session
        L: cpc_cpu,dtrace_kernel,dtrace_proc,dtrace_user,...,sys_time
% ppriv -vl proc_clock_highres
        Allows a process to use high resolution timers.
```

Example 9–31 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/bin/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
```

▼ 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.

1 **Determine the rights profiles that you have been assigned.**

```
% profiles
Audit Review
Console User
Suspend To RAM
Suspend To Disk
Brightness
CPU Power Management
Network Autoconf
Desktop Print Management
Network Wifi Info
Desktop Removable Media User
Basic Solaris User
All
```

2 **Determine your rights from the Audit Review profile.**

```
profiles -l
Audit Review

  solaris.audit.read

  /usr/sbin/auditreduce  euid=0
  /usr/sbin/auditstat    euid=0
  /usr/sbin/praudit      euid=0
```

The Audit Review rights profile enables you to run the auditreduce, auditstat, and praudit commands with the effective UID of 0, and assigns you the solaris.audit.read authorization.

Example 9–32 Determining the Privileged Commands of a Role

In this example, a user assumes an assigned role and lists the commands that are included in one of the rights profiles.

```
% roles
devadmin
% su - devadmin
Password:      Type devadmin password
$ profiles -l
Device Security
          /usr/bin/kbd        uid=0;gid=sys
          /usr/sbin/add_allocatable    euid=0
          /usr/sbin/add_drv        uid=0
          /usr/sbin/devfsadm        uid=0
          /usr/sbin/eeprom        uid=0
          /usr/sbin/list_devices        euid=0
          /usr/sbin/rem_drv        uid=0
          /usr/sbin/remove_allocatable    euid=0
          /usr/sbin/strace        euid=0
          /usr/sbin/update_drv        uid=0
```

Example 9–33 Running the Privileged Commands in Your Role

In the following example, the admin role can change the permissions on the useful.script file.

```
% whoami
jdoe
% ls -l useful.script
-rwxr-xr-- 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/bin/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
```

Managing Privileges (Task Map)

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.

The following task map points to procedures for assigning, removing and debugging privileges, and for running a script that contains privileged commands.

Task	Description	For Instructions
Determine which privileges are in a process.	Lists the effective, inheritable, permitted, and limit privilege sets for a process.	"How to Determine the Privileges on a Process" on page 191
Determine which privileges are missing from a process.	Lists the privileges that a failed process requires to succeed.	"How to Determine Which Privileges a Program Requires" on page 193
Add privileges to a command.	Adds privileges to a command in a rights profile. Users or roles can be assigned the rights profile. The users can then run the command with the assigned privileges in a profile shell.	Example 9–14
Assign privileges to a user or role.	Expands a user's or role's inheritable set of privileges. Use this procedure with caution.	Example 9–24
Restrict a user's privileges.	Limits the user's basic set of privileges. Use this procedure with caution.	Example 9–12
Run a privileged shell script.	Adds privilege to a shell script and to the commands in the shell script. Then, runs the script in a profile shell.	"How to Run a Shell Script With Privileged Commands" on page 194

▼ 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.

Example 9–34 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,net_access,proc_exec,proc_fork,proc_info,proc_session
        I: file_link_any,net_access,proc_exec,proc_fork,proc_info,proc_session
        P: file_link_any,net_access,proc_exec,proc_fork,proc_info,proc_session
        L: cpc_cpu,dtrace_kernel,dtrace_proc,dtrace_user,...,sys_time
```

Example 9–35 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/bin/whoami
sysadmin
$ ppriv -v $$
1400:   pfksh
flags = <none>
        E: file_link_any,file_read,file_write,net_access,proc_exec,proc_fork,
           proc_info,proc_session
        I: file_link_any,file_read,file_write,net_access,proc_exec,proc_fork,
           proc_info,proc_session
        P: file_link_any,file_read,file_write,net_access,proc_exec,proc_fork,
           proc_info,proc_session
        L: cpc_cpu,dtrace_kernel,dtrace_proc,dtrace_user,...,win_upgrade_sl
```

▼ 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 fail for this debugging 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 9–36 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 9–37 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:

```
jdoe% ls -l useful.script
-rw-r--r--  1 aloe  staff  2303 Apr 10 10:10 useful.script
```

```
jdoe% chown objadmin useful.script
chown: useful.script: Not owner
jdoe% ppriv -eD chown objadmin useful.script
chown[11444]: missing privilege "file_chown"
            (euid = 130, syscall = 16) needed at zfs_zaccess+0x258
chown: useful.script: Not owner
```

When jdoe assumes the objadmin role, the permissions on the file are changed:

```
jdoe% 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 objadmin.script
-rw-r--r--  1 objadmin  admin  2303 Apr 10 10:11 useful.script
```

Example 9–38 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 208. 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 Run a Shell Script With Privileged Commands

Note – When you create a shell script that runs commands that require privilege, the appropriate rights profile must contain the commands with privileges assigned to them.

Before You Begin You must be in the root role.

1 **Start the script with `/bin/pfsh`, or any other profile shell, on the first line.**

```
#!/bin/pfsh
# Copyright (c) 2011 by Oracle
```

2 **Determine the privileges that the commands in the script need.**

```
% ppriv -eD script-full-path
```

3 **Become an administrator with the required security attributes.**

For more information, see "How to Obtain Administrative Rights" on page 160.

4 **Create or modify a rights profile for the script.**

You need to add the shell script, and the commands in the shell script, with their required security attributes to the rights profile. For the steps, see "How to Create or Change a Rights Profile" on page 170.

5 **Add the rights profile to a role and assign the role to a user.**

To run the script, the user assumes the role and runs the script in the role's profile shell.

- To add the rights profile to a role, see "How to Change the Security Attributes of a Role" on page 178.
- To assign the role to a user, see Example 9–20.

CHAPTER 10

Security Attributes in Oracle Solaris (Reference)

This chapter provides reference material about RBAC and privileges. The following is a list of the reference information in this chapter:

- "Rights Profiles" on page 197
- "Order of Search for Assigned Security Attributes" on page 199
- "Authorizations" on page 200
- "RBAC Databases" on page 201
- "RBAC Commands" on page 204
- "Administrative Commands for Handling Privileges" on page 206
- "Files With Privilege Information" on page 207
- "Privileges and Auditing" on page 207
- "Prevention of Privilege Escalation" on page 208
- "Legacy Applications and the Privilege Model" on page 209

For information about using RBAC, see Chapter 9, "Using Role-Based Access Control (Tasks)." For overview information, see "Role-Based Access Control (Overview)" on page 133.

To use privileges, see "Using Privileges (Tasks)" on page 186. For overview information, see "Privileges (Overview)" on page 145.

Rights Profiles

This section describes some typical rights profiles. Rights profiles are convenient collections of authorizations and other security attributes, commands with security attributes, and supplementary rights profiles. Oracle Solaris provides many rights profiles. If they are not sufficient for your needs, you can modify existing ones and create new ones.

Rights profiles must be assigned in order, from most to least powerful. For more information, see "Order of Search for Assigned Security Attributes" on page 199.

- **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. Note that the All rights profile is assigned at the end of the list of supplementary rights profiles. The `profiles` command displays the contents of the profile.

  ```
  % profiles -p "System Administrator" info
  ```
- **Operator rights profile** – Provides limited capabilities to manage files and offline media. This profile includes supplementary rights profiles to create a simple role. The `profiles` command displays the contents of the profile.

  ```
  % profiles -p Operator info
  ```
- **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. The `profiles` command displays the contents of the profile.

  ```
  % profiles -p "Printer Management" info
  ```
- **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. 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 or assign the Stop rights profile. The `profiles` command displays the contents of the profile.

  ```
  % profiles -p "Basic Solaris User" info
  ```
- **Console User rights profile** – For the workstation owner, provides access to authorizations, commands, and actions for the person who is seated at the computer. The `profiles` command displays the contents of the profile.

  ```
  % profiles -p "Console User" info
  ```
- **All rights profile** – For roles, provides access to commands that do not have security attributes. This profile can be appropriate for users with limited rights. The `profiles` command displays the contents of the profile.

  ```
  % profiles -p All info
  ```
- **Stop rights profile** – Is a special rights profile that stops the evaluation of further profiles. This profile prevents the evaluation of the `AUTHS_GRANTED`, `PROFS_GRANTED`, and `CONSOLE_USER` variables in the `policy.conf` file. With this profile, you can provide roles and users with a restricted profile shell.

  Note – The Stop profile affects privilege assignment indirectly. Rights profiles that are listed after the Stop profile are not evaluated. Therefore, the commands with privileges in those profiles are not in effect. To use this profile, see "How to Restrict an Administrator to Explicitly Assigned Rights" on page 182.

  The `profiles` command displays the contents of the profile.

```
% profiles -p Stop info
```

Each rights profile has an associated help file. The help files are in HTML and are customizable. The files reside in the `/usr/lib/help/profiles/locale/C` directory.

Viewing the Contents of Rights Profiles

You have three views into the contents of rights profiles.

- The `getent` command enables you to view the contents of all of the rights profiles on the system. For sample output, see "How to View All Defined Security Attributes" on page 156.
- The `profiles -p` "*Profile Name*" `info` command enables you to view the contents of a specific rights profile.
- The `profiles -l` *account-name* command enables you to view the contents of the rights profiles that are assigned to a specific user or role.

For more information, see the `getent`(1M) and `profiles`(1) man pages.

Order of Search for Assigned Security Attributes

A user or role can be assigned security attributes directly or through a rights profile. The order of search affects which security attribute value is used. The value of the first found instance of the attribute is used.

Note – The order of authorizations is not important. Authorizations are cumulative.

When a user logs in, security attributes are assigned in the following search order:

- **security attributes** that are assigned to the user with the `useradd` and `usermod` commands. For a list, see "`user_attr` Database" on page 202.
- **rights profiles** that are assigned to the user with the `useradd` and `usermod` commands. These assignments are searched in order.

  The order is first profile in the list, then its list of rights profiles, second profile in the list, then its list of profiles, and so on. The first instance of a value is the one that the system uses, except for `auths` values, which are cumulative. The attributes in rights profiles include all the security attributes for users, plus supplementary profiles. For a list, see "`user_attr` Database" on page 202.
- **Console User rights profile** value. For a description, see "Rights Profiles" on page 197.
- If the **Stop rights profile** is assigned, the evaluation of security attributes stops. No attributes are assigned after the Stop profile is assigned. The Stop profile is evaluated after the Console User rights profile and before the other security attributes in the `policy.conf` file, including `AUTHS_GRANTED`. For a description, see "Rights Profiles" on page 197.

- **Basic Solaris User rights profile** value in the `policy.conf` file.
- **AUTHS_GRANTED** value in the `policy.conf` file.
- **PROFS_GRANTED** value in the `policy.conf` file.
- **PRIV_DEFAULT** value in the `policy.conf` file.
- **PRIV_LIMIT** value in the `policy.conf` file.

Authorizations

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.

Authorizations are user-level, therefore extensible. You can write a program that requires authorization, add the authorizations to your system, create a rights profile for these authorizations, and assign the rights profile to users or roles who are allowed to use the program.

Authorization Naming Conventions

An authorization has a name that is used internally. For example, `solaris.system.date` is the name of an authorization. An authorization has a short description, which appears in the graphical user interfaces (GUIs). For example, `Set Date & Time` is the description of the `solaris.system.date` 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 Oracle Solaris, 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 Network Link Security role would be limited to the `solaris.network.link.security` authorization, while the Network Security role has the Network Link Security rights profile as a supplementary profile, plus the `solaris.network.*` and `solaris.smf.manage.ssh` authorizations.

Delegation Authority in Authorizations

An authorization that ends with the suffix delegate enables a user or a role to delegate to other users any assigned authorizations that begin with the same prefix.

The solaris auth.delegate authorization enables a user or a role to delegate to other users any authorizations that these users or roles are assigned.

For example, a role with the solaris auth.delegate and solaris.network.wifi.wep authorizations can delegate the solaris.network.wifi.wep authorization to another user or role. Similarly, a role with the solaris auth.delegate and solaris.network.wifi.wep authorizations can delegate the solaris.network.wifi.wep authorization to another user or role.

RBAC Databases

The following databases store the data for the RBAC elements:

- **Extended user attributes database** (user_attr) – Associates users and roles with authorizations, privileges, keywords, and rights profiles
- **Rights profile attributes database** (prof_attr) – Defines rights profiles, lists the profiles' assigned authorizations, privileges, and keywords, 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 203.

RBAC Databases and the Naming Services

The name service scope of the RBAC databases is defined in the SMF service for the naming service switch, svc:/system/name-service/switch. The properties in this service for the RBAC databases are auth_attr, password, and prof_attr. The password property sets the naming service precedence for the passwd and user_attr databases. The prof_attr property sets the naming service precedence for the prof_attr and exec_attr databases.

In the following output. the auth_attr, password, and prof_attr entries are not listed. Therefore, the RBAC databases are using the files naming service.

```
# svccfg -s name-service/switch listprop config
config                        application
config/value_authorization    astring       solaris.smf.value.name-service.switch
```

```
config/default                astring       files
config/host                   astring       "files ldap dns"
config/printer                astring       "user files ldap"
```

user_attr Database

The user_attr database contains user and role information that supplements the passwd and shadow databases.

The following security attributes can be set by using the roleadd, rolemod, useradd, usermod, and profiles commands:

- For a user, the roles keyword assigns one or more defined roles.
- For a role, the user value to the roleauth keyword enables the role to authenticate with the user password rather than with the role password. By default, the value is role.
- For a user or role, the following attributes can be set:
  - audit_flags keyword - Modifies the audit mask. For reference, see the audit_flags(5) man page.
  - auths keyword - Assigns authorizations. For reference, see the auths(1) man page.
  - defaultpriv keyword - Adds privileges or removes them from the default basic set of privileges. For reference, see "How Privileges Are Implemented" on page 149.
  - limitpriv keyword - Adds privileges or removes them from the default limit set of privileges. For reference, see "How Privileges Are Implemented" on page 149.

    These privileges are always in effect, they are not attributes of a command. For reference, see the privileges(5) man page and "How Privileges Are Implemented" on page 149.
  - projects keyword - Adds a default project. For reference, see the project(4) man page.
  - lock_after_retries keyword - If the value is yes, the system is locked after the number of retries exceeds the number that is allowed in the /etc/default/login file.
  - profiles keyword - Assigns rights profiles.

For more information, see the user_attr(4) man page. To view the contents of this database, use the getent user_attr command. For more information, see the getent(1M) man page and "How to View All Defined Security Attributes" on page 156.

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.

To view the contents of this database, use the getent prof_attr command. For more information, see the getent(1M) man page and "How to View All Defined Security Attributes" on page 156.

prof_attr Database

The prof_attr database stores the name, description, help file location, privileges, 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 203.

For more information, see the prof_attr(4) man page. To view the contents of this database, use the getent exec_attr command. For more information, see the getent(1M) man page and "How to View All Defined Security Attributes" on page 156.

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 or users to whom the profile is assigned.

For more information, see the exec_attr(4) man page. To view the contents of this database, use the getent command. For more information, see the getent(1M) man page and "How to View All Defined Security Attributes" on page 156.

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. To view the profile contents, see "Rights Profiles" on page 197.
- 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:

```
# grep AUTHS /etc/security/policy
AUTHS_GRANTED=solaris.device.cdrw
```

```
# grep PROFS /etc/security/policy
PROFS_GRANTED=Basic Solaris User

# grep PRIV /etc/security/policy

#PRIV_DEFAULT=basic
#PRIV_LIMIT=all
```

For more information about privileges, see “Privileges (Overview)” on page 145.

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

The following commands retrieve and set RBAC information.

TABLE 10–1 RBAC Administration Commands

Man Page for Command	Description
auths(1)	Displays authorizations for a user.
getent(1M)	Interface to list the contents of the user_attr, prof_attr, and exec_attr databases.
nscd(1M)	Name service cache daemon, useful for caching the user_attr, prof_attr, and exec_attr databases. Use the svcadm command to restart the daemon.
pam_roles(5)	Role account management module for PAM. Checks for the authorization to assume role.
pfexec(1)	Used by profile shells to execute commands with security attributes that are specified in the exec_attr database.
policy.conf(4)	Configuration file for system security policy. Lists granted authorizations, granted privileges, and other security information.
profiles(1)	Displays rights profiles for a specified user. Creates or modifies a rights profile on a local system or an LDAP network.
roles(1)	Displays roles that a specified user can assume.
roleadd(1M)	Adds a role to a local system or to an LDAP network.
roleadd(1M)	Adds a role to a local system or to an LDAP network.
rolemod(1M)	Modifies a role's properties on a local system or on an LDAP network.
userattr(1)	Displays the value of a specific right that is assigned to a user or role account.

TABLE 10–1 RBAC Administration Commands *(Continued)*

Man Page for Command	Description
useradd(1M)	Adds a user account to the system or to an LDAP network. The -R option assigns a role to a user's account.
userdel(1M)	Deletes a user's login from the system or from an LDAP network.
usermod(1M)	Modifies a user's account properties on the system.

Selected Commands That Require Authorizations

The following table provides examples of how authorizations are used to limit command options on an Oracle Solaris system. For more discussion of authorizations, see "Authorizations" on page 200.

TABLE 10–2 Commands and Associated Authorizations

Man Page for Command	Authorization Requirements
at(1)	solaris.jobs.user required for all options (when neither at.allow nor at.deny files exist)
atq(1)	solaris.jobs.admin required for all options
cdrw(1)	solaris.device.cdrw required for all options, and is granted by default in the policy.conf file
crontab(1)	solaris.jobs.user required for the option to submit a job (when neither crontab.allow nor crontab.deny files exist)
	solaris.jobs.admin required for the options to list or modify other users' crontab files
allocate(1)	solaris.device.allocate (or other authorization as specified in device_allocate file) required to allocate a device
	solaris.device.revoke (or other authorization as specified in device_allocate file) required to allocate a device to another user (-F option)
deallocate(1)	solaris.device.allocate (or other authorization as specified in device_allocate file) required to deallocate another user's device
	solaris.device.revoke (or other authorization as specified in device_allocate) required to force deallocation of the specified device (-F option) or all devices (-I option)
list_devices(1)	solaris.device.revoke required to list another user's devices (-U option)
roleadd(1M)	solaris.user.manage required to create a role. solaris.account.activate required to set the initial password. solaris.account.setpolicy required to set password policy, such as account locking and password aging.

TABLE 10–2 Commands and Associated Authorizations *(Continued)*

Man Page for Command	Authorization Requirements
`roledel`(1M)	`solaris.passwd.assign` authorization required to delete the password.
`rolemod`(1M)	`solaris.passwd.assign` authorization required to change the password. `solaris.account.setpolicy` required to change password policy, such as account locking and password aging.
`sendmail`(1M)	`solaris.mail` required to access mail subsystem functions; `solaris.mail.mailq` required to view mail queue
`useradd`(1M)	`solaris.user.manage` required to create a user. `solaris.account.activate` required to set the initial password. `solaris.account.setpolicy` required to set password policy, such as account locking and password aging.
`userdel`(1M)	`solaris.passwd.assign` authorization required to delete the password.
`usermod`(1M)	`solaris.passwd.assign` authorization required to change the password. `solaris.account.setpolicy` required to change password policy, such as account locking and password aging.

Privileges

Privileges restrict processes are implemented in the kernel, and can restrict processes at the command, user, role, or system level.

Administrative Commands for Handling Privileges

The following table lists the commands that are available to handle privileges.

TABLE 10–3 Commands for Handling Privilege

Purpose	Command	Man Page
Examine process privileges	`ppriv -v` *pid*	`ppriv`(1)
Set process privileges	`ppriv -s` *spec*	
List the privileges on the system	`ppriv -l`	
List a privilege and its description	`ppriv -lv` *priv*	
Debug privilege failure	`ppriv -eD` *failed-operation*	
Assign privileges to a new user	`useradd`	`useradd`(1M)
Add privileges to an existing user	`usermod`	`usermod`(1M)
Assign privileges to a rights profile	`profiles`	`profiles`(1)

TABLE 10–3 Commands for Handling Privilege *(Continued)*

Purpose	Command	Man Page
Assign privileges to a new role	`roleadd`	`roleadd`(1M)
Add privileges to an existing role	`rolemod`	`rolemod`(1M)
View device policy	`getdevpolicy`	`getdevpolicy`(1M)
Set device policy	`devfsadm`	`devfsadm`(1M)
Update device policy on open devices	`update_drv -p` *policy driver*	`update_drv`(1M)
Add device policy to a device	`add_drv -p` *policy driver*	`add_drv`(1M)

Files With Privilege Information

The following files contain information about privileges.

TABLE 10–4 Files That Contain Privilege Information

File and Man Page	Privilege Information	Description
`/etc/security/policy.conf`	`PRIV_DEFAULT`	Inheritable set of privileges for the system
`policy.conf`(4)	`PRIV_LIMIT`	Limit set of privileges for the system
`syslog.conf`	System log file for debug messages	Privilege debugging log
`syslog.conf`(4)	Path set in `priv.debug` entry	

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_MODALLOCPRIV` **audit event** – The audit event generates an audit record when a privilege is added from outside the kernel. The `AUE_MODALLOCPRIV` 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_PFEXEC` **audit event** – The audit event generates an audit record when a call is made to `execve()` with `pfexec()` enabled. The `AUE_PFEXEC` audit event is in the `as`,`ex`, `ps`, 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 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, vulnerable system modifications require the full set of privileges. For example, a file or process that is owned by root (UID=0) can only be changed by a process with the full set of privileges. The root account 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 about 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 152.

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.

PART IV

Cryptographic Services

This section describes the centralized cryptographic and public key technology features that Oracle Solaris provides.

- Chapter 11, “Cryptographic Framework (Overview)”
- Chapter 12, “Cryptographic Framework (Tasks)”
- Chapter 13, “Key Management Framework”

CHAPTER 11

Cryptographic Framework (Overview)

This chapter describes the Cryptographic Framework feature of Oracle Solaris. The following is a list of the information in this chapter.

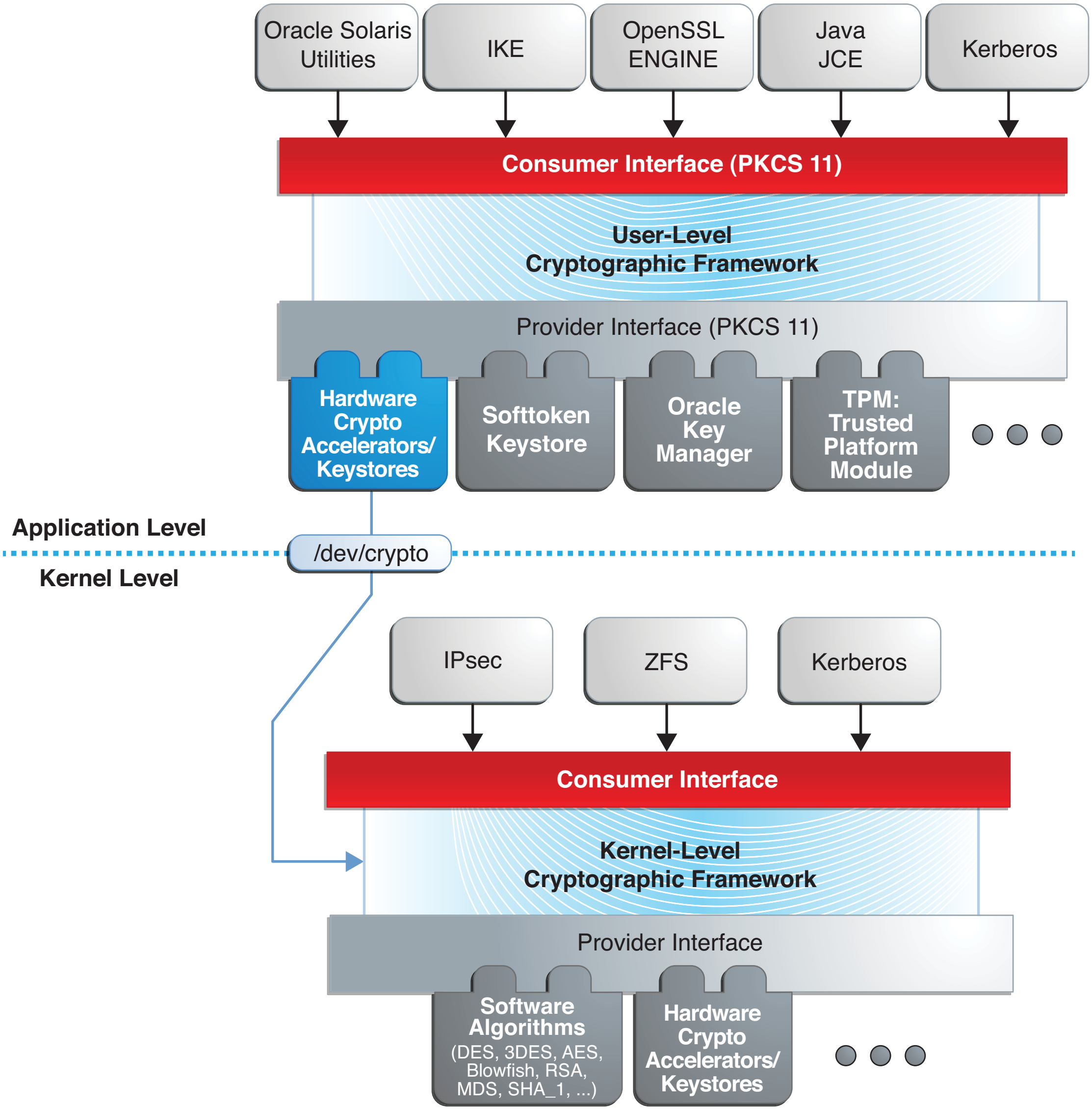
- "Introduction to the Cryptographic Framework" on page 213
- "Terminology in the Cryptographic Framework" on page 215
- "Scope of the Cryptographic Framework" on page 217
- "Administrative Commands in the Cryptographic Framework" on page 217
- "User-Level Commands in the Cryptographic Framework" on page 217
- "Plugins to the Cryptographic Framework" on page 218
- "Cryptographic Services and Zones" on page 219

To administer and use the Cryptographic Framework, see Chapter 12, "Cryptographic Framework (Tasks)."

Introduction to the Cryptographic Framework

The 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).

FIGURE 11–1 Cryptographic Framework Levels


Oracle Solaris Utilities
IKE
OpenSSL ENGINE
Java JCE
Kerberos
Consumer Interface (PKCS 11)
User-Level Cryptographic Framework
Provider Interface (PKCS 11)
Hardware Crypto Accelerators/ Keystores
Softtoken Keystore
Oracle Key Manager
TPM: Trusted Platform Module
Application Level
/dev/crypto
Kernel Level
IPsec
ZFS
Kerberos
Consumer Interface
Kernel-Level Cryptographic Framework
Provider Interface
Software Algorithms
(DES, 3DES, AES, Blowfish, RSA, MDS, SHA_1, ...)
Hardware Crypto Accelerators/ Keystores

At the kernel level, the framework currently handles cryptographic requirements for Kerberos and IPsec. User-level consumers include `libsasl` and IKE. The kernel SSL (`kssl`) proxy uses the Cryptographic Framework. For more information, see "Web Servers Using the Secure Sockets Layer Protocol" in *Oracle Solaris Administration: Network Services* and the `ksslcfg`(1M) man page.

Export law in the United States requires that the use of open cryptographic interfaces be restricted. The 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 218.

The framework enables *providers* of cryptographic services to have their services used by many *consumers* in Oracle Solaris. 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 Oracle Solaris Cryptographic Framework," in *Developer's Guide to Oracle Solaris 11 Security*.

Terminology in the 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 a regular 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_softtoken.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 Accelerator 6000

- **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 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. Regular 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 Cryptographic Framework (Task Map)" on page 222.

Administrative Commands in the 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
- 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 Service Management Facility (SMF) feature of Oracle Solaris. `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 Cryptographic Framework

The 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 Cryptographic Framework. Typically, this command is run by the developer of a provider.

The `elfsign` command has subcommands to request a certificate, sign binaries, and verify the signature on a binary. Unsigned binaries cannot be used by the Cryptographic Framework. Providers that have verifiable signed binaries can use the framework.

Plugins to the Cryptographic Framework

Third parties can plug their providers into the 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 Oracle. The certificate request is based on a private key that the third party selects, and a certificate that Oracle provides. The certificate request is sent to Oracle, which registers the third party and then issues the certificate. The third party then signs its provider object with the certificate from Oracle.

The loadable kernel software modules and the kernel device drivers for hardware accelerators must also register with the kernel. Registration is through the Cryptographic Framework SPI (service provider interface).

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 about zones, see Part II, “Oracle Solaris Zones,” in *Oracle Solaris Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management*. For more information about SMF that manages persistent applications, see Chapter 6, “Managing Services (Overview),” in *Oracle Solaris Administration: Common Tasks* and the `smf(5)` man page.

CHAPTER 12

Cryptographic Framework (Tasks)

This chapter describes how to use the Cryptographic Framework. The following is a list of information in this chapter.

- "Using the Cryptographic Framework (Task Map)" on page 221
- "Protecting Files With the Cryptographic Framework (Tasks)" on page 221
- "Administering the Cryptographic Framework (Tasks)" on page 235

Using the Cryptographic Framework (Task Map)

The following task map points to the tasks for using the Cryptographic Framework.

Task	Description	For Instructions
Protect individual files or sets of files.	Ensures that file content has not been tampered with. Prevents files from being read by intruders. These procedures can be done by regular users.	"Protecting Files With the Cryptographic Framework (Task Map)" on page 222
Administer the framework.	Adds, configures, and removes software providers. Disables and enables hardware provider mechanisms. These procedures are administrative procedures.	"Administering the Cryptographic Framework (Task Map)" on page 235

Protecting Files With the Cryptographic Framework (Tasks)

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.

Protecting Files With the Cryptographic Framework (Task Map)

The 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.

Task	Description	For Instructions
Generate a symmetric key.	Generates a random key for use with algorithms that the user specifies.	"How to Generate a Symmetric Key by Using the `dd` Command" on page 222
	Generates a key of user-specified length. Optionally, stores the key in a file, a PKCS #11 keystore, or an NSS keystore.	"How to Generate a Symmetric Key by Using the `pktool` Command" on page 224
Provide a checksum that ensures the integrity of a file.	Verifies that the receiver's copy of a file is identical to the file that was sent.	"How to Compute a Digest of a File" on page 229
Protect a file with a message authentication code (MAC).	Verifies to the receiver of your message that you were the sender.	"How to Compute a MAC of a File" on page 230
Encrypt a file, and then decrypt the encrypted file.	Protects the content of files by encrypting the file. Provides the encryption parameters to decrypt the file.	"How to Encrypt and Decrypt a File" on page 232

▼ 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 224.
- 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:  Min   Max (bits)
------------------------------------------
aes                       128   128
arcfour                     8   128
des                        64    64
3des                      192   192

% mac -l
Algorithm       Keysize:  Min   Max (bits)
------------------------------------------
des_mac                    64    64
sha1_hmac                   8   512
md5_hmac                    8   512
sha256_hmac                 8   512
sha384_hmac                 8  1024
sha512_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 12–1 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.

```
% ls -al ~/keyf
drwx------   2 jdoe  staff        512 May 3 11:32 ./
```

```
% dd if=/dev/urandom of=$HOME/keyf/05.07.aes16 bs=16  count=1
% chmod 400 ~/keyf/05.07.aes16
```

Example 12–2 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.des8 bs=8  count=1
% chmod 400 ~/keyf/05.07.des8
```

Example 12–3 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/05.07.3des.24 bs=24 count=1
% chmod 400 ~/keyf/05.07.3des.24
```

Example 12–4 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 /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 222.

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 256.**
- **To create and initialize an NSS database, see Example 13–5.**

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 keystore=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 `keystore=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 12–5 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.

```
# 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 12–6 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.

```
% pktool genkey keystore=file outkey=64bit.file1 keytype=des print=y
        Key Value ="a3237b2c0a8ff9b3"
```

```
% od -x 64bit.file1
0000000 a323 7b2c 0a8f f9b3
```

Example 12–7 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 ="294979e512cb8e79370dabecadc3fcbb849e78d2d6bd2049"
# 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=ipsha21 keytype=generic keylen=256 print=y
   Key Value ="659c20f2d6c3f9570bcee93e96d95e2263aca4eeb3369f72c5c786af4177fe9e"
# pktool genkey keystore=file outkey=ipsha22 keytype=generic keylen=256 print=y
   Key Value ="b041975a0e1fce0503665c3966684d731fa3dbb12fcf87b0a837b2da5d82c810"
```

- 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 aes auth_alg sha256  \
   encrkey  294979e512cb8e79370dabecadc3fcbb849e78d2d6bd2049 \
   authkey  659c20f2d6c3f9570bcee93e96d95e2263aca4eeb3369f72c5c786af4177fe9e
##
## for inbound packets
add esp spi 0x19174215 \
   src 192.168.2.1 dst 192.168.1.1 \
   encr_alg aes auth_alg sha256  \
   encrkey 9678f80e33406c86e3d1686e50406bd0434819c20d09d204 \
   authkey b041975a0e1fce0503665c3966684d731fa3dbb12fcf87b0a837b2da5d82c810
```

- 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 ipsha21 ipsha22
```

- The administrator copies the `ipseckeys` 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 `ipseckeys` 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.**

```
% 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.

`% 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 12–8 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.

```
% 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:

```
% digest -a md5 email.attach >>  $HOME/digest.emails.05.07
% cat  ~/digest.emails.05.07
85c0a53d1a5cc71ea34d9ee7b1b28b01
```

Example 12–9 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) = 085d991238d61bd0cfa2946c183be8e32cccf6c9
sha1 (docs/legal4) = f3085eae7e2c8d008816564fdf28027d10e1d983
```

▼ 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 222. 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 224.

3. **Create a MAC for a file.**

   Provide a key and use a symmetric key algorithm with the `mac` command.

   `% 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 12–10 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.

```
% mac -v -a des_mac email.attach
Enter passphrase:      <Type passphrase>
des_mac (email.attach) = dd27870a
% echo "des_mac (email.attach) = dd27870a" >> ~/desmac.daily.05.07
```

Example 12–11 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.

```
% 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 12–12 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.

```
% mac -v -a sha1_hmac \
-k $HOME/keyf/05.07.mack64 docs/* > $HOME/mac.docs.legal.05.07
% more ~/mac.docs.legal.05.07
sha1_hmac (docs/legal1) = 9b31536d3b3c0c6b25d653418db8e765e17fe07a
sha1_hmac (docs/legal2) = 865af61a3002f8a457462a428cdb1a88c1b51ff5
sha1_hmac (docs/legal3) = 076c944cb2528536c9aebd3b9fbe367e07b61dc7
sha1_hmac (docs/legal4) = 7aede27602ef6e4454748cbd3821e0152e45beb4
```

Example 12–13 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
sha1_hmac (docs/legal1) = 9b31536d3b3c0c6b25d653418db8e765e17fe07a
sha1_hmac (docs/legal2) = 865af61a3002f8a457462a428cdb1a88c1b51ff5
sha1_hmac (docs/legal3) = 076c944cb2528536c9aebd3b9fbe367e07b61dc7
sha1_hmac (docs/legal4) = 7aede27602ef6e4454748cbd3821e0152e45beb4
```

▼ 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 222. 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 224.

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 12–14 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
```

Example 12–15 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>
```

Example 12–16 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
```

Example 12–17 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.

```
% decrypt -a arcfour -i ~/enc/e.personal.txt \
-k ~/keyf/05.07.rc4.8 -o ~/personal.txt
```

Example 12–18 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 (Tasks)

This section describes how to administer the software providers and the hardware providers in the 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.

Administering the Cryptographic Framework (Task Map)

The following task map points to procedures for administering software and hardware providers in the Cryptographic Framework.

Task	Description	For Instructions
List the providers in the Cryptographic Framework.	Lists the algorithms, libraries, and hardware devices that are available for use in the Cryptographic Framework.	"How to List Available Providers" on page 236
Add a software provider.	Adds a PKCS #11 library or a kernel module to the Cryptographic Framework. The provider must be signed.	"How to Add a Software Provider" on page 240
Prevent the use of a user-level mechanism.	Removes a software mechanism from use. The mechanism can be enabled again.	"How to Prevent the Use of a User-Level Mechanism" on page 241
Temporarily disable mechanisms from a kernel module.	Temporarily removes a mechanism from use. Usually used for testing.	"How to Prevent the Use of a Kernel Software Provider" on page 243

Task	Description	For Instructions
Uninstall a provider.	Removes a kernel software provider from use.	Example 12–27
List available hardware providers.	Shows the attached hardware, shows the mechanisms that the hardware provides, and shows which mechanisms are enabled for use.	"How to List Hardware Providers" on page 245
Disable mechanisms from a hardware provider.	Ensures that selected mechanisms on a hardware accelerator are not used.	"How to Disable Hardware Provider Mechanisms and Features" on page 246
Restart or refresh cryptographic services.	Ensures that cryptographic services are available.	"How to Refresh or Restart All Cryptographic Services" on page 248

▼ How to List Available Providers

The 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 Oracle 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 Oracle 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.

```
% cryptoadm list
User-level providers:
Provider: /usr/lib/security/$ISA/pkcs11_kernel.so
Provider: /usr/lib/security/$ISA/pkcs11_softtoken.so
Provider: /usr/lib/security/$ISA/pkcs11_tpm.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 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 12–20.

   The output is truncated for display purposes.

```
% 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
...
CKM_ECDSA_SHA1
CKM_ECDH1_DERIVE

Provider: /usr/lib/security/$ISA/pkcs11_tpm.so
/usr/lib/security/$ISA/pkcs11_tpm.so: no slots presented.

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_SHA_1,CKM_SHA_1_HMAC,CKM_SHA_1_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 12–19 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_DES_MAC_GENERAL
CKM_DES_MAC
...
CKM_ECDSA
CKM_ECDSA_SHA1
CKM_ECDH1_DERIVE
```

Example 12–20 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:
=====================
/usr/lib/security/$ISA/pkcs11_kernel.so: all mechanisms are enabled.
/usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled,
except CKM_MD5. random is enabled.
/usr/lib/security/$ISA/pkcs11_tpm.so: all mechanisms are enabled.

Kernel software providers:
==========================
des: all mechanisms are enabled.
aes: all mechanisms are enabled.
arcfour: all mechanisms are enabled.
blowfish: all mechanisms are enabled.
ecc: all mechanisms are enabled.
sha1: all mechanisms are enabled.
sha2: all mechanisms are enabled.
md4: all mechanisms are enabled.
md5: all mechanisms are enabled.
rsa: all mechanisms are enabled.
swrand: random is enabled.

Kernel hardware providers:
==========================
ncp/0: all mechanisms are enabled. random is enabled.
```

Example 12–21 Determining Which Cryptographic Mechanisms Perform Which Functions

Mechanisms perform specific cryptographic functions, such as signing or key generation. The `-v -m` options display every mechanism and its functions.

In this instance, the administrator wants to determine for which functions the CKM_ECDSA* mechanisms can be used.

```
% cryptoadm list -vm
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
...
CKM_ECDSA       112 571  .  .  .  .  X  .  X  .  .  .  .  .  .
CKM_ECDSA_SHA1  112 571  .  .  .  .  X  .  X  .  .  .  .  .  .
...
```

The listing indicates that these user-level mechanisms are available from the /usr/lib/security/$ISA/pkcs11_softtoken.so library.

Each item in an entry represents a piece of information about the mechanism. For these ECC mechanisms, the listing indicates the following:

- Minimum length – 112 bytes
- Maximum length – 571 bytes
- Hardware – Is not available on hardware.
- Encrypt – Is not used to encrypt data.
- Decrypt – Is not used to decrypt data.
- Digest – Is not used to create message digests.
- Sign – Is used to sign data.
- Sign + Recover – Is not used to sign data, where the data can be recovered from the signature.
- Verify – Is used to verify signed data.
- Verify + Recover– Is not used to verify data that can be recovered from the signature.
- Key generation – Is not used to generate a private key.
- Pair generation – Is not used to generate a key pair.
- Wrap – Is not used to wrap. that is, encrypt, an existing key.
- Unwrap – Is not used to unwrap a wrapped key.
- Derive – Is not used to derive a new key from a base key.

▼ How to Add a Software Provider

Before You Begin You must be assigned the Crypto Management rights profile.

1 **Become an administrator with the required security attributes.**

For more information, see "How to Obtain Administrative Rights" on page 160.

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
/usr/lib/security/$ISA/pkcs11_tpm.so: all mechanisms are enabled.

Kernel software providers:
    des
    aes
    arcfour
    blowfish
    sha1
    sha2
    md4
    md5
    rsa
    swrand

Kernel hardware providers:
    ncp/0
```

3 **Add the provider from a repository.**

Existing provider software has been issued a certificate by Oracle.

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.

```
# svcadm refresh svc:/system/cryptosvc
```

5 **Locate the new provider on the list.**

In this case, a new kernel software provider was installed.

```
# cryptoadm list
...
Kernel software providers:
    des
    aes
    arcfour
    blowfish
    ecc
    sha1
    sha2
    md4
    md5
    rsa
    swrand
  sha3      <-- added provider
...
```

Example 12–22 Adding a User-Level Software Provider

In the following example, a signed PKCS #11 library is installed.

```
# 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
    /usr/lib/security/$ISA/pkcs11_tpm.so
    /opt/lib/$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/lib/\$ISA/libpkcs11.so.1
```

▼ 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.

Before You Begin You must be assigned the Crypto Management rights profile.

1. **Become an administrator with the required security attributes.**

   For more information, see "How to Obtain Administrative Rights" on page 160.

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 12–23 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 12–24 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 12–25 Permanently Removing User-Level Software Provider Availability

In the following example, the `libpkcs11.so.1` library is removed.

```
$ cryptoadm uninstall provider=/opt/lib/\$ISA/libpkcs11.so.1
$ cryptoadm list
user-level providers:
    /usr/lib/security/$ISA/pkcs11_kernel.so
    /usr/lib/security/$ISA/pkcs11_softtoken.so
    /usr/lib/security/$ISA/pkcs11_tpm.so

kernel software providers:
...
```

▼ 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.

Before You Begin You must be assigned the Crypto Management rights profile.

1. **Become an administrator with the required security attributes.**

   For more information, see "How to Obtain Administrative Rights" on page 160.

2. **List the mechanisms that are offered by a particular kernel software provider.**

   ```
   $ cryptoadm list -m provider=aes
   aes: CKM_AES_ECB,CKM_AES_CBC,CKM_AES_CTR,CKM_AES_CCM,CKM_AES_GCM,CKM_AES_GMAC
   ```

3. **List the mechanisms that are available for use.**

   ```
   $ cryptoadm list -p provider=aes
   aes: all mechanisms are enabled.
   ```

4. **Disable the mechanism that should not be used.**

   ```
   $ cryptoadm disable provider=aes mechanism=CKM_AES_ECB
   ```

5. **List the mechanisms that are available for use.**

   ```
   $ cryptoadm list -p provider=aes
   aes: all mechanisms are enabled, except CKM_AES_ECB.
   ```

Example 12–26 Enabling a Kernel Software Provider Mechanism

In the following example, a disabled AES mechanism is again made available for use.

```
cryptoadm list -m provider=aes
aes: CKM_AES_ECB,CKM_AES_CBC,CKM_AES_CTR,CKM_AES_CCM,CKM_AES_GCM,CKM_AES_GMAC
$ 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 12–27 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 12–28 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 12–29 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,CKM_AES_CTR,CKM_AES_CCM,CKM_AES_GCM,CKM_AES_GMAC

$ 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 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 about getting your provider signed, see "Binary Signatures for Third-Party Software" on page 218.

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_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 12–30. The hardware in this example, the Sun Crypto Accelerator 1000 board, provides a random number generator.

Before You Begin You must be assigned the Crypto Management rights profile.

1. **Become an administrator with the required security attributes.**

   For more information, see "How to Obtain Administrative Rights" on page 160.

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 12–30 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 Cryptographic Framework is enabled. When the `kcfd` daemon fails for any reason, the Service Management Facility (SMF) 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 219.

Before You Begin You must be assigned the Crypto Management rights profile.

1. **Become an administrator with the required security attributes.**

   For more information, see "How to Obtain Administrative Rights" on page 160.

2. **Check the status of cryptographic services.**

   ```
   % svcs cryptosvc
    STATE         STIME    FMRI
   offline        Dec_09   svc:/system/cryptosvc:default
   ```

3. **Enable cryptographic services.**

   ```
   # svcadm enable svc:/system/cryptosvc
   ```

Example 12–31 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
```

CHAPTER 13

Key Management Framework

The Key Management Framework (KMF) feature of Oracle Solaris 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. KMF supports third-party plugins.

- "Managing Public Key Technologies" on page 249
- "Key Management Framework Utilities" on page 250
- "Using the Key Management Framework (Tasks)" on page 252

Managing Public Key Technologies

The Key Management Framework (KMF) provides a unified approach to managing public key technologies (PKI). Oracle Solaris 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.
- **kmfcfg 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.

- **KMF library –** This library contains programming interfaces that abstract the underlying keystore mechanism.

  Applications do not have to choose one particular keystore 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 keystore mechanisms can be added. Therefore, applications that use the new mechanisms would require only minor modifications to use a new keystore.

Note – To determine the version of OpenSSL that is running, type `openssl version`. The output is similar to the following:

```
OpenSSL 1.0.0d 8 Feb 2011
```

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 Cryptographic Framework
- NSS, that is, Network Security Services
- OpenSSL, a file-based keystore

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 keystore for the plugin. To remove the plugin from KMF, you remove the keystore.
- `uninstall` *plugin* – Removes the plugin from KMF by removing its keystore.
- `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 `kmfcfg(1)` man page. For the procedure, see "How to Manage Third-Party Plugins in KMF" on page 262.

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:

- Generate a self-signed certificate.
- Generate a certificate request.
- Generate a symmetric key.
- Generate a public/private key pair.
- Generate a PKCS #10 certificate signing request (CSR) to be sent to an external certificate authority (CA) to be signed.
- Sign a PKCS #10 CSR.
- 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 252.

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.

Using the Key Management Framework (Task Map)

The Key Management Framework (KMF) enables you to centrally manage public key technologies.

Task	Description	For Instructions
Create a certificate.	Creates a certificate for use by PKCS #11, NSS, or SSL.	"How to Create a Certificate by Using the `pktool gencert` Command" on page 253
Export a certificate.	Creates a file with the certificate and its supporting keys. The file can be protected with a password.	"How to Export a Certificate and Private Key in PKCS #12 Format" on page 255
Import a certificate.	Imports a certificate from another system.	"How to Import a Certificate Into Your Keystore" on page 254
	Imports a certificate in PKCS #12 format from another system.	Example 13–2
Generate a passphrase.	Generates a passphrase for access to a PKCS #11 keystore or an NSS keystore.	"How to Generate a Passphrase by Using the `pktool setpin` Command" on page 256
Generate a symmetric key.	Generates symmetric keys for use in encrypting files, creating a MAC of a file, and for applications.	"How to Generate a Symmetric Key by Using the `pktool` Command" on page 224
Generate a key pair.	Generates a public/private key pair for use with applications.	"How to Generate a Key Pair by Using the `pktool genkeypair` Command" on page 257
Generate a PKCS #10 CSR.	Generates a PKCS #10 certificate signing request (CSR) for an external certificate authority (CA) to sign.	`pktool(1)` man page
Sign a PKCS #10 CSR.	Signs a PKCS #10 CSR.	"How to Sign a Certificate Request by Using the `pktool signcsr` Command" on page 261
Add a plugin to KMF.	Installs, modifies, and lists a plugin. Also, removes the plugin from the KMF.	"How to Manage Third-Party Plugins in KMF" on page 262

▼ 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* Specifies a unique name that the issuer gives to the certificate.

`subject=`*subject-DN* Specifies the distinguished name for the certificate.

`serial=`*hex-serial-number* Specifies 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
      Subject: subject-DN
      Issuer:  distinguished-name
      Serial:  hex-serial-number
n. ...
```

This command lists all certificates in the keystore. In the following example, the keystore contains one certificate only.

Example 13–1 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
      ID: 12:82:17:5f:80:78:eb:44:8b:98:e3:3c:11:c0:32:5e:b6:4c:ea:eb
      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 13–4.

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 13–2 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
        ID: 12:82:17:5f:80:78:eb:44:8b:98:e3:3c:11:c0:32:5e:b6:4c:ea:eb
        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 13–3 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
        ID: 21:ae:83:98:24:d1:1f:cb:65:5b:48:75:7d:02:47:cf:98:1f:ec:a0
        Subject: C=US, O=TheirCompany, OU=Security, CN=TheirCompany Root CA
        Issuer: C=US, O=TheirCompany, OU=Security, CN=TheirCompany Root CA
        Serial: 0x01
```

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 13–4 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
      ID: 12:82:17:5f:80:78:eb:44:8b:98:e3:3c:11:c0:32:5e:b6:4c:ea:eb
      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 13–4.

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 13–5 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 Generate a Key Pair by Using the pktool genkeypair Command

Some applications require a public/private key pair. In this procedure, you create these key pairs and store them.

1 **(Optional) If you plan to use a keystore, create the keystore.**

- **To create and initialize a PKCS #11 keystore, see "How to Generate a Passphrase by Using the `pktool setpin` Command" on page 256.**
- **To create and initialize an NSS keystore, see Example 13–5.**

2 Create the key pair.

Use one of the following methods.

- **Create the key pair, and store the key pair in a file.**

  File-based keys are created for applications that read keys directly from files on the disk. Typically, applications that directly use OpenSSL cryptographic libraries require that your store the keys and certificates for the application in files.

  Note – The `file` keystore does not support elliptic curve (`ec`) keys and certificates.

  ```
  % pktool genkeypair keystore=file outkey=key-filename \
  [format=der|pem] [keytype=rsa|dsa] [keylen=key-size]
  ```

  `keystore=file`
  : The value `file` specifies the file type of storage location for the key.

  `outkey=`*key-filename*
  : Specifies the name of the file where the key pair is stored.

  `format=der|pem`
  : Specifies the encoding format of the key pair. `der` output is binary, and `pem` output is ASCII.

  `keytype=rsa|dsa`
  : Specifies the type of key pair that can be stored in a `file` keystore. For definitions, see DSA and RSA.

  `keylen=`*key-size*
  : Specifies the length of the key in bits. The number must be divisible by 8. To determine possible key sizes, use the `cryptoadm list -vm` command.

- **Create the key pair, and store it in a PKCS #11 keystore.**

  You must complete Step 1 before using this method.

  The PKCS #11 keystore is used to store objects on a hardware device. The device could be a Sun Crypto Accelerator 6000 card, a trusted platform module (TPM) device, or a smart card that is plugged into the Cryptographic Framework. PKCS #11 can also be used to store objects in the `softtoken`, or software-based token, which stores the objects in a private subdirectory on the disk. For more information, see the `pkcs11_softtoken(5)` man page.

  You can retrieve the key pair from the keystore by a label that you specify.

  ```
  % pktool genkeypair label=key-label \
  [token=token[:manuf[:serial]]] \
  [keytype=rsa|dsa|ec]  [curve=ECC-Curve-Name]]\
  [keylen=key-size] [listcurves]
  ```

`label=`*key-label*
Specifies a label for the key pair. The key pair can be retrieved from the keystore by its label.

`token=`*token*[:*manuf*[:*serial*]]
Specifies the token name. By default, the token name is `Sun Software PKCS#11 softtoken`.

`keytype=rsa|dsa|ec [curve=`*ECC-Curve-Name*`]`
Specifies the keypair type. For the elliptic curve (`ec`) type, optionally specifies a curve name. Curve names are listed as output to the `listcurves` option.

`keylen=`*key-size*
Specifies the length of the key in bits. The number must be divisible by 8.

`listcurves`
Lists the elliptic curve names that can be used as values to the `curve=` option for an `ec` key type.

- **Generate the key pair, and store it in an NSS keystore.**

The NSS keystore is used by servers that rely on NSS as their primary cryptographic interface. For example, the Oracle iPlanet Web Server uses the NSS databases for object storage.

You must complete Step 1 before using this method.

```
% pktool keystore=nss genkeypair label=key-nickname \
[token=token[:manuf[:serial]]] \
[dir=directory-path] [prefix=database-prefix] \
[keytype=rsa|dsa|ec] [curve=ECC-Curve-Name]] \
[keylen=key-size] [listcurves]
```

`keystore=nss`
The value `nss` specifies the NSS type of storage location for the key.

`label=`*nickname*
Specifies a label for the key pair. The key pair can be retrieved from the keystore by its label.

`token=`*token*[:*manuf*[:*serial*]]
Specifies the token name. By default, the token is `Sun Software PKCS#11 softtoken`.

`dir=`*directory*
Specifies the directory path to the NSS database. By default, *directory* is the current directory.

`prefix=`*database-prefix*
Specifies the prefix to the NSS database. The default is no prefix.

`keytype=rsa|dsa|ec [curve=`*ECC-Curve-Name*`]`
Specifies the keypair type. For the elliptic curve type, optionally specifies a curve name. Curve names are listed as output to the `listcurves` option.

`keylen=`*key-size*
Specifies the length of the key in bits. The number must be divisible by 8.

`listcurves`
Lists the elliptic curve names that can be used as values to the `curve=` option for an `ec` key type.

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-filename* file.**

  ```
  % pktool list keystore=file objtype=key infile=key-filename
  Found n keys.
  Key #1 - keytype:location (keylen)
  ```

- **Verify the key in the PKCS #11 keystore.**

  ```
  $ pktool list objtype=key
  Enter PIN for keystore:
  Found n keys.
  Key #1 - keytype:location (keylen)
  ```

- **Verify the key in the NSS keystore.**

  ```
  % pktool list keystore=nss dir=directory objtype=key
  ```

Example 13–6 Creating a Key Pair by Using the pktool Command

In the following example, a user creates a PKCS #11 keystore for the first time. After determining the key sizes for RSA key pairs, the user then generates a key pair for an application. Finally, the user verifies that the key pair is in the keystore. The user notes that the second instance of the RSA key pair can be stored on hardware. Because the user does not specify a `token` argument, the key pair is stored as a `Sun Software PKCS#11 softtoken`.

```
# pktool setpin
Create new passphrase:     Easily remembered, hard-to-detect password
Re-enter new passphrase:     Retype password
Passphrase changed.
% cryptoadm list -vm | grep PAIR
...
CKM_DSA_KEY_PAIR_GEN         512  1024 .  .  .
CKM_RSA_PKCS_KEY_PAIR_GEN    256  4096 .  .  .
...
CKM_RSA_PKCS_KEY_PAIR_GEN    512  2048 X  .  .
ecc: CKM_EC_KEY_PAIR_GEN,CKM_ECDH1_DERIVE,CKM_ECDSA,CKM_ECDSA_SHA1
% pktool genkeypair label=specialappkeypair keytype=rsa keylen=2048
Enter PIN for Sun Software PKCS#11 softtoken  :     Type password

% pktool list
```

```
Enter PIN for Sun Software PKCS#11 softtoken  :     Type password

Found 1 keys.
Key #1 - keypair:  specialappkeypair (2048 bits)
```

Example 13–7 Creating a Key Pair That Uses the Elliptic Curve Algorithm

In the following example, a user adds an elliptic curve (ec)key pair to the keystore, specifies a curve name, and verifies that the key pair is in the keystore.

```
% pktool genkeypair listcurves
secp112r1, secp112r2, secp128r1, secp128r2, secp160k1
.
.
.
c2pnb304w1, c2tnb359v1, c2pnb368w1, c2tnb431r1, prime192v2
prime192v3
% pktool genkeypair label=eckeypair keytype=ec curves=c2tnb431r1
% pktool list
Enter PIN for Sun Software PKCS#11 softtoken  :     Type password

Found 2 keys.
Key #1 - keypair:  specialappkeypair (2048 bits)
Key #2 - keypair:  eckeypair (c2tnb431r1)
```

▼ How to Sign a Certificate Request by Using the pktool signcsr Command

This procedure is used to sign a PKCS #10 certificate signing request (CSR). The CSR can be in PEM or DER format. The signing process issues an X.509 v3 certificate. To generate a PKCS #10 CSR, see the `pktool(1)` man page.

Before You Begin You are a certificate authority (CA), you have received a CSR, and it is stored in a file.

1. **Collect the following information for the required arguments to the `pktool signcsr` command:**

   `signkey` If you have stored the signer's key in a PKCS #11 keystore, `signkey` is the *label* that retrieves this private key.

   If you have stored the signer's key in an NSS keystore or a file keystore, `signkey` is the file name that holds this private key.

   `csr` Specifies the file name of the CSR.

   `serial` Specifies the serial number of the signed certificate.

   `outcer` Specifies the file name for the signed certificate.

   `issuer` Specifies your CA issuer name in distinguished name (DN) format.

For information about optional arguments to the signcsr subcommand, see the pktool(1) man page.

2 **Sign the request and issue the certificate.**

For example, the following command signs the certificate with the signer's key from the PKCS #11 repository:

```
# pktool signcsr signkey=CASigningKey \
csr=fromExampleCoCSR \
serial=0x12345678 \
outcert=ExampleCoCert2010 \
issuer="O=Oracle Corporation, \
     OU=Oracle Solaris Security Technology, L=Redwood City, ST=CA, C=US, \
     CN=rootsign Oracle"
```

The following command signs the certificate with the signer's key from a file:

```
# pktool signcsr signkey=CASigningKey \
csr=fromExampleCoCSR \
serial=0x12345678 \
outcert=ExampleCoCert2010 \
issuer="O=Oracle Corporation, \
     OU=Oracle Solaris Security Technology, L=Redwood City, ST=CA, C=US, \
     CN=rootsign Oracle"
```

3 **Send the certificate to the requester.**

You can use email, a web site, or other mechanism to deliver the certificate to the requester.

For example, you could use email to send the ExampleCoCert2010 file to the requester.

▼ 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 – Specifies a unique name for the keystore that you provide.

path-to-plugin – Specifies the full path to the shared library object for the KMF plugin.

option-string – Specifies 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 13–8 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
# 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.

PART V

Authentication Services and Secure Communication

This section discusses authentication services that can be configured on a non-networked system, or between two systems.

- Chapter 14, "Network Services Authentication (Tasks)"
- Chapter 15, "Using PAM"
- Chapter 16, "Using SASL"
- Chapter 17, "Using Secure Shell (Tasks)"
- Chapter 18, "Secure Shell (Reference)"

To configure a network of authenticated users and systems, see Part VI, "Kerberos Service."

CHAPTER 14

Network Services Authentication (Tasks)

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 267
- "Administering Authentication With Secure RPC (Tasks)" on page 272

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 NIS naming service.

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 *Oracle Solaris Administration: Network Services*.
- For an outline of the transactions that are involved in RPC authentication, see "Implementation of Diffie-Hellman Authentication" on page 269.

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.

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 19, "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 *Oracle Solaris Administration: Network Services*.

The public keys and private keys are stored in an NIS database. NIS stores the keys in the `publickey` map. This file contains the public key and the private key for all potential users.

The system administrator is responsible for setting up NIS maps 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 Authentication With Secure RPC (Tasks)

By requiring authentication for use of mounted NFS file systems, you increase the security of your network.

Administering Secure RPC (Task Map)

The following task map points to procedures that configure Secure RPC for NIS, and NFS.

Task	Description	For Instructions
1. Start the keyserver.	Ensures that keys can be created so that users can be authenticated.	"How to Restart the Secure RPC Keyserver" on page 272
2. Set up credentials on an NIS host.	Ensures that the root user on a host can be authenticated in an NIS environment.	"How to Set Up a Diffie-Hellman Key for an NIS Host" on page 272
3. Give an NIS user a key.	Enables a user to be authenticated in an NIS environment.	"How to Set Up a Diffie-Hellman Key for an NIS User" on page 273
4. Share NFS files with authentication.	Enables an NFS server to securely protect shared file systems using authentication.	"How to Share NFS Files With Diffie-Hellman Authentication" on page 274

▼ How to Restart the Secure RPC Keyserver

Before You Begin You must be in the root role.

1. **Verify that the keyserv daemon is running.**

```
# svcs \*keyserv\*
STATE    STIME   FMRI
disabled Dec_14  svc:/network/rpc/keyserv
```

2. **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.

Before You Begin You must be in the root role.

1 **If the default naming service is not NIS, add the `publickey` map to the naming service.**

a. **Verify that the value of `config/default` for the naming service is not `nis`.**

```
# svccfg -s name-service/switch listprop config
config                       application
config/value_authorization   astring       solaris.smf.value.name-service.switch
config/default               astring       files
config/host                  astring       "files nis dns"
config/printer               astring       "user files nis"
```

If the value of `config/default` is `nis`, you can stop here.

b. **Set the naming service for `publickey` to `nis`.**

```
# svccfg
# svccfg -s name-service/switch setprop config/publickey = astring: "nis"
# svccfg -s name-service/switch:default refresh
```

c. **Confirm the `publickey` value.**

```
# svccfg
# svccfg -s name-service/switch listprop
config                       application
config/value_authorization   astring       solaris.smf.value.name-service.switch
config/default               astring       files
config/host                  astring       "files nis dns"
config/printer               astring       "user files nis"
config/publickey             astring       nis
```

On this system, the value of `publickey` is listed because it differs from the default, `files`.

2 **Create a new key pair by using the `newkey` command.**

```
# newkey -h hostname
```

where *hostname* is the name of the client.

Example 14–1 Setting Up a New Key for root on an NIS Client

In the following example, `earth` is set up as a secure NIS client. The administrator is assigned the Name Service Security rights profile.

```
# newkey -h earth
Adding new key for unix.earth@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.
#
```

▼ 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. The administrators must be assigned the Name Service Security rights profile.

1. **Become an administrator with the required security attributes.**

   For more information, see "How to Obtain Administrative Rights" on page 160.

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 14–2 Setting Up and Encrypting a New User Key in NIS

In this example, superuser sets up the key.

```
# 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 user `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 centralexample...
```

▼ 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, complete "How to Set Up a Diffie-Hellman Key for an NIS Host" on page 272.

You must be assigned the System Management rights profile to perform this task.

1. **Become an administrator with the required security attributes.**

   For more information, see "How to Obtain Administrative Rights" on page 160.

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
   ```

   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.

◆ ◆ ◆ CHAPTER 15

Using PAM

This chapter covers the Pluggable Authentication Module (PAM) framework. PAM provides a method to "plug in" authentication services into the Oracle Solaris OS. PAM provides support for multiple authentication services when accessing a system.

- "PAM (Overview)" on page 277
- "PAM (Tasks)" on page 279
- "PAM Configuration (Reference)" on page 282

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 retyping 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

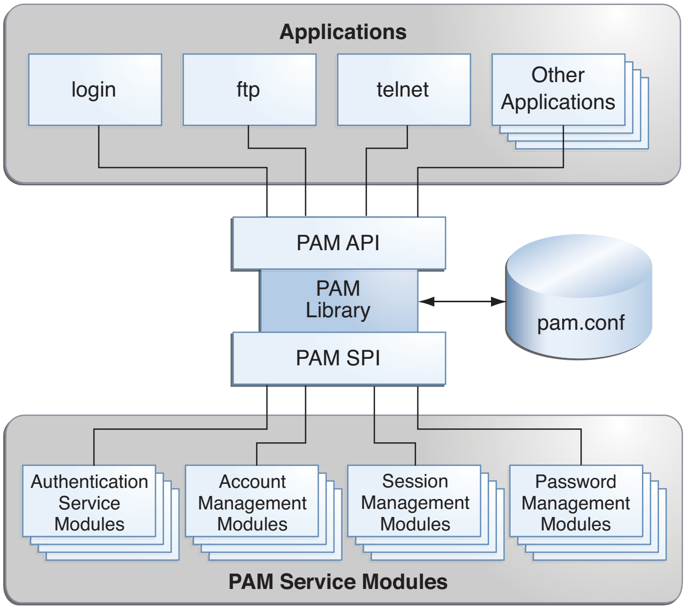
The PAM framework consists of four parts:

- 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.

FIGURE 15–1 PAM Architecture


Applications
login
ftp
telnet
Other Applications
PAM API
PAM Library
pam.conf
PAM SPI
Authentication Service Modules
Account Management Modules
Session Management Modules
Password Management Modules
PAM Service Modules

Changes to PAM for This Release

The PAM framework for the Oracle Solaris 11 Express 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 280.

PAM (Task Map)

Task	Description	For Instructions
Plan for your PAM installation.	Consider configuration issues and make decisions about them before you start the software configuration process.	"Planning for Your PAM Implementation" on page 280
Add new PAM modules.	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.	"How to Add a PAM Module" on page 281
Block access through `~/.rhosts`.	Further increase security by preventing access through `~/.rhosts`.	"How to Prevent Rhost-Style Access From Remote Systems With PAM" on page 282
Initiate error logging.	Start the logging of PAM error messages through `syslog`.	"How to Log PAM Error Reports" on page 282

Planning for Your PAM Implementation

As delivered, the `pam.conf` configuration file implements the standard 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 283 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 `sulogin` 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 an administrator.**

   For more information, see "How to Obtain Administrative Rights" on page 160.

2. **Determine which control flags and which other options should be used.**

   Refer to "How PAM Stacking Works" on page 283 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 an administrator.**

   For more information, see "How to Obtain Administrative Rights" on page 160.

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 an administrator.**

   For more information, see "How to Obtain Administrative Rights" on page 160.

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 283.

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 Oracle 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 283 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.

FIGURE 15–2 PAM Stacking: Effect of Control Flags

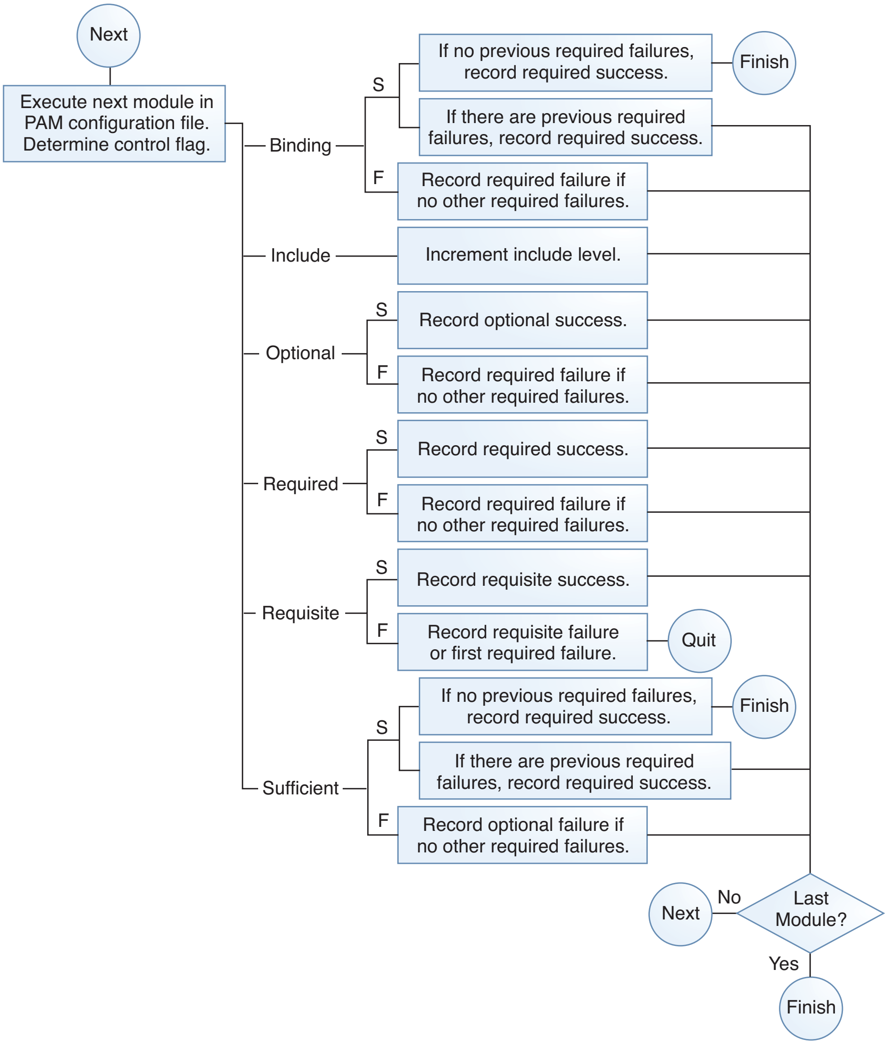
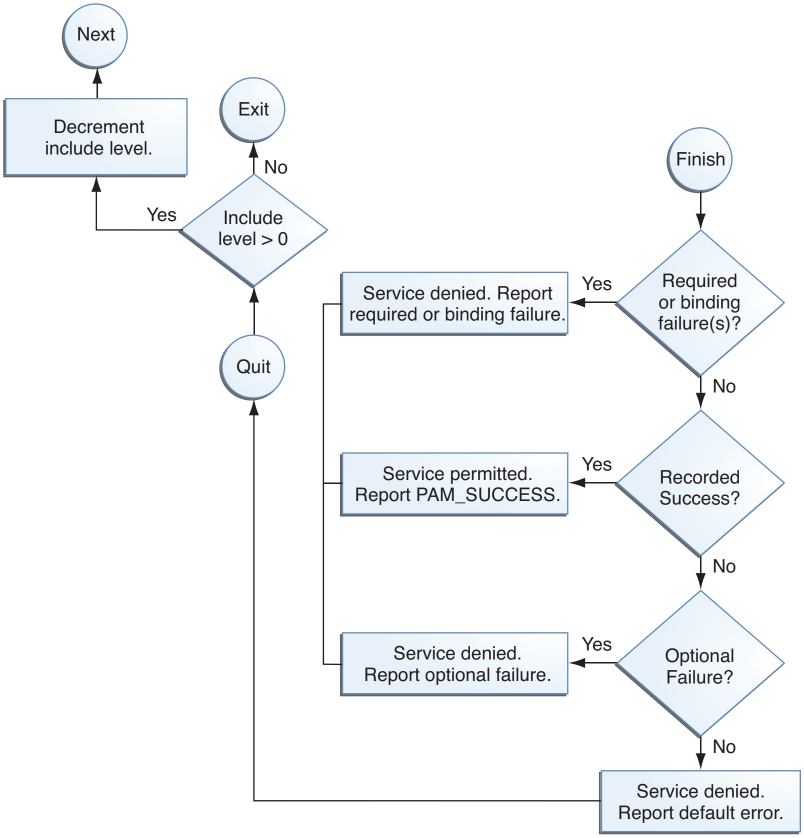

Next
Execute next module in PAM configuration file. Determine control flag.
Binding
S
If no previous required failures, record required success.
Finish
If there are previous required failures, record required success.
F
Record required failure if no other required failures.
Include
Increment include level.
Optional
S
Record optional success.
F
Record required failure if no other required failures.
Required
S
Record required success.
F
Record required failure if no other required failures.
Requisite
S
Record requisite success.
F
Record requisite failure or first required failure.
Quit
Sufficient
S
If no previous required failures, record required success.
Finish
If there are previous required failures, record required success.
F
Record optional failure if no other required failures.
Next
No
Last Module?
Yes
Finish

FIGURE 15–3 PAM Stacking: How Integrated Value Is Determined


Next
Decrement include level.
Exit
No
Yes
Include level > 0
Finish
Service denied. Report required or binding failure.
Yes
Required or binding failure(s)?
No
Quit
Service permitted. Report PAM_SUCCESS.
Yes
Recorded Success?
No
Service denied. Report optional failure.
Yes
Optional Failure?
No
Service denied. Report default error.

PAM Stacking Example

Consider the following example of an rlogin service that requests authentication.

EXAMPLE 15–1 Partial Contents of a Typical PAM Configuration File

The pam.conf file in this example has the following contents for rlogin services:

```
# Authentication management
...
# rlogin service
rlogin  auth sufficient         pam_rhosts_auth.so.1
rlogin  auth requisite          pam_authtok_get.so.1
rlogin  auth required           pam_dhkeys.so.1
rlogin  auth required           pam_unix_auth.so.1
...
```

EXAMPLE 15–1 Partial Contents of a Typical PAM Configuration File *(Continued)*

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.

CHAPTER 16

Using SASL

This chapter includes information about the Simple Authentication and Security Layer (SASL).

- "SASL (Overview)" on page 289
- "SASL (Reference)" on page 290

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 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

SASL (Reference)

The following section provides information about the implementation of SASL.

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:

`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 Oracle Solaris 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 `salauthd` and `pwcheck` daemons are not provided in the Oracle 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:

`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:

`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 Oracle Solaris 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 Oracle Solaris 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.

CHAPTER 17

Using Secure Shell (Tasks)

The Secure Shell feature of Oracle Solaris provides secure access to 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.

- "Secure Shell (Overview)" on page 293
- "Secure Shell and the OpenSSH Project" on page 295
- "Secure Shell and FIPS-140 Support" on page 296
- "Secure Shell (Task Map)" on page 297

For reference information, see Chapter 18, "Secure Shell (Reference)."

Secure Shell (Overview)

In Secure Shell, authentication is provided by the use of passwords, public keys, or both. All network traffic is encrypted. Thus, Secure Shell prevents a would-be intruder from being able to read an intercepted communication. Secure Shell also prevents an adversary from spoofing the system.

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 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.

On the server side, Secure Shell supports two versions of the Secure Shell protocol, Version 1 (v1) and Version 2. Version 2 (v2) is more secure. Secure Shell provides v1 only to assist users who are migrating to v2. For information about v1, see *System Administration Guide: Security Services*.

Secure Shell Authentication

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 about GSS-API, see "Introduction to GSS-API" in *Developer's Guide to Oracle Solaris 11 Security*.
- **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 Secure Shell authentication methods, the compatible protocol versions, and the host requirements.

TABLE 17–1 Authentication Methods for Secure Shell

Authentication Method	Local Host (Client) Requirements	Remote Host (Server) Requirements
GSS-API	Initiator credentials for the GSS mechanism.	Acceptor credentials for the GSS mechanism. For more information, see "Acquiring GSS Credentials in Secure Shell" on page 312.

Host-based	User account
	Local host private key in `/etc/ssh/ssh_host_rsa_key` or `/etc/ssh/ssh_host_dsa_key`
	`HostbasedAuthentication yes` in `/etc/ssh/ssh_config`
		User account
		Local host public key in `/etc/ssh/known_hosts` or `~/.ssh/known_hosts`
		`HostbasedAuthentication yes` in `/etc/ssh/sshd_config`
		`IgnoreRhosts no` in `/etc/ssh/sshd_config`
		Local host entry in `/etc/ssh/shosts.equiv`, `/etc/hosts.equiv`, `~/.rhosts`, or `~/.shosts`

TABLE 17–1 Authentication Methods for Secure Shell *(Continued)*

Authentication Method	Local Host (Client) Requirements	Remote Host (Server) Requirements
RSA or DSA public key	User account Private key in ~/.ssh/id_rsa or ~/.ssh/id_dsa User's public key in ~/.ssh/id_rsa.pub or ~/.ssh/id_dsa.pub	User account User's public key in ~/.ssh/authorized_keys
Password-based	User account	User account Supports PAM.
.rhosts with RSA (v1) on server only	User account Local host public key in /etc/ssh/ssh_host_rsa1_key	User account Local host public key in /etc/ssh/ssh_known_hosts or ~/.ssh/known_hosts IgnoreRhosts no in /etc/ssh/sshd_config Local host entry in /etc/ssh/shosts.equiv, /etc/hosts.equiv, ~/.shosts, or ~/.rhosts

Secure Shell in the Enterprise

For a comprehensive discussion of Secure Shell on an Oracle 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 published by Sun Microsystems Press.

Secure Shell and the OpenSSH Project

The 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 Secure Shell, as are individual bug fixes and features. Internal development continues on the Secure Shell fork.

The following features are:implemented for the v2 protocol in this release of Secure Shell:

- ForceCommand keyword – Forces the execution of the specified command regardless of what the user types on the command line. This keyword is very useful inside a Match block. This sshd_config configuration option is similar to the command="..." option in $HOME/.ssh/authorized_keys.
- AES-128 passphrase protection – In this release, private keys that are generated by the ssh-keygen command are protected with the AES-128 algorithm. This algorithm protects newly-generated keys and re-encrypted keys, such as when a passphrase is changed.

- -u option to sftp-server command – Enables user to to set an explicit umask on files and directories. This option overrides the user's default umask. For an example, see the description of Subsystem on the sshd_config(4) man page.
- Additional keywords for Match blocks – AuthorizedKeysFile, ForceCommand, and HostbasedUsesNameFromPacketOnly are supported inside Match blocks. By default, the value of AuthorizedKeysFile is $HOME/.ssh/authorized_keys and HostbasedUsesNameFromPacketOnly is no. To use Match blocks, see "How to Create User and Host Exceptions to SSH System Defaults" on page 300.

While Oracle Solaris engineers provide bug fixes to the project, they have also integrated the following Oracle Solaris features into the fork of Secure Shell:

- PAM - Secure Shell uses PAM. The OpenSSH UsePAM configuration option is not supported.
- Privilege separation - Secure Shell does not use the privilege separation code from the OpenSSH project. Secure Shell separates the processing of auditing, record keeping and re-keying from the processing of the session protocols.

  Secure Shell privilege separation code is always on and cannot be switched off. The OpenSSH UsePrivilegeSeparation option is not supported.
- Locale - 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 Secure Shell negotiated locale settings.
- Auditing - Secure Shell is fully integrated into the Solaris audit service. For information about the audit service, see Part VII, "Auditing in Oracle Solaris."
- 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 - 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 309.

In the Oracle Solaris releases, Secure Shell resyncs the SSH_OLD_FORWARD_ADDR compatibility flag from the OpenSSH project. As of March 2011, the Secure Shell version is 1.5.

Secure Shell and FIPS-140 Support

When you use a Sun Crypto Accelerator 6000 card for Secure Shell operations, Secure Shell runs with FIPS-140 support at Level 3. Level 3 hardware is certified to resist physical tampering, use identity-based authentication, and isolate the interfaces that handle critical security parameters from the hardware's other interfaces.

Secure Shell (Task Map)

The following task map links to the task maps about configuring Secure Shell and for using the Secure Shell feature in Oracle Solaris.

Task	Description	For Instructions
Configure Secure Shell.	Guides administrators in configuring Secure Shell for users.	"Configuring Secure Shell (Task Map)" on page 297
Use Secure Shell.	Guides users in using Secure Shell.	"Using Secure Shell (Task Map)" on page 301

Configuring Secure Shell (Tasks)

By default, host-based authentication and the use of both protocols are not enabled in Secure Shell. Changing these defaults requires administrative intervention. Also, for port forwarding to work requires administrative intervention.

Configuring Secure Shell (Task Map)

The following task map points to procedures for configuring Secure Shell.

Task	Description	For Instructions
Configure host-based authentication.	Configures host-based authentication on the client and server.	"How to Set Up Host-Based Authentication for Secure Shell" on page 297
Configure port forwarding.	Enables users to use port forwarding.	"How to Configure Port Forwarding in Secure Shell" on page 300
Configure exceptions to SSH system defaults.	For users, hosts, groups, and addresses, specifies SSH settings that are different from the system defaults.	"How to Create User and Host Exceptions to SSH System Defaults" on page 300

▼ How to Set Up Host-Based Authentication for Secure Shell

The following procedure sets 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.

Before You Begin You must be in the root role.

1. **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

2. **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

3. **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/shosts.equiv` file.**

     client-host

   - **Or, you can instruct users to add an entry for the client to their `~/.shosts` file on the server.**

     client-host

4. **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
   ```

5. **Ensure that users of Secure Shell at your site have accounts on both hosts.**

6. **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 Secure Shell" on page 302.

- **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.

  a. **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"'
     ```

  b. **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

  c. **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 17–1 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 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 Configure Port Forwarding in 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 – Secure Shell port forwarding must use TCP connections. Secure Shell does not support UDP connections for port forwarding.

Before You Begin You must be in the root role.

1. **Configure a 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.

   ```
   # Port forwarding
   AllowTcpForwarding yes
   ```

2. **Restart the Secure Shell service.**

   ```
   remoteHost# svcadm restart network/ssh:default
   ```

   For information about managing persistent services, see Chapter 6, "Managing Services (Overview)," in *Oracle Solaris Administration: Common Tasks* and the svcadm(1M) man page.

3. **Verify that port forwarding can be used.**

   ```
   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.

Before You Begin You must be in the root role.

1. **Edit the sshd_config file.**

2 **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:

```
## sshd_config file
## Global settings

# 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 Secure Shell (Tasks)

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.

Using Secure Shell (Task Map)

The following task map points to user procedures for using Secure Shell.

Task	Description	For Instructions
Create a public/private key pair.	Enables access to Secure Shell for sites that require public-key authentication.	"How to Generate a Public/Private Key Pair for Use With Secure Shell" on page 302

Task	Description	For Instructions
Change your passphrase.	Changes the phrase that authenticates your private key.	"How to Change the Passphrase for a Secure Shell Private Key" on page 304
Log in with Secure Shell.	Provides encrypted Secure Shell communication when logging in remotely. The process is similar to using the `rsh` command.	"How to Log In to a Remote Host With Secure Shell" on page 304
Log in to Secure Shell without being prompted for a password.	Enables login by using an agent which provides your password to Secure Shell.	"How to Reduce Password Prompts in Secure Shell" on page 305
Use port forwarding in Secure Shell.	Specifies a local port or a remote port to be used in a Secure Shell connection over TCP.	"How to Use Port Forwarding in Secure Shell" on page 307
Copy files with Secure Shell.	Securely copies files between hosts.	"How to Copy Files With Secure Shell" on page 308
Securely connect from a host inside a firewall to a host outside the firewall.	Uses Secure Shell commands that are compatible with HTTP or SOCKS5 to connect hosts that are separated by a firewall.	"How to Set Up Default Connections to Hosts Outside a Firewall" on page 309

▼ How to Generate a Public/Private Key Pair for Use With 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.**

```
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.

```
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:
0e:fb:3d:57:71:73:bf:58:b8:eb:f3:a3:aa:df:e0:d1 jdoe@myLocalHost
```

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 Secure Shell" on page 304.

  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 Secure Shell" on page 305. For more information, see the `ssh-agent(1)` and `ssh-add(1)` man pages.

▼ How to Change the Passphrase for a 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 Secure Shell

1 **Start a Secure Shell session.**

Type the `ssh` command, and specify the name of the remote host and your login.

```
myLocalHost% ssh myRemoteHost -l username
```

A prompt questions the authenticity of the remote host:

```
The authenticity of host 'myRemoteHost' can't be established.
RSA key fingerprint in md5 is: 04:9f:bd:fc:3d:3e:d2:e7:49:fd:6e:18:4f:9c:26
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 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: Wed Sep  7 09:07:49 2011 from myLocalHost
Oracle Corporation      SunOS 5.11       September 2011
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 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 Secure Shell

If you do not want to type your passphrase and your password to use 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 Secure Shell session.**

```
myLocalHost% ssh myRemoteHost -l jdoe
```

You are not prompted for a passphrase.

Example 17–2 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)

myLocalHost% ssh-add -l
md5 1024 0e:fb:3d:53:71:77:bf:57:b8:eb:f7:a7:aa:df:e0:d1
/home/jdoe/.ssh/id_rsa(RSA)
md5 1024 c1:d3:21:5e:40:60:c5:73:d8:87:09:3a:fa:5f:32:53
/home/jdoe/.ssh/id_dsa(DSA)
```

User conducts Oracle Solaris Secure Shell transactions

```
myLocalHost% ssh-add -D
Identity removed:
/home/jdoe/.ssh/id_rsa(/home/jdoe/.ssh/id_rsa.pub)
/home/jdoe/.ssh/id_dsa(DSA)
```

▼ How to Use Port Forwarding in 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 Secure Shell server. For details, see "How to Configure Port Forwarding in Secure Shell" on page 300.

● **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 17–3 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 specifies the local port number for the IMAP server, as in `localhost:9143`.

Do not confuse `localhost` with `myLocalHost`. `myLocalHost` is a hypothetical host name. `localhost` is a keyword that identifies your local system.

Example 17–4 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
myLocalHost%
```

▼ How to Copy Files With Secure Shell

The following procedure shows 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 `scp` command prompts for authentication. For more information, see the `scp(1)` man page.

You can also use the `sftp` secure file transfer program. For more information, see the `sftp(1)` man page. For an example, see Example 17–5.

Note – The audit service can audit `sftp` transactions through the `ft` audit class. For `scp`, the audit service can audit access and exit for the `ssh` session.

---

1 **Start the secure copy program.**

Specify the source file, the user name 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
myfile.1
```

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)

Example 17–5 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 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 17–6.

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.

1 **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`, `HTTPPROXYPORT`, `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 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 17–6 Connecting to Hosts Outside a Firewall From the Command Line

"How to Set Up Default Connections to Hosts Outside a Firewall" on page 309 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

◆ ◆ ◆ CHAPTER 18

Secure Shell (Reference)

This chapter describes the configuration options in the Secure Shell feature of Oracle Solaris. The following is a list of the reference information in this chapter.

- "A Typical Secure Shell Session" on page 311
- "Client and Server Configuration in Secure Shell" on page 313
- "Keywords in Secure Shell" on page 314
- "Maintaining Known Hosts in Secure Shell" on page 319
- "Secure Shell Files" on page 320
- "Secure Shell Commands" on page 322

For procedures to configure Secure Shell, see Chapter 17, "Using Secure Shell (Tasks)."

A Typical Secure Shell Session

The Secure Shell daemon (`sshd`) is normally started at boot time when network services are started. The daemon listens for connections from clients. A 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 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 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 Secure Shell

The Secure Shell protocol supports client user/host authentication and server host authentication. Cryptographic keys are exchanged for the protection of Secure Shell sessions. Secure Shell provides various methods for authentication and key exchange. Some methods are optional. Client authentication mechanisms are listed in Table 17–1. Servers are authenticated by using known host public keys.

For authentication, Secure Shell supports user authentication and generic interactive authentication, which usually involves passwords. Secure Shell 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, Secure Shell can use GSS credentials for authentication.

Acquiring GSS Credentials in Secure Shell

To use GSS-API for authentication in 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 14, "Network Services Authentication (Tasks)." For the use of mech_krb5, see Chapter 19, "Introduction to the Kerberos Service." For more information about mechanisms, see the mech(4) and mech_spnego(5) man pages.

Command Execution and Data Forwarding in Secure Shell

After authentication is complete, the user can use 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 Secure Shell

The characteristics of a 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 Secure Shell

In most cases, the client-side characteristics of a 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 Secure Shell" on page 314. For detailed information, see the sshd_config(4) man page.

The keywords in the client configuration file are listed in "Keywords in Secure Shell" on page 314. 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 18–8.

Server Configuration in Secure Shell

The server-side characteristics of a Secure Shell session are governed by the /etc/ssh/sshd_config file. The keywords in the server configuration file are listed in "Keywords in Secure Shell" on page 314. 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 Secure Shell

The following tables list the keywords and their default values, if any. The keywords are in alphabetical order. Keywords that apply to the client are in the ssh_config file. Keywords that apply to the server are in the sshd_config file. Some keywords are set in both files. Keywords for a Secure Shell server that is running the v1 protocol are marked.

TABLE 18–1 Keywords in Secure Shell Configuration Files (A to Escape)

Keyword	Default Value	Location
AllowGroups	No default.	Server
AllowTcpForwarding	yes	Server
AllowUsers	No default.	Server
AuthorizedKeysFile	~/.ssh/authorized_keys	Server
Banner	/etc/issue	Server
Batchmode	no	Client
BindAddress	No default.	Client
CheckHostIP	yes	Client
ChrootDirectory	no	Server
Cipher	blowfish, 3des	Client

TABLE 18–1 Keywords in Secure Shell Configuration Files (A to Escape) *(Continued)*

Keyword	Default Value	Location
`Ciphers`	`aes128-ctr, aes128-cbc, 3des-cbc, blowfish-cbc, arcfour`	Both
`ClearAllForwardings`	`no`	Client
`ClientAliveCountMax`	`3`	Server
`ClientAliveInterval`	`0`	Server
`Compression`	`no`	Both
`CompressionLevel`	No default.	Client
`ConnectionAttempts`	`1`	Client
`ConnectTimeout`	System TCP timeout	Client
`DenyGroups`	No default	Server
`DenyUsers`	No default	Server
`DisableBanner`	`no`	Client
`DynamicForward`	No default.	Client
`EscapeChar`	`~`	Client

TABLE 18–2 Keywords in Secure Shell Configuration Files (Fall to Local)

Keyword	Default Value	Location
`FallBackToRsh`	`no`	Client
`ForwardAgent`	`no`	Client
`ForwardX11`	`no`	Client
`ForwardX11Trusted`	`yes`	Client
`GatewayPorts`	`no`	Both
`GlobalKnownHostsFile`	`/etc/ssh/ssh_known_hosts`	Client
`GSSAPIAuthentication`	`yes`	Both
`GSSAPIDelegateCredentials`	`no`	Client
`GSSAPIKeyExchange`	`yes`	Both
`GSSAPIStoreDelegateCredentials`	`yes`	Server
`HashKnownHosts`	`no`	Client

TABLE 18–2 Keywords in Secure Shell Configuration Files (Fall to Local) *(Continued)*

Keyword	Default Value	Location
Host	* For more information, see "Host-Specific Parameters in Secure Shell" on page 318.	Client
HostbasedAuthentication	no	Both
HostbasedUsesNameFromPacketOnly	no	Server
HostKey	/etc/ssh/ssh_host_key	Server, v1
HostKey	/etc/ssh/host_rsa_key, /etc/ssh/host_dsa_key	Server
HostKeyAlgorithms	ssh-rsa, ssh-dss	Client
HostKeyAlias	No default.	Client
HostName	No default.	Client
IdentityFile	~/.ssh/id_dsa, ~/.ssh/id_rsa	Client
IgnoreIfUnknown	No default	Client
IgnoreRhosts	yes	Server
IgnoreUserKnownHosts	yes	Server
KbdInteractiveAuthentication	yes	Both
KeepAlive	yes	Both
KeyRegenerationInterval	3600 (seconds)	Server
ListenAddress	No default.	Server
LocalForward	No default.	Client

TABLE 18–3 Keywords in Secure Shell Configuration Files (Login to R)

Keyword	Default Value	Location
LoginGraceTime	120 (seconds)	Server
LogLevel	info	Both
LookupClientHostnames	yes	Server
MACs	hmac-sha1,hmac-md5	Both
Match	No default	Server
MaxStartups	10:30:60	Server

TABLE 18–3 Keywords in Secure Shell Configuration Files (Login to R) *(Continued)*

Keyword	Default Value	Location
NoHostAuthenticationForLocalHost	no	Client
NumberOfPasswordPrompts	3	Client
PAMServiceName	No default	Server
PAMServicePrefix	No default	Server
PasswordAuthentication	yes	Both
PermitEmptyPasswords	no	Server
PermitRootLogin	no	Server
PermitUserEnvironment	no	Server
PidFile	/system/volatile/sshd.pid	Server
Port	22	Both
PreferredAuthentications	hostbased,publickey,keyboard-interactive,passwor	Client
PreUserauthHook	No default	Server
PrintLastLog	yes	Server
PrintMotd	no	Server
Protocol	2,1	Both
ProxyCommand	No default.	Client
PubkeyAuthentication	yes	Both
RekeyLimit	1G to 4G	Client
RemoteForward	No default.	Client
RhostsAuthentication	no	Server, v1
RhostsRSAAuthentication	no	Server, v1
RSAAuthentication	no	Server, v1

TABLE 18–4 Keywords in Secure Shell Configuration Files (S to X)

Keyword	Default Value	Location
ServerAliveCountMax	3	Client
ServerAliveInterval	0	Client

TABLE 18–4 Keywords in Secure Shell Configuration Files (S to X) *(Continued)*

Keyword	Default Value	Location
`ServerKeyBits`	512 to 768	Server, v1
`StrictHostKeyChecking`	`ask`	Client
`StrictModes`	`yes`	Server
`Subsystem`	`sftp /usr/lib/ssh/sftp-server`	Server
`SyslogFacility`	`auth`	Server
`UseOpenSSLEngine`	`yes`	Both
`UsePrivilegedPort`	`no`	Both
`User`	No default	Client
`UserKnownHostsFile`	`~/.ssh/known_hosts`	Client
`UseRsh`	`no`	Client
`VerifyReverseMapping`	`no`	Server
`X11DisplayOffset`	`10`	Server
`X11Forwarding`	`yes`	Server
`X11UseLocalHost`	`yes`	Server
`XAuthLocation`	`/usr/openwin/bin/xauth`	Both

Host-Specific Parameters in Secure Shell

If it is useful to have different 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.

Secure Shell and Login Environment Variables

When the following Secure Shell keywords are not set in the `sshd_config` file, they obtain their value from equivalent entries in the `/etc/default/login` file.

Entry in `/etc/default/login`	Keyword and Value in `sshd_config`
`CONSOLE=*`	`PermitRootLogin=without-password`

Entry in /etc/default/login	Keyword and Value in sshd_config
#CONSOLE=*	PermitRootLogin=yes
PASSREQ=YES	PermitEmptyPasswords=no
PASSREQ=NO	PermitEmptyPasswords=yes
#PASSREQ	PermitEmptyPasswords=no
TIMEOUT=*secs*	LoginGraceTime=*secs*
#TIMEOUT	LoginGraceTime=120
RETRIES and SYSLOG_FAILED_LOGINS	Apply only to password and keyboard-interactive authentication methods.

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.

TIMEZONE Controls the setting of the TZ environment variable. When not set, the sshd daemon uses value of TZ when the daemon was started.

ALTSHELL Controls the setting of the SHELL environment variable. The default is ALTSHELL=YES, where the sshd daemon uses the value of the user's shell. When ALTSHELL=NO, the SHELL value is not set.

PATH Controls the setting of the PATH environment variable. When the value is not set, the default path is /usr/bin.

SUPATH Controls the setting of the PATH environment variable for root. When the value is not set, the default path is /usr/sbin:/usr/bin.

For more information, see the login(1) and sshd(1M) man pages.

Maintaining Known Hosts in 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 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 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.

Secure Shell Files

The following table shows the important Secure Shell files and the suggested file permissions.

TABLE 18–5 Secure Shell Files

File Name	Description	Suggested Permissions and Owner
/etc/ssh/sshd_config	Contains configuration data for sshd, the Secure Shell daemon.	-rw-r--r-- root
/etc/ssh/ssh_host_dsa_key or /etc/ssh/ssh_host_rsa_key	Contains the host private key.	-rw------- root
host-private-key.pub	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.	-rw-r--r-- root
/system/volatile/sshd.pid	Contains the process ID of the Secure Shell daemon, sshd. If multiple daemons are running, the file contains the last daemon that was started.	-rw-r--r-- root
~/.ssh/authorized_keys	Holds the public keys of the user who is allowed to log in to the user account.	-rw-r--r-- *username*
/etc/ssh/ssh_known_hosts	Contains the host public keys for all hosts with which the client can communicate securely. The file is populated by the administrator.	-rw-r--r-- root
~/.ssh/known_hosts	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.	-rw-r--r-- *username*
/etc/default/login	Provides defaults for the sshd daemon when corresponding sshd_config parameters are not set.	-r--r--r-- root

TABLE 18–5 Secure Shell Files *(Continued)*

File Name	Description	Suggested Permissions and Owner
`/etc/nologin`	If this file exists, the `sshd` daemon only permits `root` to log in. The contents of this file are displayed to users who are attempting to log in.	`-rw-r--r-- root`
`~/.rhosts`	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 `rlogind` and `rshd` daemons.	`-rw-r--r--` *username*
`~/.shosts`	Contains the host-user name pairs that specify the hosts to which the user can log in without a password. This file is not used by other utilities. For more information, see the `sshd`(1M) man page in the `FILES` section.	`-rw-r--r--` *username*
`/etc/hosts.equiv`	Contains the hosts that are used in `.rhosts` authentication. This file is also used by the `rlogind` and `rshd` daemons.	`-rw-r--r-- root`
`/etc/ssh/shosts.equiv`	Contains the hosts that are used in host-based authentication. This file is not used by other utilities.	`-rw-r--r-- root`
`~/.ssh/environment`	Contains initial assignments at login. By default, this file is not read. The `PermitUserEnvironment` keyword in the `sshd_config` file must be set to `yes` for this file to be read.	`-rw-r--r--` *username*
`~/.ssh/rc`	Contains initialization routines that are run before the user shell starts. For a sample initialization routine, see the `sshd`(1M) man page.	`-rw-r--r--` *username*
`/etc/ssh/sshrc`	Contains host-specific initialization routines that are specified by an administrator.	`-rw-r--r-- root`
`/etc/ssh/ssh_config`	Configures system settings on the client system.	`-rw-r--r-- root`
`~/.ssh/config`	Configures user settings which override system settings.	`-rw-r--r--` *username*

The following table lists the Secure Shell files that can be overridden by keywords or command options.

TABLE 18–6 Overrides for the Location of Secure Shell Files

File Name	Keyword Override	Command-Line Override
`/etc/ssh/ssh_config`		`ssh -F` *config-file* `scp -F` *config-file*
`~/.ssh/config`		`ssh -F` *config-file*
`/etc/ssh/host_rsa_key` `/etc/ssh/host_dsa_key`	`HostKey`	

TABLE 18–6 Overrides for the Location of Secure Shell Files *(Continued)*

File Name	Keyword Override	Command-Line Override

`~/.ssh/identity`
`~/.ssh/id_dsa`, `~/.ssh/id_rsa`
	`IdentityFile`	`ssh -i` *id-file*
		`scp -i` *id-file*

`~/.ssh/authorized_keys`	`AuthorizedKeysFile`	
`/etc/ssh/ssh_known_hosts`	`GlobalKnownHostsFile`	
`~/.ssh/known_hosts`	`UserKnownHostsFile` `IgnoreUserKnownHosts`	

Secure Shell Commands

The following table summarizes the major Secure Shell commands.

TABLE 18–7 Commands in Secure Shell

Man Page for Command	Description
`ssh(1)`	Logs a user in to a remote machine and securely executes commands on a remote machine. This command is the Secure Shell replacement for the `rlogin` and `rsh` commands. The `ssh` 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.
`sshd(1M)`	Is the daemon for Secure Shell. The daemon listens for connections from clients and enables secure encrypted communications between two untrusted hosts over an insecure network.
`ssh-add(1)`	Adds RSA or DSA identities to the authentication agent, `ssh-agent`. Identities are also called *keys*.
`ssh-agent(1)`	Holds private keys that are used for public key authentication. The `ssh-agent` 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 `ssh-agent` program. Through the use of environment variables, the agent can be located and used for authentication when users use the `ssh` command to log in to other systems.
`ssh-keygen(1)`	Generates and manages authentication keys for Secure Shell.
`ssh-keyscan(1)`	Gathers the public keys of a number of Secure Shell hosts. Aids in building and verifying `ssh_known_hosts` files.
`ssh-keysign(1M)`	Is used by the `ssh` command to access the host keys on the local host. Generates the digital signature that is required during host-based authentication with Secure Shell v2. The command is invoked by the `ssh` command, not by the user.
`scp(1)`	Securely copies files between hosts on a network over an encrypted `ssh` transport. Unlike the `rcp` command, the `scp` command prompts for passwords or passphrases, if password information is needed for authentication.

TABLE 18–7 Commands in Secure Shell *(Continued)*

Man Page for Command	Description
sftp(1)	Is an interactive file transfer program that is similar to the `ftp` command. Unlike the `ftp` command, the `sftp` command performs all operations over an encrypted `ssh` transport. The command connects, logs in to the specified host name, and then enters interactive command mode.

The following table lists the command options that override Secure Shell keywords. The keywords are specified in the `ssh_config` and `sshd_config` files.

TABLE 18–8 Command-Line Equivalents for Secure Shell Keywords

Keyword	`ssh` Command-Line Override	`scp` Command-Line Override
`BatchMode`		`scp -B`
`BindAddress`	`ssh -b` *bind-addr*	`scp -a` *bind-addr*
`Cipher`	`ssh -c` *cipher*	`scp -c` *cipher*
`Ciphers`	`ssh -c` *cipher-spec*	`scp -c` *cipher-spec*
`Compression`	`ssh -C`	`scp -C`
`DynamicForward`	`ssh -D` *SOCKS4-port*	
`EscapeChar`	`ssh -e` *escape-char*	
`ForwardAgent`	`ssh -A` to enable `ssh -a` to disable	
`ForwardX11`	`ssh -X` to enable `ssh -x` to disable	
`GatewayPorts`	`ssh -g`	
`IPv4`	`ssh -4`	`scp -4`
`IPv6`	`ssh -6`	`scp -6`
`LocalForward`	`ssh -L` *localport:remotehost:remoteport*	
`MACS`	`ssh -m` *mac-spec*	
`Port`	`ssh -p` *port*	`scp -P` *port*
`Protocol`	`ssh -2` for v2 only	
`RemoteForward`	`ssh -R` *remoteport:localhost:localport*	

PART VI

Kerberos Service

This section provides information about the configuration, management and use of the Kerberos service in the following chapters:

- Chapter 19, "Introduction to the Kerberos Service"
- Chapter 20, "Planning for the Kerberos Service"
- Chapter 21, "Configuring the Kerberos Service (Tasks)"
- Chapter 22, "Kerberos Error Messages and Troubleshooting"
- Chapter 23, "Administering Kerberos Principals and Policies (Tasks)"
- Chapter 24, "Using Kerberos Applications (Tasks)"
- Chapter 25, "The Kerberos Service (Reference)"

CHAPTER 19

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 327
- "How the Kerberos Service Works" on page 328
- "Kerberos Security Services" on page 334
- "Components of Various Kerberos Releases" on page 335

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 Kerberos service in the Oracle Solaris release is based on the Kerberos V5 network authentication protocol that was developed at the Massachusetts Institute of Technology (MIT). People who have used the Kerberos V5 product will therefore find the Oracle Solaris version very familiar. Because the Kerberos V5 protocol is a *de facto* industry standard for network security, the Oracle Solaris version promotes interoperability with other systems. In other words, because the Kerberos service in the Oracle Solaris release 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 Oracle 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 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 505.

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. "Transparently" 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 506.

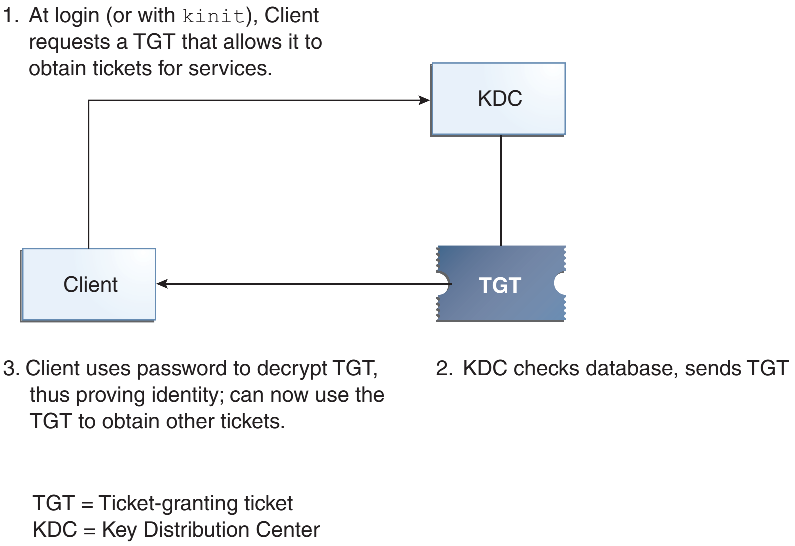
The following sections further explain the Kerberos authentication process.

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.

FIGURE 19–1 Initial Authentication for a Kerberos Session



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

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.

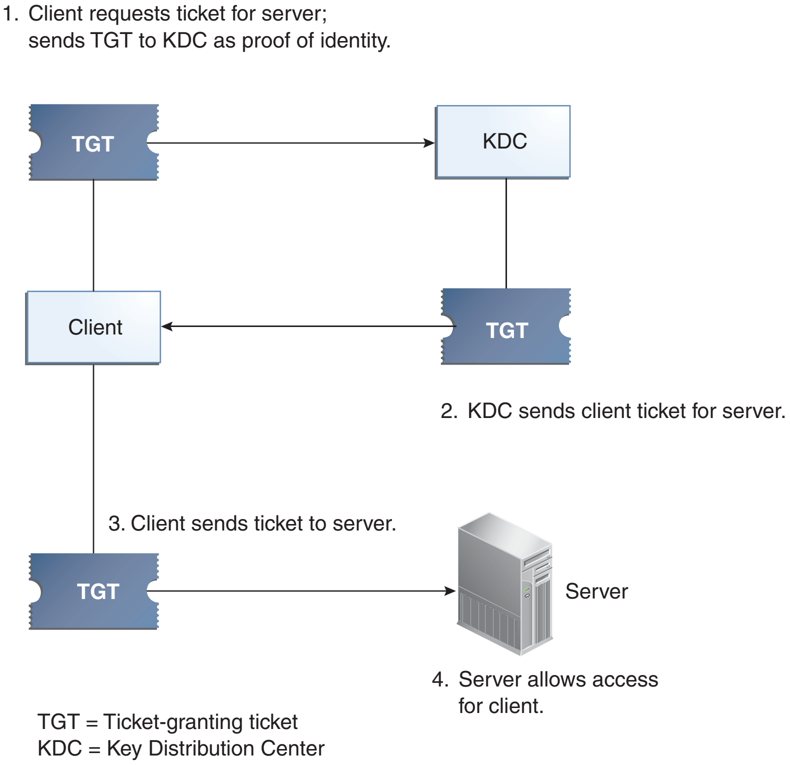
Another analogy for the ticket-granting ticket is that of a three-day ski pass that is good at four different ski resorts. You show the pass at whichever resort you decide to go to and you receive a lift ticket for that resort, as long as the pass has not expired. Once you have the lift ticket, you can 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.

FIGURE 19–2 Obtaining Access to a Service Using Kerberos Authentication


1. Client requests ticket for server;
sends TGT to KDC as proof of identity.
TGT
KDC
Client
TGT
2. KDC sends client ticket for server.
3. Client sends ticket to server.
TGT
Server
4. Server allows access
for client.
TGT = Ticket-granting ticket
KDC = Key Distribution Center

1. The client requests a ticket for a particular service, for example, to remote log in 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.
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.

Kerberos Remote Applications

The Kerberos-based (or "Kerberized") commands that a user such as `joe` can use are the following:

- `ftp`
- `rcp`
- `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 332 for information on principals.

These commands are discussed further in "Kerberos User Commands" on page 490.

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 333.

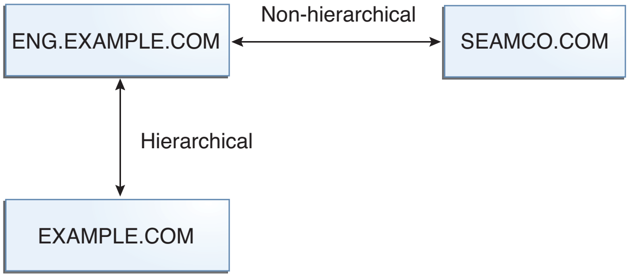
The following are all valid principal names:

- 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 19–3 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*.

FIGURE 19–3 Kerberos Realms



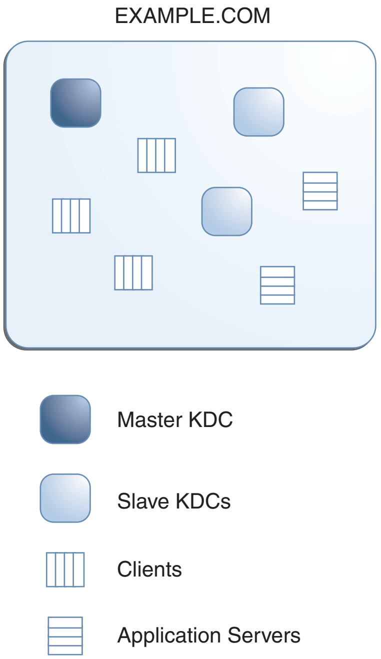
Kerberos Servers

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.

FIGURE 19–4 A Typical Kerberos Realm



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.

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 components of the SEAM product were included in the Oracle Solaris software, the contents of the SEAM release decreased. Starting with the Oracle Solaris 10 release, all components 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 19–1 Kerberos Release Contents

Release Name	Contents
SEAM 1.0 in Solaris Easy Access Server 3.0	Full release of the Kerberos service for the Solaris 2.6 and 7 releases
The Kerberos service in the Solaris 8 release	Kerberos client software only
SEAM 1.0.1 in the Solaris 8 Admin Pack	Kerberos KDC and remote applications for the Solaris 8 release
The Kerberos service in the Solaris 9 release	Kerberos KDC and client software only
SEAM 1.0.2	Kerberos remote applications for the Solaris 9 release
The Kerberos service starting in the Oracle Solaris 10 release	Full release of the Kerberos service with enhancements

For more information about enhancements included in the Oracle Solaris 10 release, see Kerberos Components.

Kerberos Components

Similar to the MIT distribution of the Kerberos V5 product, the Kerberos service in the Oracle Solaris release includes the following:

- Key Distribution Center (KDC):
  - Kerberos database administration daemon – `kadmind`.
  - 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, 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 Kerberos service in the Oracle Solaris release includes the following:

- A Kerberos Administration GUI-based Tool (gkadmin) – Enables you to administer the principals and principal policies. This Java technology-based GUI is an alternative to the kadmin 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.

About Kerberos in The Oracle Solaris 11 Release

This section lists the changes that are available in the Oracle Solaris 11 release.

- The Kerberos software has been synchronized with the MIT 1.8 release. The following features were included:
  - The arcfour-hmac-md5-exp, des-cbc-md5, and des-cbc-crc weak encryption types are disallowed by default. The allow_weak_crypto = true declaration in the /etc/krb5/krb5.conf file can be added to allow use of the weaker encryption algorithms.
  - In the /etc/krb5/krb5.conf file, the permitted_enctypes relation can take an optional DEFAULT keyword with + or – enctyp_family to add of remove a specific encryption type from the default set.

  - In most cases, you can eliminate the need for the `domain_realm` mapping table on the client side, by implementing minimal referral support in the KDC and providing the mapping information to clients through that protocol. Clients can function with no `domain_realm` mapping table, by sending requests for the service principal name `service/canonical-fqdn@LOCAL.REALM` to the local KDC and requesting referrals. This capability can be limited to service principal names with specific name types or in specific forms. The KDC can use only its `domain_realm` mapping table. No blocking queries to DNS can be introduced.
  - You can create aliases for principal entries if you are using an LDAP backend for the Kerberos database. Principal alias support is useful if a service can be accessed by different host names or if DNS is not available to canonicalize the host name, meaning that the short form is being used. You can use an alias for the various principal names a service is known by and the system only needs one set of keys for the actual service principal in its keytab file.
  - You can use the `kvno` utility to diagnose issues with service principal keys that are stored in `/etc/krb5/krb5.keytab`.
  - The `kadmin ktadd` command supports the `-norandkey` option which prevents the `kadmind` command from creating a new randomized key. The `-norandkey` option can be useful when you want to create a keytab for a principal that has a password-derived key. You can create a keytab that can be used to run the `kinit` command without having to specify a password.
  - Principals can be locked out after a certain number of preauthentication failures within a given time limit. See "How to Configure Account Lockout" on page 397 for more information.
  - The `OK_AS_DELEGATE` flag enables the KDC to communicate the local realm policy to a client regarding whether an intermediate server is trusted to accept delegated credentials. See "Trusts of Services for Delegation" on page 346 for more information.
- A set of user-level statically defined trace points for Kerberos has been added. These probes provide a logical view into Kerberos protocol messages. See "Using DTrace With the Kerberos Service" on page 442 for an example.
- The `kclient` script has been enhanced. The script includes the capability of joining Microsoft Active Directory servers. For instructions, see "How to Interactively Configure a Kerberos Client" on page 384 and "How to Configure a Kerberos Client for an Active Directory Server" on page 387. In addition, the script includes a `-T` option that can 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.
- 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.

- Support for accessing Kerberos principal and policy records by 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 directory servers. See "Managing a KDC on an LDAP Directory Server" on page 419 for a list of LDAP-related procedures.
- The new `kdcmgr` command can be used to automatically or interactively setup any KDC. This command creates both master and slave KDC servers. Also, when used with the `status` option, the `kdcmgr` command shows information about any KDC that is installed on the local host. Look for the pointers to the automatic and interactive procedures in Table 21–1.
- Support for Oracle Solaris clients that require no additional setup has been added to this release. Changes were made to the Kerberos service and to some default values. Kerberos clients work with no client-side configuration in environments that are appropriately configured. See "Client Configuration Options" on page 344 for more information.

CHAPTER 20

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 339
- “Planning Kerberos Realms” on page 340
- “Mapping Host Names Onto Realms” on page 341
- “Client and Service Principal Names” on page 341
- “Ports for the KDC and Admin Services” on page 342
- “The Number of Slave KDCs” on page 342
- “Which Database Propagation System to Use” on page 344
- “Clock Synchronization Within a Realm” on page 344
- “Client Configuration Options” on page 344
- “Improving Client Login Security” on page 345
- “KDC Configuration Options” on page 345
- “Trusts of Services for Delegation” on page 346
- “Kerberos Encryption Types” on page 346
- “Online Help URL in the Graphical Kerberos Administration Tool” on page 347

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 hierarchal 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 333. For the configuration procedures for multiple realms, see "Configuring Cross-Realm Authentication" on page 370.

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.

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`, `kadmind`, 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 399.

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 378
- “How to Add a Single Entry to the Credential Table” on page 378
- “How to Provide Credential Mapping Between Realms” on page 379
- “Observing Mapping From GSS Credentials to UNIX Credentials” on page 442

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 395 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 403.

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 415.

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 398 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 file.

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 381 for a description of all the client configuration processes.

Improving Client Login Security

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" on page 393 for more information.

KDC Configuration Options

There are several ways to configure a KDC. The simplest ways 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 21–1 for pointers to the instructions for using this command.

Also available is support for using LDAP to manage the database files for Kerberos. See "How to Configure a KDC to Use an LDAP Data Server" on page 358 for instructions. Using LDAP simplifies administration for sites that require better coordination between the Kerberos databases and their existing directory server setup.

Trusts of Services for Delegation

For some applications, a client might need to delegate authority to a server to act on its behalf in contacting other services. The client must forward credentials to an intermediate server. The client's ability to obtain a service ticket to a server conveys no information to the client about whether the server should be trusted to accept delegated credentials. The `ok_to_auth_as_delegate` option to the `kadmin` command provides a way for a KDC to communicate the local realm policy to a client regarding whether an intermediate server is trusted to accept such credentials.

The copy of the credential ticket flags in the encrypted part of the KDC reply might have the `ok_to_auth_as_delegate` option set to indicate to the client that the server specified in the ticket has been determined by the policy of the realm to be a suitable recipient of delegation. A client can use the presence of this information to determine whether to delegate credentials (by granting either a proxy or a forwarded TGT) to this server. When setting this option, an administrator must consider the security and placement of the server on which the service runs, as well as whether the service requires the use of delegated credentials.

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 509 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.

The `arcfour-hmac-md5-exp`, `des-cbc-md5`, and `des-cbc-crc` weak encryption types are disallowed by default in the Oracle Solaris 11 release. If you need to continue using these encryption types, then set `allow_weak_crypto = true` in the `libdefaults` section of the `/etc/krb5/krb5.conf` file.

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://www.oracle.com/technetwork/indexes/documentation/index.html`.

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 "SEAM Tool" on page 444 in the *Administering Kerberos Principals and Policies (Tasks)* chapter in this book. You can choose another HTML page, if another location is more appropriate.

◆ ◆ ◆ CHAPTER 21

Configuring the Kerberos Service (Tasks)

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 349
- "Configuring KDC Servers" on page 350
- "Configuring Kerberos Clients" on page 381
- "Configuring Cross-Realm Authentication" on page 370
- "Configuring Kerberos Network Application Servers" on page 372
- "Configuring Kerberos NFS Servers" on page 375
- "Synchronizing Clocks Between KDCs and Kerberos Clients" on page 398
- "Swapping a Master KDC and a Slave KDC" on page 399
- "Administering the Kerberos Database" on page 403
- "Increasing Security on Kerberos Servers" on page 421

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.

Task	Description	For Instructions
1. Plan for your Kerberos installation.	Lets you resolve configuration issues before you start the software configuration process. Planning ahead saves you time and other resources in the long run.	Chapter 20, "Planning for the Kerberos Service"

Task	Description	For Instructions
2. (Optional) Install NTP.	Configures the Network Time Protocol (NTP) software, or another clock synchronization protocol. In order for the Kerberos service to work properly, the clocks on all systems in the realm must be synchronized.	"Synchronizing Clocks Between KDCs and Kerberos Clients" on page 398
3. Configure the KDC servers.	Configures and builds the master KDC and the slave KDC servers and the KDC database for a realm.	"Configuring KDC Servers" on page 350
4. (Optional) Increase security on the KDC servers.	Prevents security breaches on the KDC servers.	"How to Restrict Access to KDC Servers" on page 422
5. (Optional) Configure swappable KDC servers.	Makes the task of swapping the master KDC and a slave KDC easier.	"How to Configure a Swappable Slave KDC" on page 399

Configuring Additional Kerberos Services (Task Map)

Once the required steps have been completed, the following procedures can be used, when appropriate.

Task	Description	For Instructions
Configure cross-realm authentication.	Enables communications from one realm to another realm.	"Configuring Cross-Realm Authentication" on page 370
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 372
Configure Kerberos clients.	Enables a client to use Kerberos services.	"Configuring Kerberos Clients" on page 381
Configure Kerberos NFS server.	Enables a server to share a file system that requires Kerberos authentication.	"Configuring Kerberos NFS Servers" on page 375
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 421

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 21–1 Configuring KDC Servers (Task Map)

Task	Description	For Instructions
Configure a master KDC.	Configures and builds the master KDC server and database for a realm by using an automatic process, which is good for scripts.	"How to Automatically Configure a Master KDC" on page 351
	Configures and builds the master KDC server and database for a realm using an interactive process, which is sufficient for most installations	"How to Interactively Configure a Master KDC" on page 352
	Configures and builds the master KDC server and database for a realm using a manual process, which is needed for more complex installations	"How to Manually Configure a Master KDC" on page 353
	Configures and builds the master KDC server and database for a realm using a manual process and using LDAP for the KDC	"How to Configure a KDC to Use an LDAP Data Server" on page 358
Configure a slave KDC server.	Configures and builds a slave KDC server for a realm using an automatic process, which is good for scripts	"How to Automatically Configure a Slave KDC" on page 364
	Configures and builds a slave KDC server for a realm using an interactive process, which is sufficient for most installations	"How to Interactively Configure a Slave KDC" on page 365
	Configures and builds a slave KDC server for a realm using a manual process, which is needed for more complex installations	"How to Manually Configure a Slave KDC" on page 366
Refresh principal keys on a KDC server.	Updates the session key on a KDC server to use new encryption types.	"How to Refresh the Ticket-Granting Service Keys on a Master Server" on page 369

▼ How to Automatically Configure a Master KDC

In the Oracle Solaris 11 release, a master KDC can be automatically configured by using the following procedure.

1 **Become an administrator or assume a role or user name that has been assigned to the Kerberos Server Management profile.**

For more information, see "How to Obtain Administrative Rights" on page 160.

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

In the Oracle Solaris release, a master KDC can be interactively configured by using the following procedure.

1 **Become an administrator or assume a role or user name that has been assigned to the Kerberos Server Management profile.**

For more information, see "How to Obtain Administrative Rights" on page 160.

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 21–1 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`
- Online help URL = `http://download.oracle.com/docs/cd/E23824_01/html/821-1456/aadmin-23.html`

Note – Adjust the URL to point to the section, as described in "Online Help URL in the Graphical Kerberos Administration Tool" on page 347.

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 399.

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.

```
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 = {
        help_url = http://download.oracle.com/docs/cd/E23824_01/html/821-1456/aadmin-23.html
        }
```

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 509 for a description of the issues involved with restricting the encryption types.

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.

```
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 509 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.

```
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>
```

5 Edit the Kerberos access control list file (`kadm5.acl`).

Once populated, the `/etc/krb5/kadm5.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 `kadm5.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 `kiprop` principals.**

The `kiprop` principal is used to authorize updates from the master KDC.

```
kadmin.local: addprinc -randkey kiprop/kdc1.example.com
Principal "kiprop/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 `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 `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 `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 name service.

```
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.

```
kadmin: addprinc 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 by application servers, like `sshd`, 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/sha1 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.
kadmin:
```

d. **Quit `kadmin`.**

```
kadmin: quit
```

9 **(Optional) 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 `libdefaults` section of the `krb5.conf` file for authentication to succeed. See "Synchronizing Clocks Between KDCs and Kerberos Clients" on page 398 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 366 for specific instructions.

▼ How to Configure a KDC to Use an LDAP Data Server

Use the following procedure to configure a KDC to use an LDAP data server..

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`
- FMRI for the LDAP service = `svc:/application/sun/ds:ds--var-opt-SUNWdsee-dsins1`
- Online help URL = `http://download.oracle.com/docs/cd/E23824_01/html/821-1456/aadmin-23.html`

> **Note –** Adjust the URL to point to the section, as described in "Online Help URL in the Graphical Kerberos Administration Tool" on page 347.

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 following procedure works with servers using the Sun Directory Server Enterprise Edition 7.0 release.

1 **Become superuser on the KDC.**

2 **Configure the master KDC to use SSL to reach the directory server.**

The following steps configure an Oracle Solaris release KDC to use the Directory Server self-signed certificate. If the certificate has expired, follow the instructions for renewing a certificate in "To Manage Self-Signed Certificates".

a. **On the directory server, export the self-signed directory server certificate.**

```
# /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.**

```
# 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 managerentry has administration privileges.

```
/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 Kerberos schema to the existing schema.**

```
# 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.

```
database_module = LDAP
```

b. **Define the database module.**

```
[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"
        ldap_kadmind_dn = "cn=kadmin service,ou=profile,dc=example,dc=com"
        ldap_cert_path = /var/ldap
        ldap_servers = ldaps://dsserver.example.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: aci
aci: (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";)

# 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")(targetattr="krb*")(version 3.0;\
      acl kadmin_ACL; allow (all)\
      userdn = "ldap:///cn=kadmin 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 = {
        help_url = http://download.oracle.com/docs/cd/E23824_01/html/821-1456/aadmin-23.html
        }
```

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 509 for a description of the issues involved with restricting the encryption types.

---

10 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.

```
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 509 for a description of the issues involved with restricting the encryption types.

---

11 **Edit the Kerberos access control list file (`kadm5.acl`).**

Once populated, the /etc/krb5/kadm5.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 kadm5.acl(4) 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 **(Optional) Configure LDAP dependencies for Kerberos services.**

If the LDAP and KDC servers are running on the same host and if the LDAP service is configured with a SMF FMRI, add a dependency to the LDAP service for the Kerberos daemons. This dependency will restart the KDC service if the LDAP service is restarted.

a. **Add the dependency to the `krb5kdc` service.**

```
# svccfg -s security/krb5kdc
svc:/network/security/krb5kdc> addpg dsins1 dependency
svc:/network/security/krb5kdc> setprop dsins1/entities = \
    fmri: "svc:/application/sun/ds:ds--var-opt-SUNWdsee-dsins1"
svc:/network/security/krb5kdc> setprop dsins1/grouping = astring: "require_all"
svc:/network/security/krb5kdc> setprop dsins1/restart_on = astring: "restart"
svc:/network/security/krb5kdc> setprop dsins1/type = astring: "service"
svc:/network/security/krb5kdc> exit
```

b. **Add the dependency to the `kadmin` service.**

```
# svccfg -s security/kadmin
svc:/network/security/kadmin> addpg dsins1 dependency
svc:/network/security/kadmin> setprop dsins1/entities =\
    fmri: "svc:/application/sun/ds:ds--var-opt-SUNWdsee-dsins1"
svc:/network/security/kadmin> setprop dsins1/grouping = astring: "require_all"
svc:/network/security/kadmin> setprop dsins1/restart_on = astring: "restart"
svc:/network/security/kadmin> setprop dsins1/type = astring: "service"
svc:/network/security/kadmin> exit
```

14 **Start the Kerberos daemons.**

```
kdc1 # svcadm enable -r network/security/krb5kdc
kdc1 # svcadm enable -r network/security/kadmin
```

15 **Start `kadmin` and add more principals.**

At this point, you can add principals by using the GUI 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`. 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 name service.

```
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.

```
kadmin: addprinc 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/sha1 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.
kadmin:
```

d. **Quit `kadmin`.**

```
kadmin: quit
```

16 **(Optional) 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 `libdefaults` section of the `krb5.conf` file for authentication to succeed. See "Synchronizing Clocks Between KDCs and Kerberos Clients" on page 398 for information about NTP.

17 **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 366 for specific instructions.

▼ How to Automatically Configure a Slave KDC

In the Oracle Solaris release, a slave KDC can be automatically configured by using the following procedure.

1 **Become an administrator or assume a role or user name that has been assigned to the Kerberos Server Management profile.**

For more information, see "How to Obtain Administrative Rights" on page 160.

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.

```
kdc2# kdcmgr -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:      <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 Interactively Configure a Slave KDC

Use the following procedure to interactively configure a slave KDC.

1 **Become an administrator or assume a role or user name that has been assigned to the Kerberos Server Management profile.**

For more information, see "How to Obtain Administrative Rights" on page 160.

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 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 399.

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 name service.

      ```
      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
```

5 **On the master KDC, restart `kadmind` to use the new entries in the `kadm5.acl` file.**

```
kdc1 # svcadm restart network/security/kadmin
```

6 **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`

7 **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
```

8 **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.

9 **On the new slave, change an entry in `kdc.conf`.**

Replace the `sunw_dbprop_master_ulogsize` entry with an entry defining `sunw_dbprop_slave_poll`. The entry sets the poll time to two 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>
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 name service.

```
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/sha1 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.
kadmin:
```

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/sha1 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.
kadmin:
```

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.**

```
kdc2 # /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>
```

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 libdefaults section of the krb5.conf file for authentication to succeed. See “Synchronizing Clocks Between KDCs and Kerberos Clients” on page 398 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 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 KDC is updated to a release that supports additional, stronger encryption types, the administrator can 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 can be used.

- **Refresh the TGS service principal key.**

```
kdc1 % /usr/sbin/kadmin -p kws/admin
Enter password:      <Type kws/admin password>
kadmin: cpw -randkey krbtgt/EXAMPLE.COM@EXAMPLE.COM
```

Example 21–2 Refreshing the Principal Keys from a Master Server

If you are logged on to 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 341).

▼ 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.

```
# /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 krgtgt/ENG.EAST.EXAMPLE.COM@EAST.EXAMPLE.COM:      <Type password>
kadmin: addprinc krbtgt/EAST.EXAMPLE.COM@ENG.EAST.EXAMPLE.COM
Enter password for principal krgtgt/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.**

```
# cat /etc/krb5/krb5.conf
[libdefaults]
 .
 .
[domain_realm]
        .eng.east.example.com = ENG.EAST.EXAMPLE.COM
        .east.example.com = EAST.EXAMPLE.COM
```

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
  krgtgt/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 **Become superuser on the server.**

2 **(Optional) Install the NTP client or another clock synchronization mechanism.**

See "Synchronizing Clocks Between KDCs and Kerberos Clients" on page 398 for information about NTP.

3 **Add principals for the new server and update the server's keytab file.**

The following command reports the existence of the host principal:

```
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 GUI Kerberos Administration Tool to add a principal is explained in "How to Create a New Kerberos Principal" on page 453. The example in the following steps shows how to add the required principals using the command line. You must log in with one of the `admin` principal names that you created when configuring the master KDC.

```
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 in the following ways:

- 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.

```
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 file.**

If the kadmin command is not running, restart it with a command similar to the following: /usr/sbin/kadmin -p *kws*/admin

If the server has multiple host names 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/sha1 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:
```

c. **Quit kadmin.**

```
kadmin: quit
```

▼ How to Use the Generic Security Service With Kerberos When Running FTP

The generic security service (GSS) can be used to applications to easily use Kerberos for authentication, integrity, and privacy. The following steps show how to enable the GSS service for ProFTPD.

1 **Become superuser on the FTP server.**

2 **Add principals for the FTP server and update the server's keytab file.**

These steps might not be needed if the changes were made earlier.

a. **Start the `kadmin` command.**

```
ftpserver1 # /usr/sbin/kadmin -p kws/admin
Enter password:      <Type kws/admin password>
kadmin:
```

b. **Add the `host` service principal for the FTP server.**

```
kadmin: addprinc -randkey host/ftpserver1.example.com
```

c. **Add the `host` service principal to the server's `keytab` file..**

```
kadmin: ktadd host/ftpserver1.example.com
```

3 **Enable GSS for the FTP server.**

Make the following changes to the `/etc/proftpd.conf` file.

```
# cat /etc/proftpd.conf
#User          ftp
#Group         ftp

User           root
Group          root

UseIPv6 off

LoadModule     mod_gss.c

GSSEngine      on
GSSKeytab      /etc/krb5/krb5.keytab
```

4 **Restart the FTP server.**

```
# svcadm restart network/ftp
```

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 343 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 21–2 Configuring Kerberos NFS Servers (Task Map)

Task	Description	For Instructions
Configure a Kerberos NFS server.	Enables a server to share a file system that requires Kerberos authentication.	"How to Configure Kerberos NFS Servers" on page 376
Create a credential table.	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.	"How to Create a Credential Table" on page 378
Change the credential table that maps user credentials to UNIX UIDs.	Updates information in the credential table.	"How to Add a Single Entry to the Credential Table" on page 378
Create credential mappings between two like realms.	Provides instructions on how to map UIDs from one realm to another if the realms share a password file.	"How to Provide Credential Mapping Between Realms" on page 379
Share a file system with Kerberos authentication.	Shares a file system with security modes so that Kerberos authentication is required.	"How to Set Up a Secure NFS Environment With Multiple Kerberos Security Modes" on page 380

▼ 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 **Become superuser on the NFS server.**

2 **Complete the prerequisites for configuring a Kerberos NFS server.**

The master KDC must be configured. To fully test the process, you need several clients.

3 **(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 398 for information about NTP.

4 **Configure the NFS server as a Kerberos client.**

Follow the instructions in "Configuring Kerberos Clients" on page 381.

5 **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 453. 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 name service.

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.
kadmin:
```

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:
```

c. **Quit kadmin.**

```
kadmin: quit
```

6 **(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 343. If the default mapping is not sufficient, see "How to Create a Credential Table" on page 378 for more information.

7 **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 380 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.

1. **Become superuser on the NFS server.**

2. **Edit /etc/gss/gsscred.conf and change the security mechanism.**

   Change the mechanism to files.

3. **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 svc:/system/name-service/switch:default service. 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 378 for instructions.

1. **Become superuser on the NFS server.**

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 21–3 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 21–4 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 on the client system.**

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 21–5 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 442 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 376.

```
# 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
#
krb5            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.**

share -F nfs -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 21–6 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.

```
# share -F nfs -o sec=krb5 /export/home
```

Example 21–7 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.

```
# share -F nfs -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.

Task	Description	For Instructions
Establish a Kerberos client installation profile.	Generates a client installation profile that can be used to automatically install a Kerberos client.	"How to Create a Kerberos Client Installation Profile" on page 382
Configure a Kerberos client.	Manually installs a Kerberos client. Use this procedure if each client installation requires unique installation parameters.	"How to Manually Configure a Kerberos Client" on page 388
	Automatically installs a Kerberos client. Use this procedure if the installation parameters for each client are the same.	"How to Automatically Configure a Kerberos Client" on page 383
	Interactively installs a Kerberos client. Use this procedure if only a few of the installation parameters need to change.	"How to Interactively Configure a Kerberos Client" on page 384
	Automatically installs a Kerberos client of an Active Directory server.	"How to Configure a Kerberos Client for an Active Directory Server" on page 387
Allow a client to access a NFS file system as the `root` user	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.	"How to Access a Kerberos Protected NFS File System as the `root` User" on page 394
Disable verification of the KDC that issued a client Ticket Granting Ticket (TGT).	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.	"How to Disable Verification of the Ticket-Granting Ticket" on page 393

▼ 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 382.

1. **Become an administrator or assume a role or user name that has been assigned to the Kerberos Client Management profile.**

   For more information, see "How to Obtain Administrative Rights" on page 160.

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 21–8 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 a installation profile. In the Oracle Solaris 11 release, the `kclient` utility improved ease of use and the ability to work with Active Directory servers. See "How to Configure a Kerberos Client for an Active Directory Server" on page 387 for more information. See Example 21–10 for an example of running `kclient` on a previous release.

1. **Become an administrator or assume a role or user name that has been assigned to the Kerberos Client Management profile.**

   For more information, see "How to Obtain Administrative Rights" on page 160.

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 an Oracle Solaris release.**

   If this system is a client of a KDC server that is not running an Oracle 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 21–9 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 kerveros 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:
service:{first|only|optional}: xscreensaver:first
Configuring /etc/pam.conf.

Do you want to copy over the master krb5.conf file ? [y/n]: n
        No action performed.

---------------------------------------------------
Setup COMPLETE.
```

Example 21–10 Running the kclient Installation Utility in the Oracle 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.**

   ```
   client# sharectl set -p ddns_enable=true smb
   ```

3. **Run the `kclient` utility.**

   The `-T` option selects a KDC server type. In this case an Active Directory server is selected.

   ```
   client# kclient -T ms_ad
   ```

   By default, you will need to provide the password for the Administrator principal.

Example 21–11 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`
- Online help URL = `http://download.oracle.com/docs/cd/E23824_01/html/821-1456/aadmin-23.html`

Note – Adjust the URL to point to the section, as described in "Online Help URL in the Graphical Kerberos Administration Tool" on page 347.

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 = {
        help_url = http://download.oracle.com/docs/cd/E23824_01/html/821-1456/aadmin-23.html
```

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 509 for a description of the issues involved with restricting the encryption types.

3 **(Optional) Change the process that is used to locate the KDCs.**

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.**

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 domainname 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 398 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 453. 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. **(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>
      kadmin:
      ```

   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 name service.

```
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/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. **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.

```
kadmin: addprinc -randkey host/denver.example.com
Principal "host/denver.example.com@EXAMPLE.COM" created.
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.
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.

```
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:
```

e. **Quit `kadmin`.**

```
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.

```
# cat /etc/nfssec.conf
  .
  .
#
# 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 `svc:/network/dns/client:default` service is not enabled, enable it. See the `resolv.conf(4)` man page for more information.

c. **Restart the `gssd` service.**

```
# 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 21–12 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 21–13 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

                        IN      NS      kdc1.example.com.
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"
```

Example 21–14 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.

```
@ IN SOA kdc1.example.com root.kdc1.example.com (
                                1989020501   ;serial
                                10800        ;refresh
                                3600         ;retry
                                3600000      ;expire
                                86400 )      ;minimum

                                   IN      NS      kdc1.example.com.
kdc1                               IN      A       192.146.86.20
kdc2                               IN      A       192.146.86.21

_kerberos._udp.EXAMPLE.COM         IN      SRV 0 0 88  kdc2.example.com
_kerberos._tcp.EXAMPLE.COM         IN      SRV 0 0 88  kdc2.example.com
_kerberos._udp.EXAMPLE.COM         IN      SRV 1 0 88  kdc1.example.com
_kerberos._tcp.EXAMPLE.COM         IN      SRV 1 0 88  kdc1.example.com
_kerberos-adm._tcp.EXAMPLE.COM     IN      SRV 0 0 749 kdc1.example.com
_kpasswd._udp.EXAMPLE.COM          IN      SRV 0 0 749 kdc1.example.com
```

▼ How to Disable Verification of the Ticket-Granting Ticket

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 (TGT). 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.**

2 **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.

- **Start kadmin.**

  You can use the GUI Kerberos Administration Tool to add a principal, as explained in "How to Create a New Kerberos Principal" on page 453. 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 name service.

```
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.

```
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.**

```
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 gdm 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 Set up server1 as a Kerberos client of the realm EXAMPLE.COM. See “Configuring Kerberos Clients” on page 381 for more information.

1 **Become superuser.**

2 **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
```

3 Make changes to the PAM configuration file.

a. Add entries for the gdm service.

```
# cat /etc/pam.conf
 .
 .
#
# gdm service
#
gdm       auth requisite          pam_authtok_get.so.1
gdm       auth required           pam_dhkeys.so.1
gdm       auth required           pam_unix_cred.so.1
gdm       auth sufficient         pam_krb5.so.1
gdm       auth requisite          pam_unix_auth.so.1
gdm       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 that use the pam_krb5_migrate module. See the pam_krb5_migrate(5) man page for more information.

```
# cat /etc/pam.conf
 .
 .
gdm      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
# Used when service name is not explicitly mentioned for account management
#
other   account requisite      pam_roles.so.1
other   account required       pam_krb5.so.1
other   account required       pam_unix_account.so.1
```

d. Add the pam_krb5 module to the password stack.

This addition allows for passwords to be updated when the password expire.

```
# 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
```

4. **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.

```
kdc1 # cat /etc/krb5/kadm5.acl
host/server1.example.com@EXAMPLE.COM U root
host/server1.example.com@EXAMPLE.COM ui *
*/admin@EXAMPLE.COM *
```

5. **On the master KDC, restart the Kerberos administration daemon.**

   This step allows the `kadmind` daemon to use the new `kadm5.acl` entries.

```
kdc1 # svcadm restart network/security/kadmin
```

6. **On the master KDC, add entries to the `pam.conf` file.**

   The following entries enable the `kadmind` daemon to use the `k5migrate` PAM service, to validate UNIX user password for accounts that require migration.

```
# grep k5migrate /etc/pam.conf
k5migrate        auth    required        pam_unix_auth.so.1
k5migrate        account required        pam_unix_account.so.1
```

▼ How to Configure Account Lockout

- **Start `kadmin`.**

```
denver # /usr/sbin/kadmin -p kws/admin
Enter password:      <Type kws/admin password>
kadmin:
```

  a. **Create a policy with account lockout parameters.**

  In the following example, the `add_policy` subcommand command is used to create a policy named `default`. Three authentication failures, during a span of `300` seconds will trigger an account lockout of `900` seconds.

```
kadmin: add_policy -maxfailure 3 -failurecountinterval "300 seconds"\
-lockoutduration "900 seconds" default
```

  b. **Quit `kadmin`.**

```
kadmin: quit
```

Example 21–15 Unlocking a Locked Out Principal

In the following example, a user principal is unlocked:

```
# kadmin
kadmin: add_policy -unlock principal
```

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.

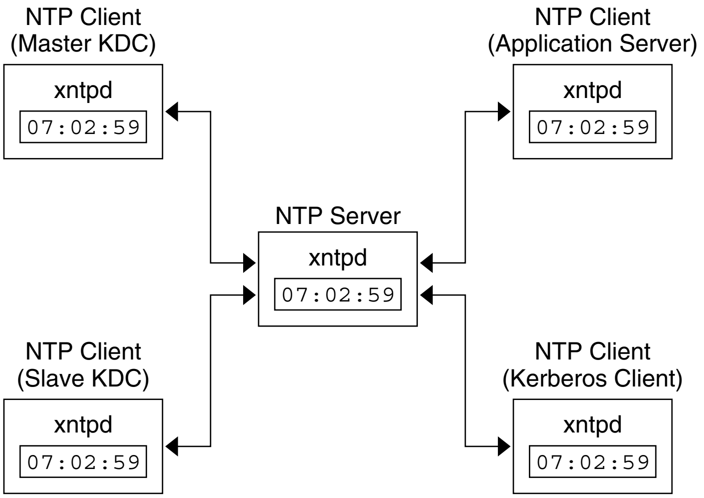
Because maintaining synchronized clocks between the KDCs and Kerberos clients is important, you should use the Network Time Protocol (NTP) software to synchronize them. NTP public domain software from the University of Delaware is included in the Oracle Solaris software.

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.

FIGURE 21–1 Synchronizing Clocks by Using NTP



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 *Oracle Solaris Administration: 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 *Oracle Solaris Administration: 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 366 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 399).

1 **Become superuser.**

2 **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
```

3 **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
```

4 **On the new master KDC, verify that the update is complete.**

```
kdc4 # /usr/sbin/kproplog -h
```

5 **On the new master KDC, restart the KDC service.**

```
kdc4 # svcadm enable -r network/security/krb5kdc
```

6 **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
```

7 **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
```

8 **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 two 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
        }
```

9 **On the old master KDC, move the master KDC commands and the `kadm5.acl` file.**

To prevent the master KDC commands from being run, move the `kprop`, `kadmind`, and `kadmin.local` commands to a reserved place.

```
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
```

10 On the DNS server, change the alias names for the master KDC.

To change the servers, edit the example.com zone file and change the entry for masterkdc.

```
masterkdc IN CNAME kdc4
```

11 On the DNS server, restart the Internet domain name server.

Run the following command to reload the new alias information:

```
# svcadm refresh network/dns/server
```

12 On the new master KDC, move the master KDC commands and the slave kpropd.acl file.

```
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
```

13 On the new master KDC, create the Kerberos access control list file (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 kadm5.acl(4) man page for more information.

```
kdc4 # cat /etc/krb5/kadm5.acl
kws/admin@EXAMPLE.COM   *
kiprop/kdc1.example.com@EXAMPLE.COM p
```

14 On the new master KDC, specify the update log size in the kdc.conf file.

Comment out the sunw_dbprop_slave_poll entry and add an entry defining sunw_dbprop_master_ulogsize. The entry sets the log size to 1000 entries.

```
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
                sunw_dbprop_master_ulogsize = 1000
        }
```

15 On the new master KDC, start kadmind and krb5kdc.

```
kdc4 # svcadm enable -r network/security/krb5kdc
kdc4 # svcadm enable -r network/security/kadmin
```

16 **On the old master KDC, add the `kiprop` service principal.**

Adding the `kiprop` 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 pricipal kws/admin@EXAMPLE.COM with password.
Enter password:      <Type kws/admin password>
kadmin: ktadd kiprop/kdc1.example.com
Entry for principal kiprop/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 kiprop/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 kiprop/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 kiprop/kdc1.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin: quit
```

17 **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
```

18 **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 353.

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 an administrator or assume a role or user name that has been assigned to the Kerberos Server Management profile.**

   For more information, see "How to Obtain Administrative Rights" on page 160.

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 21–16 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.

```
# 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.

```
# 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.**

```
kdc1 # svcadm disable network/security/krb5kdc
kdc1 # svcadm disable network/security/kadmin
```

3. **Restore the Kerberos database by using the `load` command of the `kdb_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 21–17 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 enables 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 **Become superuser.**

2 **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_ulogsize = 1000
        }
```

3 **Create the kiprop principal.**

The kiprop 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:
```

4 **On the master KDC, add a kiprop 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 *
kiprop/kdc2.example.com@EXAMPLE.COM p
```

5 **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
```

6 **Restart `kadmind`.**

```
kdc1 # svcadm restart network/security/kadmin
```

7 **Reconfigure all slave KDC servers that use incremental propagation.**

See "How to Reconfigure a Slave KDC to Use Incremental Propagation" on page 409 for complete instructions.

▼ How to Reconfigure a Slave KDC to Use Incremental Propagation

1 **Become superuser.**

2 **Add entries to `kdc.conf`.**

The first new entry enables incremental propagation. The second new entry sets the poll time to two minutes.

```
kdc2 # 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
        }
```

3 **Add the `kiprop` principal to the `krb5.keytab` file.**

```
kdc2 # /usr/sbin/kadmin -p kws/admin
Enter password:      <Type kws/admin password>
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/sha1 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.
kadmin: quit
```

4 **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 that is running the Solaris 10 release to use full propagation. Normally, the procedure would only 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 cannot support incremental propagation, so the slave must 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
- Online help URL = http://download.oracle.com/docs/cd/E23824_01/html/821-1456/aadmin-23.html

Note – Adjust the URL to point to the section, as described in "Online Help URL in the Graphical Kerberos Administration Tool" on page 347.

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 399.

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 name service.

```
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 name service.

```
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
```

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 415 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 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 `libdefaults` section of the `krb5.conf` file for authentication to succeed. See "Synchronizing Clocks Between KDCs and Kerberos Clients" on page 398 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 **Become superuser.**

2 **On the KDC master server, run the `kproplog` command.**

```
kdc1 # /usr/sbin/kproplog -h
```

3 **On a KDC slave server, run the `kproplog` command.**

```
kdc2 # /usr/sbin/kproplog -h
```

4 **Check that the last serial # and the last timestamp values match.**

Example 21–18 Verifying That the KDC Servers Are Synchronized

The following is a sample of results from running the `kproplog` command on the master KDC server.

```
kdc1 # /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: 2500
    First serial #: 137966
    Last serial #: 140465
    First time stamp: Fri Nov 28 00:59:27 2004
    Last time stamp: Fri Nov 28 01:06:13 2004
```

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 an administrator or assume a role or user name that has been assigned to the Kerberos Server Management profile.**

For more information, see "How to Obtain Administrative Rights" on page 160.

2 **Become superuser on the master KDC.**

3 **(Optional) Back up the database by using the `kdb5_util` command.**

```
# /usr/sbin/kdb5_util dump /var/krb5/slave_datatrans
```

4 **Propagate the database to a slave KDC by using the `kprop` command.**

```
# /usr/lib/krb5/kprop -f /var/krb5/slave_datatrans slave-KDC
```

Example 21–19 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

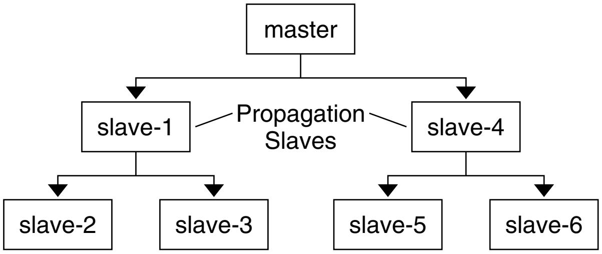
In most cases, the master KDC is used exclusively to propagate its Kerberos database to the slave KDCs. However, if your site has many slave KDCs, you might consider load-sharing the propagation process, known as *parallel propagation*.

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 21–2), 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.

FIGURE 21–2 Example of Parallel Propagation Configuration



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.

   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.

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.

EXAMPLE 21–20 Setting Up Parallel Propagation

Using the example in Figure 21–2, 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.

▼ How to Employ a New Master Key

1. **Become an administrator or assume a role or user name that has been assigned to the Kerberos Server Management profile.**

   For more information, see "How to Obtain Administrative Rights" on page 160.

2. **Create a new master key.**

   This command adds a new, randomly generated master key. The `-s` option requests that the new master key be stored in the default keytab.

   ```
   # kdb5_util add_mkey -s

   Creating new master key for master key principal 'K/M@EXAMPLE.COM'
   You will be prompted for a new 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>
   ```

3. **Verify that the new master key exists.**

   ```
   # kdb5_util list_mkeys
   Master keys for Principal: K/M@EXAMPLE.COM
   KNVO: 2, Enctype: AES-128 CTS mode with 96-bit SHA-1 HMAC, No activate time set
   KNVO: 1, Enctype: DES cbc mode with RSA-MD5, Active on: Wed Dec 31 18:00:00 CST 2001 *
   ```

   The asterisk in this output identifies the currently active master key.

4. **Set a time for the newly created master key to become active.**

```
# date
Fri Jul 1 17:57:00 CDT 2011
# kdb5_util use_mkey 2 'now+2days'
# kdb5_util list_mkeys
Master keys for Principal: K/M@EXAMPLE.COM
KNVO: 2, Enctype: AES-128 CTS mode with 96-bit SHA-1 HMAC, Active on: Sun Jul 03 17:57:15 CDT 2011
KNVO: 1, Enctype: DES cbc mode with RSA-MD5, Active on: Wed Dec 31 18:00:00 CST 2001 *
```

   In this example, the date is set to two days in the future to allow time for the new master key to propagate to all of the KDCs. Adjust the date as appropriate for your environment.

5. **(Optional) After creating a new principal, verify that the new master key is being used.**

   ```
   # kadmin.local -q 'getprinc jimf' |egrep 'Principal|MKey'
   Authenticating as principal root/admin@EXAMPLE.COM with password.
   Principal: jimf@EXAMPLE.COM
   MKey: vno 2
   ```

   In this example, `MKey: vno 2` indicates that the principal's secret key is protected by newly created master key, 2.

6 **Re-encrypt the user principal secret keys with the new master key.**

If you add a pattern argument to the end of the command, the principals that match the pattern will be updated. Add the `-n` option to this command syntax to identify which principals will be updated.

```
# kdb5_util update_princ_encryption -f -v
Principals whose keys WOULD BE re-encrypted to master key vno 2:
updating: host/kdc1.example.com@EXAMPLE.COM
skipping:   jimf@EXAMPLE.COM
updating: kadmin/changepw@EXAMPLE.COM
updating: kadmin/history@EXAMPLE.COM
updating: kdc/admin@EXAMPLE.COM
updating: host/kdc2.example.com@EXAMPLE.COM
6 principals processed: 5 updated, 1 already current
```

7 **Purge the old master key.**

After a master key is no longer used to protect any principal secret keys, it can be purged from the master key principal. This command will not purge the key if the key is still being used by any principals. Add the `-n` option to this command to verify that the correct master key will be purged.

```
# kdb5_util purge_mkeys -f -v
Purging the follwing master key(s) from K/M@EXAMPLE.COM:
KNVO: 1
1 key(s) purged.
```

8 **Verify that the old master key has been purged.**

```
# kdb5_util list_mkeys
Master keys for Principal: K/M@EXAMPLE.COM
KNVO: 2, Enctype: AES-128 CTS mode with 96-bit SHA-1 HMAC, Active on: Sun Jul 03 17:57:15 CDT 2011 *
```

9 **Update the stash file.**

```
# kdb5_util stash
Using existing stashed keys to update stash file.
```

10 **Verify that the stash file has been updated.**

```
# klist -kt /var/krb5/.k5.EXAMPLE.COM
Keytab name: FILE:.k5.EXAMPLE.COM
KVNO Timestamp              Principal
---- ---------------- ---------------------------------------------------------
   2 05/07/2011 15:08 K/M@EXAMPLE.COM
```

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.

TABLE 21–3 Configuring KDC Servers to Use LDAP (Task Map)

Task	Description	For Instructions
Configure a master KDC.	Configures and builds the master KDC server and database for a realm using a manual process and using LDAP for the KDC.	"How to Configure a KDC to Use an LDAP Data Server" on page 358
Mix Kerberos principal attributes with non-Kerberos object class types.	Allows information stored with the Kerberos records to be shared with other LDAP databases.	"How to Mix Kerberos Principal Attributes in a Non-Kerberos Object Class Type" on page 420
Destroy a realm.	Removes all of the data associated with a realm.	"How to Destroy a Realm on an LDAP Directory Server" on page 421

▼ 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`, and `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.

```
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.

```
# 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 21–4 Increasing Security on Kerberos Servers (Task Map)

Task	Description	For Instructions
Enable access using Kerberos authentication.	Restrict network access to a server to allow Kerberos authentication only	"How to Enable Only Kerberized Applications" on page 421
Restrict access to the KDC servers.	Increases the security of the KDC servers and their data.	"How to Restrict Access to KDC Servers" on page 422
Increase password security by using a dictionary file.	Increases the security of any new passwords by checking the new password against a dictionary.	"How to Use a Dictionary File to Increase Password Security" on page 423

▼ 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 **Become superuser.**

2 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"
```

3 (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"
```

4 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 Become superuser.

2 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 tli (ticlts, ticotsord, 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/rexec
# svcadm disable network/shell
# svcadm disable network/talk
# svcadm disable network/tname
# svcadm disable network/uucp
# svcadm disable network/rpc_100068_2-5/rpc_udp
```

3 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.

4 **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
                kadmind_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
```

CHAPTER 22

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 Tool Error Messages" on page 425
- "Common Kerberos Error Messages (A-M)" on page 426
- "Common Kerberos Error Messages (N-Z)" on page 435
- "Problems With the Format of the `krb5.conf` File" on page 440
- "Problems Propagating the Kerberos Database" on page 440
- "Problems Mounting a Kerberized NFS File System" on page 441
- "Problems Authenticating as the `root` User" on page 441
- "Observing Mapping From GSS Credentials to UNIX Credentials" on page 442

Kerberos Error Messages

This section provides information about Kerberos error messages, including why each error occurs and a way to fix it.

SEAM 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 (`kadm5.acl`). 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 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: host is '`*hostname*`'`

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 a kadmin KDC entry in krb5.conf(4) or DNS Service Location records for realm '`*realmname*`'`

`Cannot find a kpassword KDC entry in krb5.conf(4) or DNS Service Location records for realm '`*realmname*`'`

`Cannot find a master KDC entry in krb5.conf(4) or DNS Service Location records for realm '`*realmname*`'`

`Cannot find any KDC entries in krb5.conf(4) or DNS Service Location records for realm '`*realmname*`'`

Cause: Either the `krb5.conf` file or the DNS server record are incorrectly configured.

Solution: Make sure that the Kerberos configuration file (`/etc/krb5/krb5.conf`) or that the DNS server records for the KDC are configured properly.

`Cannot find address for '`*hostname*`': '`*error-string*`'`

Cause: No address was found in the DNS records for the given hostname.

Solution: Fix the host record in DNS or correct the error in the DNS lookup process.

`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 resolve network address for KDCs '`*hostname*`' discovered via DNS Service Location records for realm '`*realm-name*`'`

`Cannot resolve network address for KDCs '`*hostname*`' specified in krb5.conf(4) for realm '`*realm-name*`'`

Cause: Either the `krb5.conf` file or the DNS server record is incorrectly configured.

Solution: Make sure that the Kerberos configuration file (`/etc/krb5/krb5.conf`) and the DNS server records for the KDC are configured properly.

`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 '`*principal*`' not found in Kerberos database`

Cause: The principal is missing from the Kerberos database.

Solution: Add the client principal to the Kerberos database.

`Client '`*`principal`*`' pre-authentication failed`

Cause: Authentication failed for the principal.

Solution: Make sure that the user is using the correct password.

`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: '`*`client-principal`*`' requesting ticket '`*`service-principal`*`'`

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`.

`Clock skew too great: '`*`client`*`' requesting ticket '`*`service-principal`*`' from KDC '`*`KDC-hostname`*`' (`*`KDC-time`*`). Skew is `*`value`*

`Clock skew too great: '`*`client`*`' AP request with ticket for '`*`service-principal`*`'. Skew is `*`value`*` (allowable `*`value`*`)`

Cause: The difference between the time reported on the client and the KDC server or application server is too large.

Solution: Configure the Network Time Protocol (NTP) to keep the clocks synchronized. See "Synchronizing Clocks Between KDCs and Kerberos Clients" on page 398 for more information.

`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.

`Decrypt integrity check failed for client 'principal' and server 'hostname'`

Cause: You might have an invalid ticket.

Solution: Make sure that your credentials are valid. Destroy your tickets with the `kdestroy` command, and create new tickets with the `kinit` command.

`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 find realm for` *principal* `in keytab`

Cause: The realm name included in the *principal* does not match the realm name in the principal stored in the keytab file.

Solution: Make sure that the principals are using the correct realm.

`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 for '`*hostname*`': '`*error-string*`'`

Cause: The Kerberos client cannot find the fully qualified host name for the server.

Solution: Make sure that the server host name is defined in DNS and that the hostname-to-address and address-to-hostname 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 after the Solaris 10 8/07 release. Clients can request encryption types that might not be supported by a KDC running an older version of the 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 expectation: KDC not found. Probably got an unexpected realm referral`

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 RFC4120, that the request you are sending is a Kerberos V5 request, and 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 by 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 table file '`*filename*`' not found`

Cause: The named key table file does not exist.

Solution: Create the key table file.

`Key version` *number* `is not available for principal` *principal*

Cause: The key version of the keys does not match the version for the keys on the application server.

Solution: Check the version of the keys on the application server using the `klist -k` command.

`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 getting initial creds: '`*client-principal*`' requesting ticket '`*service-principal*`'. Max loops is `*value*`. Make sure a KDC is available.`

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 ('`*filename*`') matches desired name` *principal*

Cause: An error occurred during an attempt 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: Requested principal is '`*service-principal*`' and TGT principal is '`*TGT-principal*`'`

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.

`Service key` *service-principal* `not available`

Cause: The named service principal is not in the keytab file on the application server.

Solution: Make sure that the service principal matches or is included in the keytab file on the application server.

`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.

`Server` *service-principal* `not found in Kerberos database`

Cause: The service principal is not correct or is missing from the principal database.

Solution: Make sure that the service principal is correct and that it is in the database.

`Target name principal '`*principal*`' does not match` *service-principal*

Cause: The service principal that is being used does not match the service principal that the application server is using.

Solution: On the application server, make sure that the service principal is included in the keytab file. For the client, make sure that the correct service principal is being used.

`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: '`*client-principal*`' requesting ticket '`*service-principal*`' from '`*kdc-hostname*`' (`*time*`). TGT start time is` *time*`.`

Cause: The postdated ticket is not yet valid.

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.

`Unknown encryption type:` *name*

Cause: The encryption type that is included with the credential cannot be used.

Solution: Determine which encryption types the client is using with the `klist -e` command. Make sure that the application server supports at least one of the encryption types.

`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.

How to Identify Problems With Key Version Numbers

Sometimes, the key version number (KVNO) used by the KDC and the service principal keys stored in `/etc/krb5/krb5.keytab` for services hosted on the system do not match. The KVNO can get out of synchronization when a new set of keys are created on the KDC without updating the keytable file with the new keys. This problem can be diagnosed by using the following procedure.

1 **List the `keytab` entries.**

Note that the KVNO for each principal is included in the list.

```
# klist -k
Keytab name: FILE:/etc/krb5/krb5.keytab
KVNO Principal
---- --------------------------------------------------------------------------
```

```
   2 host/denver.example.com@EXAMPLE.COM
   2 host/denver.example.com@EXAMPLE.COM
   2 host/denver.example.com@EXAMPLE.COM
   2 nfs/denver.example.com@EXAMPLE.COM
   2 nfs/denver.example.com@EXAMPLE.COM
   2 nfs/denver.example.com@EXAMPLE.COM
   2 nfs/denver.example.com@EXAMPLE.COM
```

2 **Acquire an initial credential by using the `host` key.**

```
# kinit -k
```

3 **Determine the KVNO that is used by the KDC.**

```
# kvno nfs/denver.example.com
nfs/denver.example.com@EXAMPLE.COM: kvno = 3
```

Note that the KVNO listed here is 3 instead of 2.

Problems With the Format of the krb5.conf File

If the `krb5.conf` file is not formatted properly, then the following error message might be displayed in a terminal window or recorded in the log file:

```
Improper format of Kerberos configuration file while initializing krb5 library
```

If there is a problem with the format of the `krb5.conf` file, then the associated services could be vulnerable to attack. You must 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[1] :

  ```
  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 the root User

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/resolv.conf` file to make sure that the system is correctly set up as a DNS client.

[1] Ken Hornstein, "Kerberos FAQ," [http://www.cmf.nrl.navy.mil/CCS/people/kenh/kerberos-faq.html#kerbdns], accessed 10 March 2010.

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.

Using DTrace With the Kerberos Service

In this example, you would like to know if pre-authentication is required by a KDC, and if so what pre-authentication types are supported. First, as a privileged user, create a D program source file, like the following:

```
# cat kerberos_preauth.d
kerberos$target:::krb_error-read
{
    self->preauth = args[1]->kerror_error_code ==
        "KDC_ERR_PREAUTH_REQUIRED(25)" ?  "required" : "not required";

    printf(" - Preauthentication is %s for this KDC.\n", self->preauth);
}

kerberos$target:::krb_error-read
/ self->preauth == "required" /
{
    printf(" - This KDC supports the following preauth types: %s.",
        args[1]->kerror_e_data);
}
```

Next, compile the preauth.d source file to get your answer.

```
# dtrace -qs kerberos_preauth.d -c "kinit -k"
 - Preauthentication is required for this KDC.
 - This KDC supports the following preauth types: ENC_TIMESTAMP(2)
FX_FAST(136) PK_ETYPE_INFO2(19) SAM_RESPONSE(13) FX_COOKIE(133).
```

For more information about the various pre-authentication types see RFC 4120.

CHAPTER 23

Administering Kerberos Principals and Policies (Tasks)

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 19, "Introduction to the Kerberos Service," and Chapter 20, "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 443
- "SEAM Tool" on page 444
- "Administering Kerberos Principals" on page 448
- "Administering Kerberos Policies" on page 461
- "SEAM Tool Reference" on page 469
- "Administering Keytab Files" on page 473

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 Oracle Solaris release provides the SEAM Tool, gkadmin, which is an interactive graphical user interface (GUI) that provides essentially the same capabilities as the kadmin command. See "SEAM Tool" on page 444 for more information.

SEAM Tool

The SEAM Tool (gkadmin) 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 473.

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 448
- "Administering Kerberos Policies (Task Map)" on page 461

Also, go to "SEAM Tool Panel Descriptions" on page 469 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 23–1 Command-Line Equivalents of the SEAM Tool

SEAM Tool Procedure	Equivalent `kadmin` Command
View the list of principals.	`list_principals` or `get_principals`
View a principal's attributes.	`get_principal`
Create a new principal.	`add_principal`
Duplicate a principal.	No command-line equivalent
Modify a principal.	`modify_principal` or `change_password`
Delete a principal.	`delete_principal`
Set up defaults for creating new principals.	No command-line equivalent
View the list of policies.	`list_policies` or `get_policies`
View a policy's attributes.	`get_policy`
Create a new policy.	`add_policy`
Duplicate a policy.	No command-line equivalent
Modify a policy.	`modify_policy`
Delete a policy.	`delete_policy`

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.

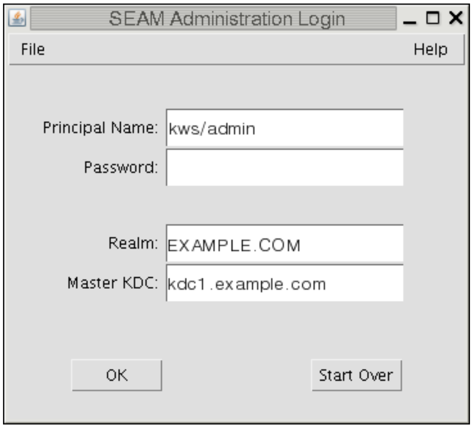
Caching the lists still enables you to use the list panels to select principals and policies, so it doesn't affect how you use the SEAM Tool as the first option does. Also, even though caching doesn't enable you to see the changes of other users, you can still see the latest list information based on 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.



Principal Name: kws/admin

Password:

Realm: EXAMPLE.COM

Master KDC: kdc1.example.com

2 **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 472.

---

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)

Task	Description	For Instructions
View the list of principals.	View the list of principals by clicking the Principals tab.	"How to View the List of Kerberos Principals" on page 449
View a principal's attributes.	View a principal's attributes by selecting the Principal in the Principal List, then clicking the Modify button.	"How to View a Kerberos Principal's Attributes" on page 451
Create a new principal.	Create a new principal by clicking the Create New button in the Principal List panel.	"How to Create a New Kerberos Principal" on page 453
Duplicate a principal.	Duplicate a principal by selecting the principal to duplicate in the Principal List, then clicking the Duplicate button.	"How to Duplicate a Kerberos Principal" on page 456
Modify a principal.	Modify a principal by selecting the principal to modify in the Principal List, then clicking the Modify button. Note that 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.	"How to Modify a Kerberos Principal" on page 456
Delete a principal.	Delete a principal by selecting the principal to delete in the Principal List, then clicking the Delete button.	"How to Delete a Kerberos Principal" on page 457
Set up defaults for creating new principals.	Set up defaults for creating new principals by choosing Properties from the Edit menu.	"How to Set Up Defaults for Creating New Kerberos Principals" on page 458
Modify the Kerberos administration privileges (`kadm5.acl` file).	*Command-line only.* The Kerberos administration privileges determine what operations a principal can perform on the Kerberos database, such as add and modify. You need to edit the `/etc/krb5/kadm5.acl` file to modify the Kerberos administration privileges for each principal.	"How to Modify the Kerberos Administration Privileges" on page 459

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 `princnames` 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.

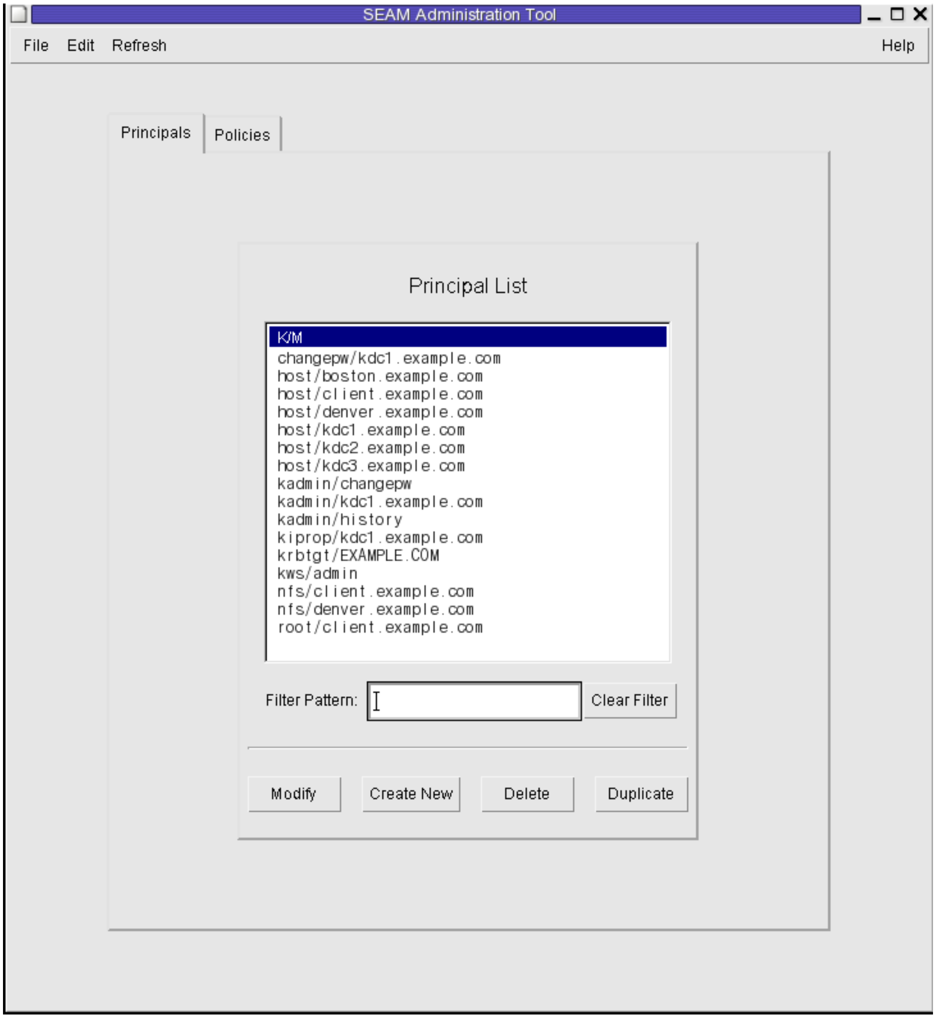
1 **If necessary, start the SEAM Tool.**

See "How to Start the SEAM Tool" on page 447 for more information.

```
$ /usr/sbin/gkadmin
```

2 **Click the Principals tab.**

The list of principals is displayed.



```
K/M
changepw/kdc1.example.com
host/boston.example.com
host/client.example.com
host/denver.example.com
host/kdc1.example.com
host/kdc2.example.com
host/kdc3.example.com
kadmin/changepw
kadmin/kdc1.example.com
kadmin/history
kiprop/kdc1.example.com
krbtgt/EXAMPLE.COM
kws/admin
nfs/client.example.com
nfs/denver.example.com
root/client.example.com
```

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 23–1 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/kdc1.example.con@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 447 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.

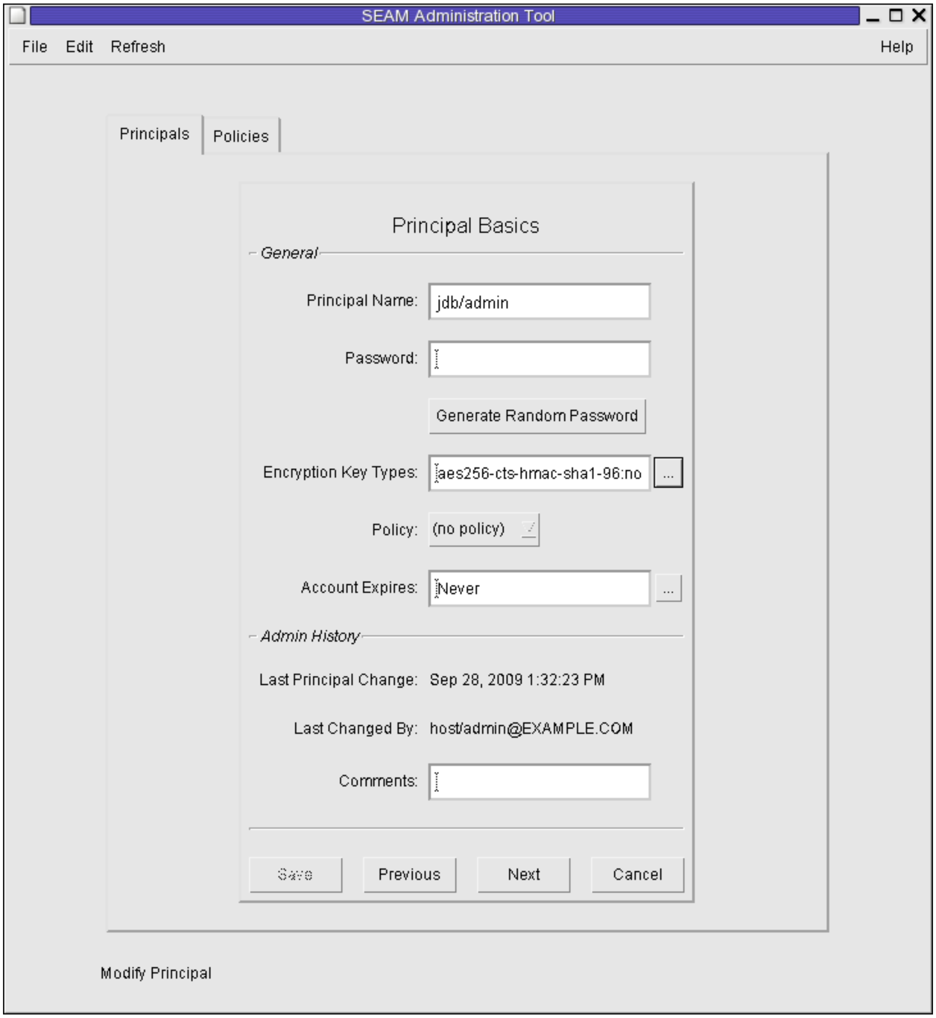
4. **Continue to click Next to view all the principal's attributes.**

   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 469.

5. **When you are finished viewing, click Cancel.**

Example 23–2 Viewing a Kerberos Principal's Attributes

The following example shows the first window when you are viewing the `jdb/admin` principal.



Principal Basics

General

Principal Name: jdb/admin

Password:

Encryption Key Types: aes256-cts-hmac-sha1-96:no

Policy: (no policy)

Account Expires: Never

Admin History

Last Principal Change: Sep 28, 2009 1:32:23 PM

Last Changed By: host/admin@EXAMPLE.COM

Comments:

Modify Principal

Example 23–3 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: [never]
Last password change: [never]

Password expiration date: Wed Apr 14 11:53:10 PDT 2011
Maximum ticket life: 1 day 16:00:00
Maximum renewable life: 1 day 16:00:00
```

```
Last modified: Mon Sep 28 13:32:23 PST 2009 (host/admin@EXAMPLE.COM)
Last successful authentication: [never]
Last failed authentication: [never]
Failed password attempts: 0
Number of keys: 1
Key: vno 1, AES-256 CTS mode with 96-bit SHA-1 HMAC, no salt
Key: vno 1, AES-128 CTS mode with 96-bit SHA-1 HMAC, no salt
Key: vno 1, Triple DES with HMAC/sha1, no salt
Key: vno 1, ArcFour with HMAC/md5, no salt
Key: vno 1, DES cbc mode with RSA-MD5, no salt
Attributes: REQUIRES_HW_AUTH
Policy: [none]
kadmin: quit
```

▼ 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 447 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 465.

```
$ /usr/sbin/gkadmin
```

2 **Click the Principals tab.**

3 **Click New.**

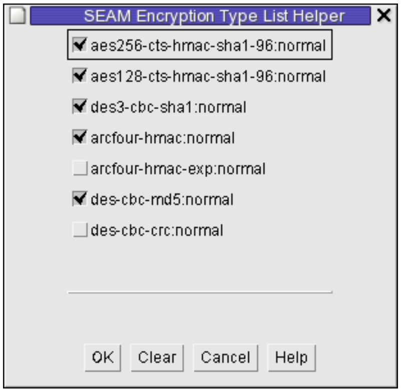
The Principal Basics panel that contains some attributes for a principal is displayed.

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.



☑ aes256-cts-hmac-sha1-96:normal
☑ aes128-cts-hmac-sha1-96:normal
☑ des3-cbc-sha1:normal
☑ arcfour-hmac:normal
□ arcfour-hmac-exp:normal
☑ des-cbc-md5:normal
□ des-cbc-crc:normal

6. **Specify the policy for the principal.**

7. **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 469.

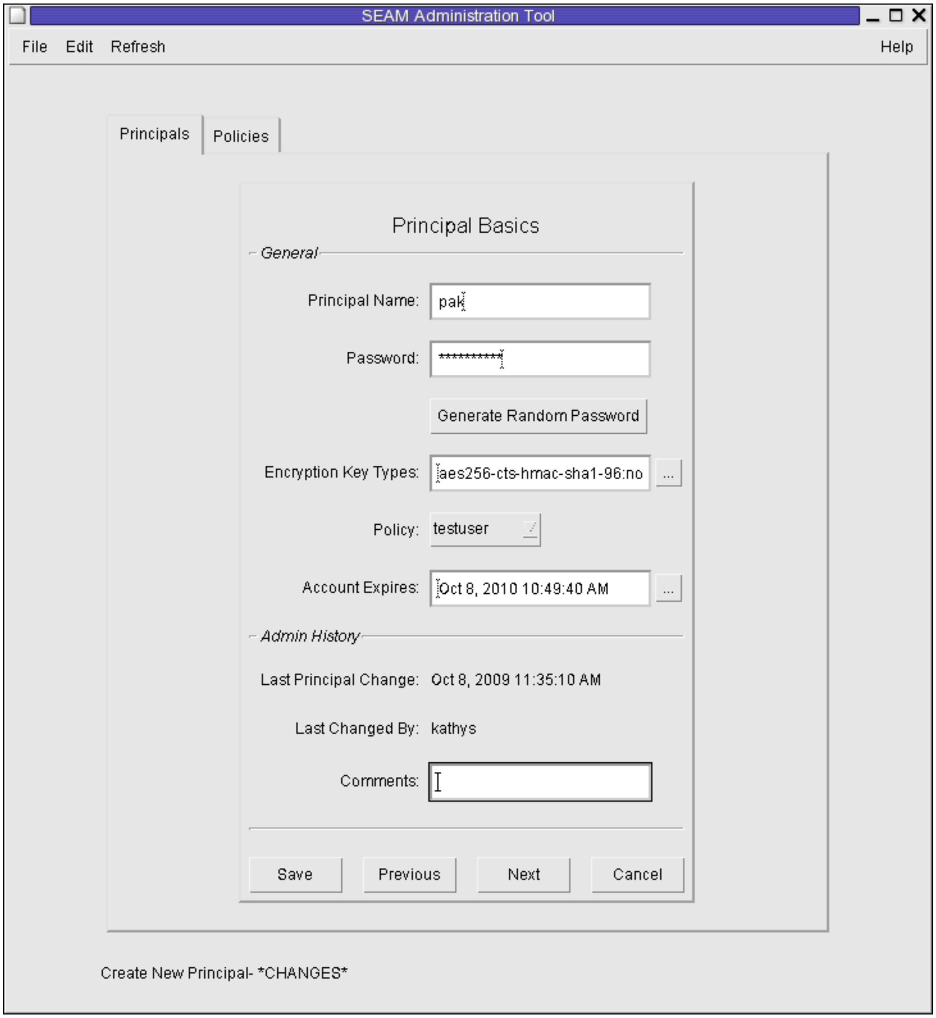
8. **Click Save to save the principal, or click Done on the last panel.**

9. **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 459 for more details.

Example 23–4 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`.



Principal Basics

General

Principal Name: pak

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:

Create New Principal- *CHANGES*

Example 23–5 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`.

```
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 447 for more information.

```
$ /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 469.

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 459 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 447 for more information.

```
$ /usr/sbin/gkadmin
```

2 **Click the Principals 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 469.

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 459 for more details.

Example 23–6 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 447 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 459 for more details.

Example 23–7 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.

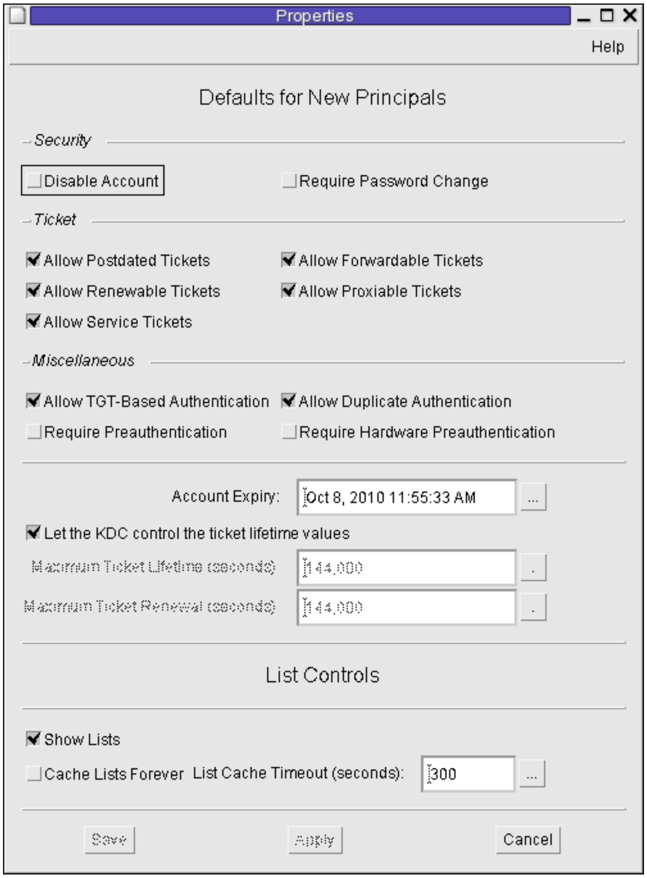
1 **If necessary, start the SEAM Tool.**

See "How to Start the SEAM Tool" on page 447 for more information.

```
$ /usr/sbin/gkadmin
```

2 **Choose Properties from the Edit Menu.**

The Properties window is displayed.



Defaults for New Principals

Security

□ Disable Account
□ Require Password Change

Ticket

☑ Allow Postdated Tickets
☑ Allow Forwardable Tickets
☑ Allow Renewable Tickets
☑ Allow Proxiable Tickets
☑ Allow Service Tickets

Miscellaneous

☑ Allow TGT-Based Authentication
☑ Allow Duplicate Authentication
□ Require Preauthentication
□ Require Hardware Preauthentication

Account Expiry: Oct 8, 2010 11:55:33 AM

☑ Let the KDC control the ticket lifetime values

Maximum Ticket Lifetime (seconds): 144,000

Maximum Ticket Renewal (seconds): 144,000

List Controls

☑ Show Lists

□ Cache Lists Forever List Cache Timeout (seconds): 300

3 **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 polices.
`c`	[Dis]allows the changing of passwords for principals.
`i`	[Dis]allows inquiries to the Kerberos database.
`l`	[Dis]allows the listing of principals or policies in the Kerberos database.
`x` or `*`	Allows all privileges (`admcil`).

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 23–8 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)

Task	Description	For Instructions
View the list of policies.	View the list of policies by clicking the Policies tab.	"How to View the List of Kerberos Policies" on page 461
View a policy's attributes.	View a policy's attributes by selecting the policy in the Policy List, then clicking the Modify button.	"How to View a Kerberos Policy's Attributes" on page 463
Create a new policy.	Create a new policy by clicking the Create New button in the Policy List panel.	"How to Create a New Kerberos Policy" on page 465
Duplicate a policy.	Duplicate a policy by selecting the policy to duplicate in the Policy List, then clicking the Duplicate button.	"How to Duplicate a Kerberos Policy" on page 467
Modify a policy.	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.	"How to Modify a Kerberos Policy" on page 467
Delete a policy.	Delete a policy by selecting the policy to delete in the Policy List, then clicking the Delete button.	"How to Delete a Kerberos Policy" on page 468

▼ How to View the List of Kerberos Policies

An example of the command-line equivalent follows this procedure.

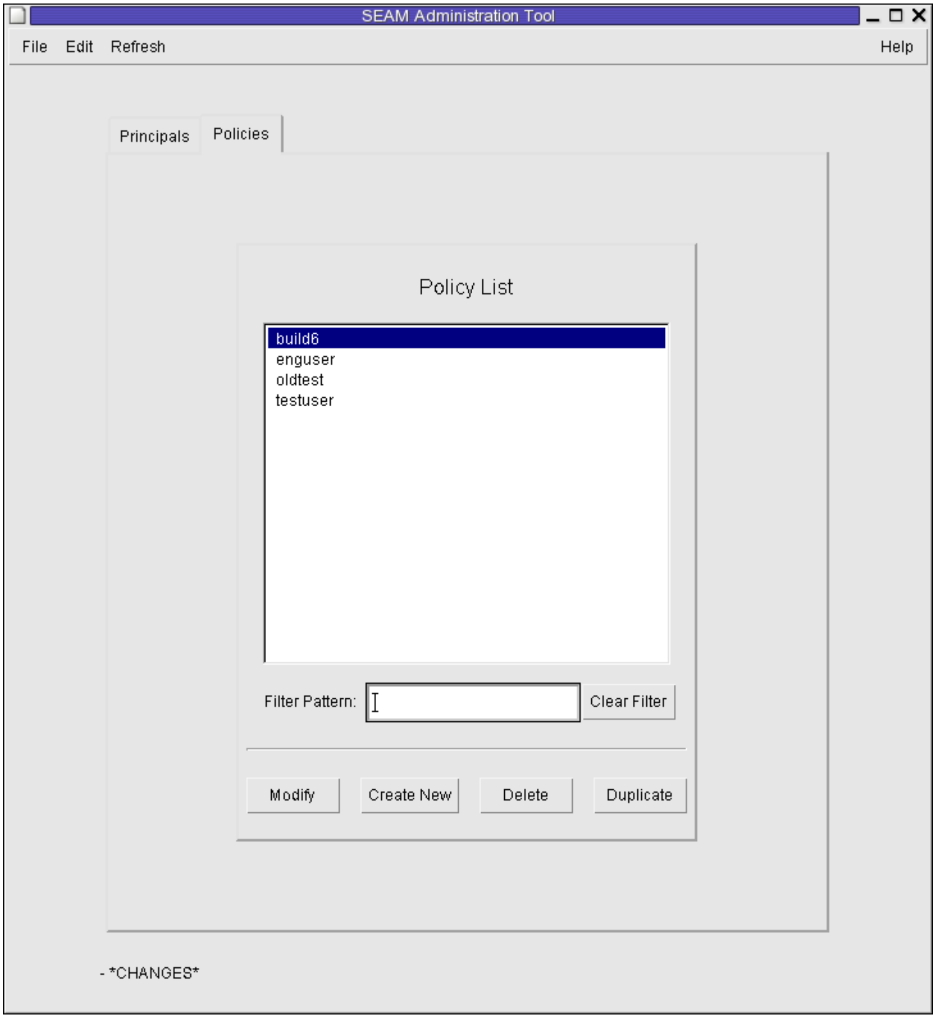
1 **If necessary, start the SEAM Tool.**

See "How to Start the SEAM Tool" on page 447 for more information.

```
$ /usr/sbin/gkadmin
```

2 Click the Policies tab.

The list of policies is displayed.



build6
enguser
oldtest
testuser

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 23–9 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
enguser
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 447 for more information.

   ```
   $ /usr/sbin/gkadmin
   ```

2. **Click the Policies tab.**

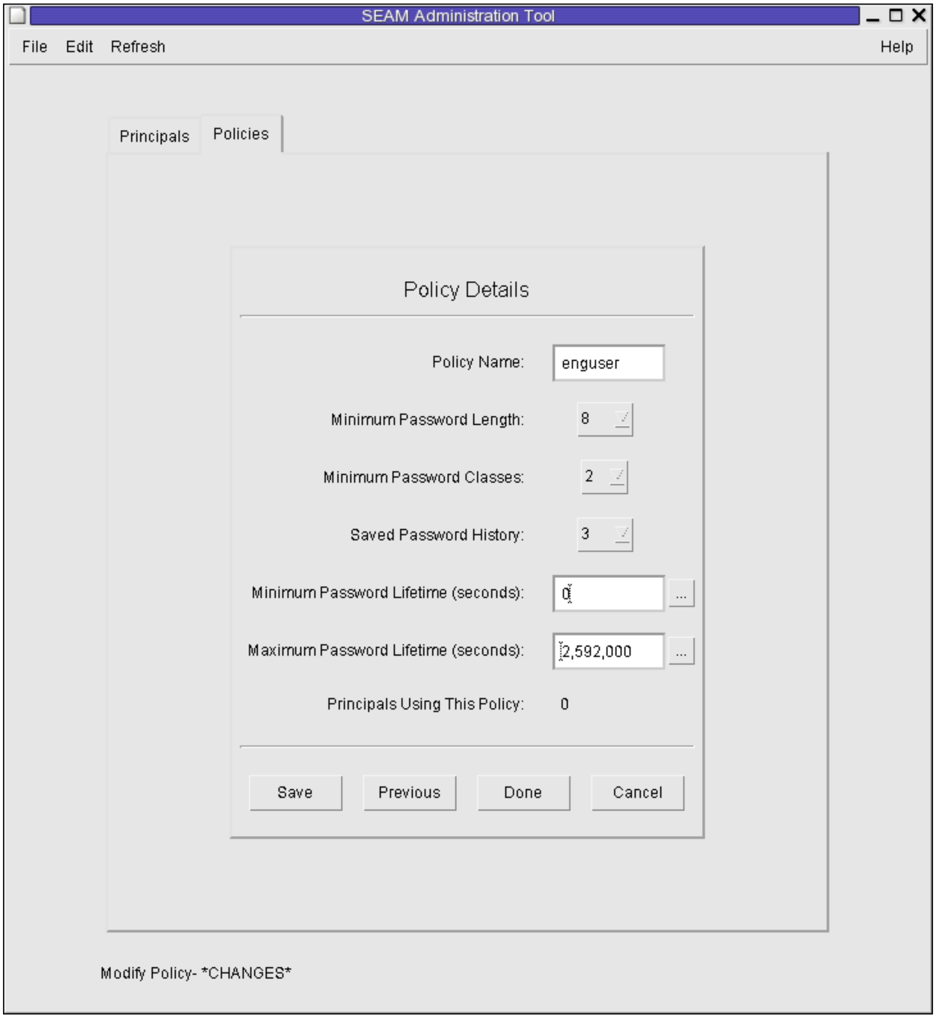
3. **Select the policy in the list that you want to view, then click Modify.**

   The Policy Details panel is displayed.

4. **When you are finished viewing, click Cancel.**

Example 23–10 Viewing a Kerberos Policy's Attributes

The following example shows the Policy Details panel when you are viewing the `test` policy.



Policy Details

Policy Name: enguser

Minimum Password Length: 8

Minimum Password Classes: 2

Saved Password History: 3

Minimum Password Lifetime (seconds): 0

Maximum Password Lifetime (seconds): 2,592,000

Principals Using This Policy: 0

Modify Policy- *CHANGES*

Example 23–11 Viewing a Kerberos Policy's Attributes (Command Line)

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
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 447 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.

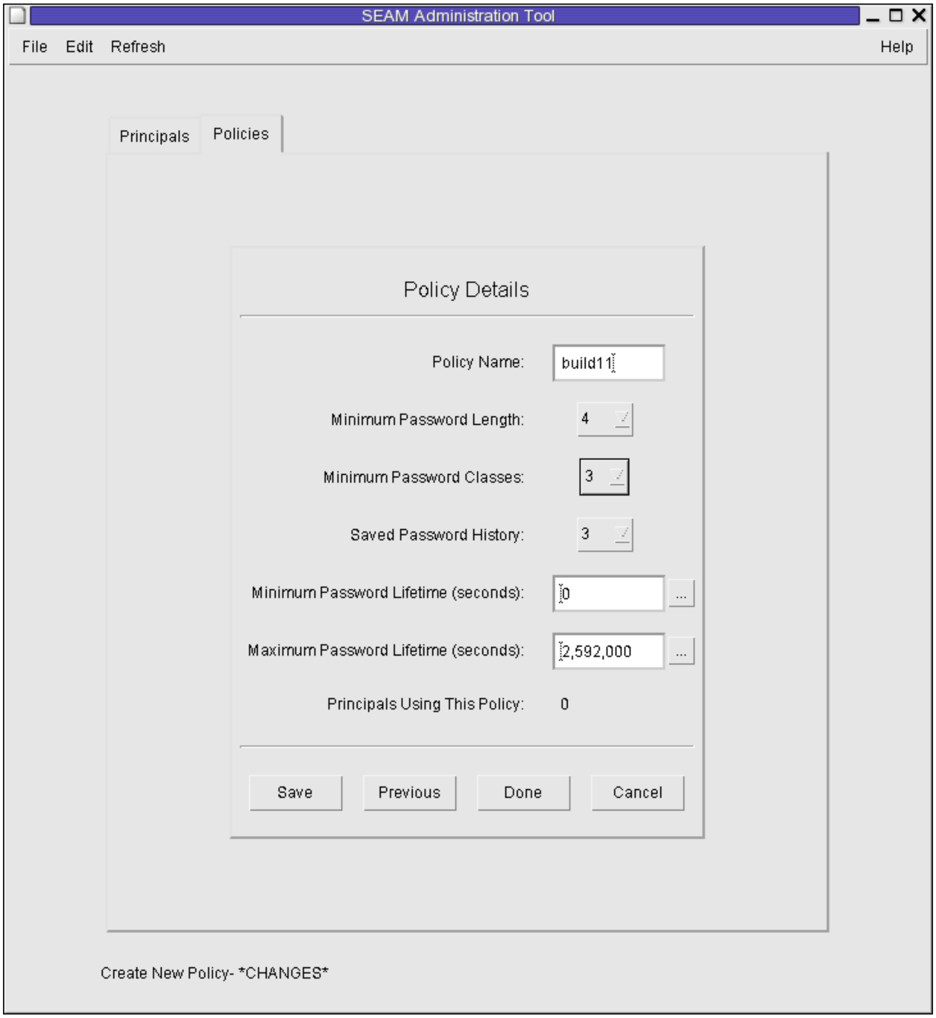
5. **Specify values for the policy's attributes.**

   Choose Context-Sensitive Help from the Help menu for information about the various attributes in this window. Or, go to Table 23–5 for all the policy attribute descriptions.

6. **Click Save to save the policy, or click Done.**

Example 23–12 Creating a New Kerberos Policy

In the following example, a new policy called `build11` is created. The Minimum Password Classes is set to 3.



Policy Name: build11

Minimum Password Length: 4

Minimum Password Classes: 3

Saved Password History: 3

Minimum Password Lifetime (seconds): 0

Maximum Password Lifetime (seconds): 2,592,000

Principals Using This Policy: 0

Create New Policy- *CHANGES*

Example 23–13 Creating a New Kerberos Policy (Command Line)

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 447 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 23–5 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 447 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 23–5 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 23–14 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 447 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 23–15 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 23–2 Attributes for the Principal Basics Panel of the SEAM Tool

Attribute	Description
Principal Name	The name of the principal (which is the *primary/instance* part of a fully qualified principal name). A principal is a unique identity to which the KDC can assign tickets. If you are modifying a principal, you cannot edit its name.
Password	The password for the principal. You can use the Generate Random Password button to create a random password for the principal.
Policy	A menu of available policies for the principal.
Account Expires	The date and time on which the principal's account expires. When the account expires, the principal can no longer get a ticket-granting ticket (TGT) and might be unable to log in.
Last Principal Change	The date on which information for the principal was last modified. (Read only)
Last Changed By	The name of the principal that last modified the account for this principal. (Read only)
Comments	Comments that are related to the principal (for example, “Temporary Account”).

TABLE 23–3 Attributes for the Principal Details Panel of the SEAM Tool

Attribute	Description
Last Success	The date and time when the principal last logged in successfully. (Read only)
Last Failure	The date and time when the last login failure for the principal occurred. (Read only)
Failure Count	The number of times a login failure has occurred for the principal. (Read only)
Last Password Change	The date and time when the principal's password was last changed. (Read only)
Password Expires	The date and time when the principal's current password expires.
Key Version	The key version number for the principal. This attribute is normally changed only when a password has been compromised.

TABLE 23–3 Attributes for the Principal Details Panel of the SEAM Tool *(Continued)*

Attribute	Description
Maximum Lifetime (seconds)	The maximum length of time for which a ticket can be granted for the principal (without renewal).
Maximum Renewal (seconds)	The maximum length of time for which an existing ticket can be renewed for the principal.

TABLE 23–4 Attributes of the Principal Flags Panel of the SEAM Tool

Attribute (Radio Buttons)	Description
Disable Account	When checked, prevents the principal from logging in. This attribute provides an easy way to temporarily freeze a principal account.
Require Password Change	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.
Allow Postdated Tickets	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.
Allow Forwardable Tickets	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.
Allow Renewable Tickets	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.
Allow Proxiable Tickets	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).
Allow Service Tickets	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.

TABLE 23–4 Attributes of the Principal Flags Panel of the SEAM Tool *(Continued)*

Attribute (Radio Buttons)	Description
Allow TGT-Based Authentication	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.
Allow Duplicate Authentication	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.
Required Preauthentication	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.
Required Hardware Authentication	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.

TABLE 23–5 Attributes for the Policy Basics Pane of the SEAM Tool

Attribute	Description
Policy Name	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.
Minimum Password Length	The minimum length for the principal's password.
Minimum Password Classes	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.
Saved Password History	The number of previous passwords that have been used by the principal, and a list of the previous passwords that cannot be reused.
Minimum Password Lifetime (seconds)	The minimum length of time that the password must be used before it can be changed.

TABLE 23–5 Attributes for the Policy Basics Pane of the SEAM Tool *(Continued)*

Attribute	Description
Maximum Password Lifetime (seconds)	The maximum length of time that the password can be used before it must be changed.
Principals Using This Policy	The number of principals to which this policy currently applies. (Read only)

Using the SEAM Tool With Limited Kerberos Administration Privileges

All capabilities of the SEAM 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 23–6 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 polices 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 in to 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 459.

TABLE 23–6 Using the SEAM Tool With Limited Kerberos Administration Privileges

Disallowed Privilege	How the SEAM Tool Changes
`a` (add)	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.
`d` (delete)	The Delete button is unavailable in the Principal List and Policy List panels. Without the delete privilege, you cannot delete principals or policies.

TABLE 23–6 Using the SEAM Tool With Limited Kerberos Administration Privileges *(Continued)*

Disallowed Privilege	How the SEAM Tool Changes
`m` (modify)	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.
`c` (change password)	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.
`i` (inquiry to database)	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.
`l` (list)	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.

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)

Task	Description	For Instructions
Add a service principal to a keytab file.	Use the ktadd command of kadmin to add a service principal to a keytab file.	"How to Add a Kerberos Service Principal to a Keytab File" on page 474
Remove a service principal from a keytab file.	Use the ktremove command of kadmin to remove a service from a keytab file.	"How to Remove a Service Principal From a Keytab File" on page 476
Display the keylist (list of principals) in a keytab file.	Use the ktutil command to display the keylist in a keytab file.	"How to Display the Keylist (Principals) in a Keytab File" on page 476
Temporarily disable authentication for a service on a host.	This procedure is a quick way to temporarily disable authentication for a service on a host without requiring kadmin privileges. Before you use ktutil 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.	"How to Temporarily Disable Authentication for a Service on a Host" on page 477

▼ 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 449 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 23–16 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/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.
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 476 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 23–17 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 23–18 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
```

10 **When you want to re-enable the service, copy the temporary (original) keytab file back to its original location.**

Example 23–19 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
```

CHAPTER 24

Using Kerberos Applications (Tasks)

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 481
- "Kerberos Password Management" on page 485
- "Kerberos User Commands" on page 490

For an overview of the Oracle Solaris Kerberos product, see Chapter 19, "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 328.

Do You Need to Worry About Tickets?

With any of the SEAM releases or the Oracle Solaris releases installed, Kerberos is built into the `login` command, and you will obtain tickets automatically when you log in. The Kerberized commands `rsh`, `rcp`, `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 behavior. See “Overview of Kerberized Commands” on page 490 and “Forwarding Kerberos Tickets” on page 493 for more information on forwarding tickets.

For information on ticket lifetimes, see “Ticket Lifetimes” on page 503.

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 24–1 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 493 and “Types of Tickets” on page 502.

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 502 describes the various attributes that a ticket can have.

EXAMPLE 24–2 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@EXAMPLE.COM

Valid starting                 Expires                 Service principal
09 Mar 04 15:09:51  09 Mar 04 21:09:51  nfs/EXAMPLE.COM@EXAMPLE.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).

```
% klist -f
Ticket cache: /tmp/krb5cc_74287
Default principal: david@EXAMPLE.COM
```

EXAMPLE 24–2 Viewing Kerberos Tickets *(Continued)*

```
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 ticket 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.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 "RrriotGrrrlsRrrule!"

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 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 461 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.

```
% 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 24–3 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 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.
```

EXAMPLE 24–3 Changing Your Password *(Continued)*

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

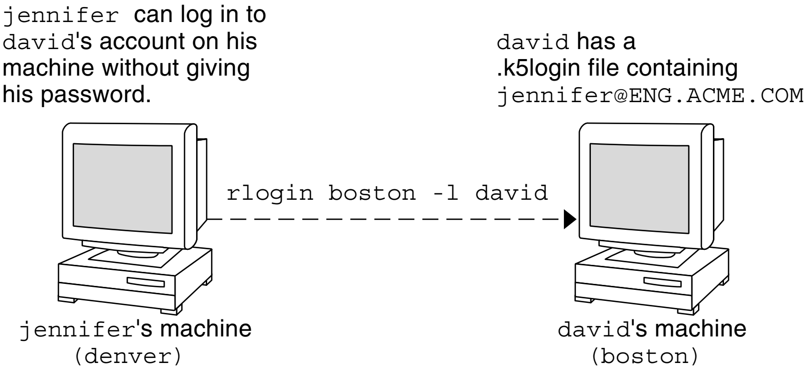
If you need to give someone access to log in to your account (as you), you can do so through Kerberos, without revealing your password, by putting a `.k5login` file in your home directory. A `.k5login` file is a list of one or more Kerberos principals corresponding to each person for whom you want to 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.

FIGURE 24–1 Using the .k5login File to Grant Access to Your Account


jennifer can log in to david's account on his machine without giving his password.
david has a .k5login file containing jennifer@ENG.ACME.COM
rlogin boston -l david
jennifer's machine (denver)
david's machine (boston)

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 482 for an example of using a forwardable ticket.

If you will be logging in to 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 24–4 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.

EXAMPLE 24–4 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`
- `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-reforwardable* 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 in to 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 493.

`-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 493.

`-k` *realm* Requests tickets for the remote host in the specified *realm*, instead of determining the realm itself using the `krb5.conf` file.

`-K` Uses your tickets to authenticate to the remote host, but does not automatically log in.

`-m` *mechanism* Specifies the GSS-API security mechanism to use, as listed in the `/etc/gss/mech` file. Defaults to `kerberos_v5`.

`-x` Encrypts this session.

`-X` *auth-type* Disables the *auth-type* type of authentication.

The following table shows which commands have specific options. An "X" indicates that the command has that option.

TABLE 24–1 Kerberos Options for Network Commands

	`ftp`	`rcp`	`rlogin`	`rsh`	`telnet`
`-a`					X
`-f`	X		X	X	X
`-F`			X	X	X
`-k`		X	X	X	X
`-K`					X
`-m`	X				
`-x`	X	X	X	X	X
`-X`					X

Additionally, `ftp` allows the protection level for a session to be set at its prompt:

clear Sets the protection level to "clear" (no protection). This protection level is the default.

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.

safe Sets the protection level to "safe." Data transmissions are integrity-protected by cryptographic checksum.

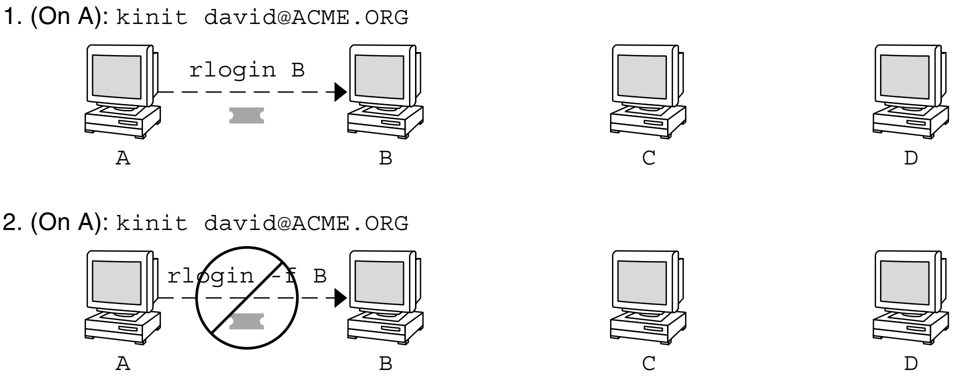
You can also set the protection level at the `ftp` prompt by typing `protect` followed by any of the protection levels shown above (`clear`, `private`, or `safe`).

Forwarding Kerberos Tickets

As described in "Overview of Kerberized Commands" on page 490, 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 the following figure, 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.

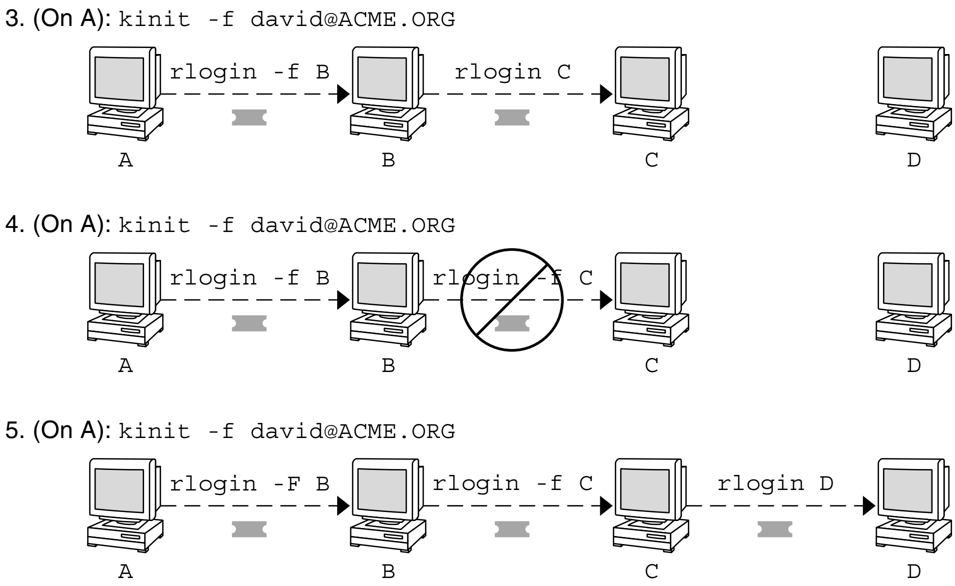
FIGURE 24–2 Using Non-Forwardable Tickets


1. (On A): kinit david@ACME.ORG
rlogin B
A
B
C
D
2. (On A): kinit david@ACME.ORG
rlogin -f B
A
B
C
D

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 the following figure, 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.

FIGURE 24–3 Using Forwardable Tickets


3. (On A): kinit -f david@ACME.ORG
rlogin -f B
rlogin C
A
B
C
D
4. (On A): kinit -f david@ACME.ORG
rlogin -f B
rlogin -f C
A
B
C
D
5. (On A): kinit -f david@ACME.ORG
rlogin -F B
rlogin -f C
rlogin D
A
B
C
D

Using Kerberized Commands (Examples)

The following examples show how the options to the Kerberized commands work.

EXAMPLE 24–5 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`.

```
% 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 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.

EXAMPLE 24–5 Using the -a, -f, and -x Options With telnet *(Continued)*

If you forward your Kerberos tickets, `telnet` (as well as the other commands discussed here) destroys them when it exits.

EXAMPLE 24–6 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 24–7 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)
150 Opening BINARY mode data connection for RMAIL (158336 bytes).
226 Transfer complete. 158336 bytes received in 1.9 seconds (1.4e+02 Kbytes/s)
ftp> quit
%
```

To encrypt the session, `joe` sets the protection level to `private`.

CHAPTER 25

The Kerberos Service (Reference)

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 497
- "Kerberos Commands" on page 499
- "Kerberos Daemons" on page 500
- "Kerberos Terminology" on page 500
- "How the Kerberos Authentication System Works" on page 505
- "Gaining Access to a Service Using Kerberos" on page 506
- "Using Kerberos Encryption Types" on page 509
- "Using the gsscred Table" on page 511
- "Notable Differences Between Oracle Solaris Kerberos and MIT Kerberos" on page 511

Kerberos Files

This section lists some of the files that are used by the Kerberos service.

TABLE 25–1 Kerberos Files

File Name	Description
`~/.gkadmin`	Default values for creating new principals in the SEAM Tool
`~/.k5login`	List of principals that grant access to a Kerberos account
`/etc/krb5/kadm5.acl`	Kerberos access control list file, which includes principal names of KDC administrators and their Kerberos administration privileges

TABLE 25–1 Kerberos Files *(Continued)*

File Name	Description
`/etc/krb5/kadm5.keytab`	Obsolete: This file was removed in the Oracle Solaris 11 release.
`/etc/krb5/kdc.conf`	KDC configuration file
`/etc/krb5/kpropd.acl`	Kerberos database propagation configuration file
`/etc/krb5/krb5.conf`	Kerberos realm configuration file
`/etc/krb5/krb5.keytab`	Keytab file for network application servers
`/etc/krb5/warn.conf`	Kerberos ticket expiration warning and automatic renewal configuration file
`/etc/pam.conf`	PAM configuration file
`/tmp/krb5cc_`*uid*	Default credentials cache, where *uid* is the decimal UID of the user
`/tmp/ovsec_adm.`*xxxxxx*	Temporary credentials cache for the lifetime of the password changing operation, where *xxxxxx* is a random string
`/var/krb5/.k5.`*REALM*	KDC stash file, which contains a copy of the KDC master key
`/var/krb5/kadmin.log`	Log file for `kadmind`
`/var/krb5/kdc.log`	Log file for the KDC
`/var/krb5/principal`	Kerberos principal database
`/var/krb5/principal.kadm5`	Kerberos administrative database, which contains policy information
`/var/krb5/principal.kadm5.lock`	Kerberos administrative database lock file
`/var/krb5/principal.ok`	Kerberos principal database initialization file that is created when the Kerberos database is initialized successfully
`/var/krb5/principal.ulog`	Kerberos update log, which contains updates for incremental propagation
`/var/krb5/slave_datatrans`	Backup file of the KDC that the `kprop_script` script uses for propagation
`/var/krb5/slave_datatrans_`*slave*	Temporary dump file that is created when full updates are made to the specified *slave*

Kerberos Commands

This section lists some commands that are included in the Kerberos product.

TABLE 25–2 Kerberos Commands

Command	Description
`/usr/bin/ftp`	File Transfer Protocol program
`/usr/bin/kdestroy`	Destroys Kerberos tickets
`/usr/bin/kinit`	Obtains and caches Kerberos ticket-granting tickets
`/usr/bin/klist`	Displays current Kerberos tickets
`/usr/bin/kpasswd`	Changes a Kerberos password
`/usr/bin/ktutil`	Manages Kerberos keytab files
`/usr/bin/kvno`	Lists key version numbers for Kerberos principals
`/usr/bin/rcp`	Remote file copy program
`/usr/bin/rlogin`	Remote login program
`/usr/bin/rsh`	Remote shell program
`/usr/bin/telnet`	Kerberized `telnet` program
`/usr/lib/krb5/kprop`	Kerberos database propagation program
`/usr/sbin/gkadmin`	Kerberos database administration GUI program, which is used to manage principals and policies
`/usr/sbin/gsscred`	Manage gsscred table entries
`/usr/sbin/kadmin`	Remote Kerberos database administration program (run with Kerberos authentication), which is used to manage principals, policies, and keytab files
`/usr/sbin/kadmin.local`	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
`/usr/sbin/kclient`	Kerberos client installation script which is used with or without a installation profile
`/usr/sbin/kdb5_ldap_util`	Creates LDAP containers for Kerberos databases
`/usr/sbin/kdb5_util`	Creates Kerberos databases and stash files
`/usr/sbin/kgcmgr`	Configures Kerberos master and slave KDCs
`/usr/sbin/kproplog`	Lists a summary of update entries in the update log

Kerberos Daemons

The following table lists the daemons that the Kerberos product uses.

TABLE 25–3 Kerberos Daemons

Daemon	Description
`/usr/sbin/in.ftpd`	File Transfer Protocol daemon
`/usr/lib/krb5/kadmind`	Kerberos database administration daemon
`/usr/lib/krb5/kpropd`	Kerberos database propagation daemon
`/usr/lib/krb5/krb5kdc`	Kerberos ticket processing daemon
`/usr/lib/krb5/ktkt_warnd`	Kerberos ticket expiration warning and automatic renewal daemon
`/usr/sbin/in.rlogind`	Remote login daemon
`/usr/sbin/in.rshd`	Remote shell daemon
`/usr/sbin/in.telnetd`	`telnet` daemon

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. An 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 483.

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 24–1 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 Oracle 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 483.

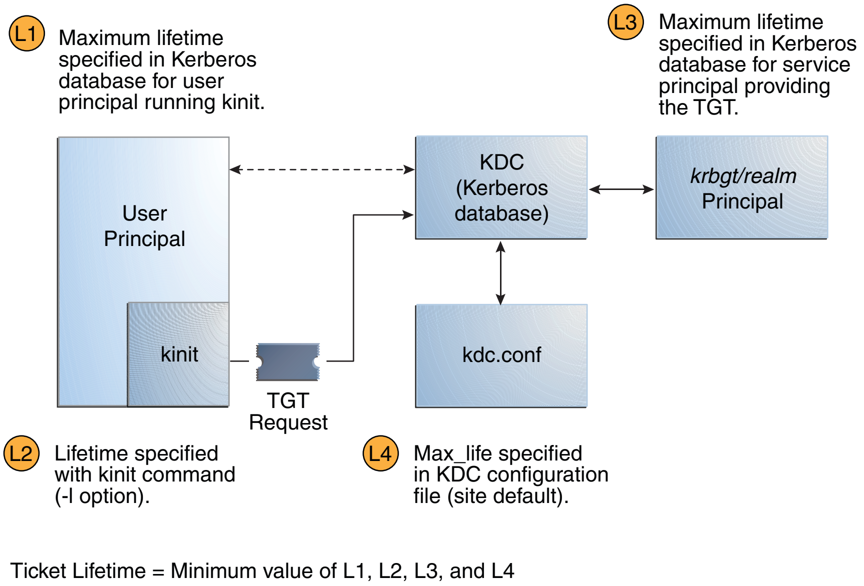
Ticket Lifetimes

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 25–1 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).

FIGURE 25–1 How a TGT's Lifetime is Determined


L1
Maximum lifetime specified in Kerberos database for user principal running kinit.
L3
Maximum lifetime specified in Kerberos database for service principal providing the TGT.
User Principal
kinit
KDC (Kerberos database)
krbgt/realm Principal
kdc.conf
TGT Request
L2
Lifetime specified with kinit command (-l option).
L4
Max_life specified in KDC configuration file (site default).
Ticket Lifetime = Minimum value of L1, L2, L3, and L4

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 25–4 Examples of Kerberos Principal Names

Principal Name	Description
`changepw/kdc1.example.com@EXAMPLE.COM`	A principal for the master KDC server that allows access to the KDC when you are changing passwords.
`clntconfig/admin@EXAMPLE.COM`	A principal that is used by the `kclient` installation utility.
`ftp/boston.example.com@EXAMPLE.COM`	A principal used by the `ftp` service. This principal can be used instead of a `host` principal.
`host/boston.example.com@EXAMPLE.COM`	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.
`K/M@EXAMPLE.COM`	The master key name principal. One master key name principal is associated with each master KDC.
`kadmin/history@EXAMPLE.COM`	A principal that includes a key used to keep password histories for other principals. Each master KDC has one of these principals.
`kadmin/kdc1.example.com@EXAMPLE.COM`	A principal for the master KDC server that allows access to the KDC by using `kadmind`.
`kadmin/changepw.example.com@EXAMPLE.COM`	A principal that is used to accept password change requests from clients that are not running an Oracle Solaris release.
`krbtgt/EXAMPLE.COM@EXAMPLE.COM`	This principal is used when you generate a ticket-granting ticket.
`krbtgt/EAST.EXAMPLE.COM@WEST.EXAMPLE.COM`	This principal is an example of a cross-realm ticket-granting ticket.
`nfs/boston.example.com@EXAMPLE.COM`	A principal that is used by the NFS service. This principal can be used instead of a `host` principal.
`root/boston.example.com@EXAMPLE.COM`	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..
username`@EXAMPLE.COM`	A principal for a user.
username`/admin@EXAMPLE.COM`	An `admin` principal that can be used to administer the KDC database.

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.

How the Kerberos Service Interacts With DNS and the nsswitch Service

The Kerberos service is compiled to use DNS to resolve host names. The `nsswitch` service is not checked at all when host name resolution is done.

Gaining Access to a Service Using Kerberos

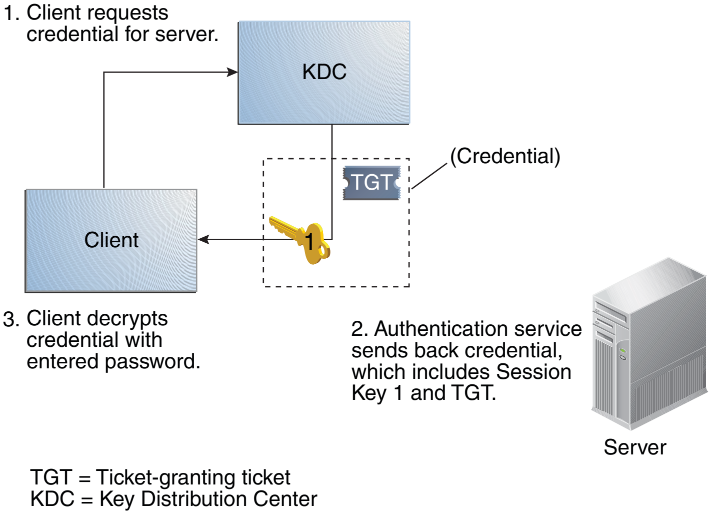
To access a specific service on a specific server, the user must obtain two credentials. The first credential is for the ticket-granting ticket (known as the TGT). Once the ticket-granting service has decrypted this credential, the service creates a second credential for the server that the user is requesting 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.

FIGURE 25–2 Obtaining a Credential for the Ticket-Granting Service

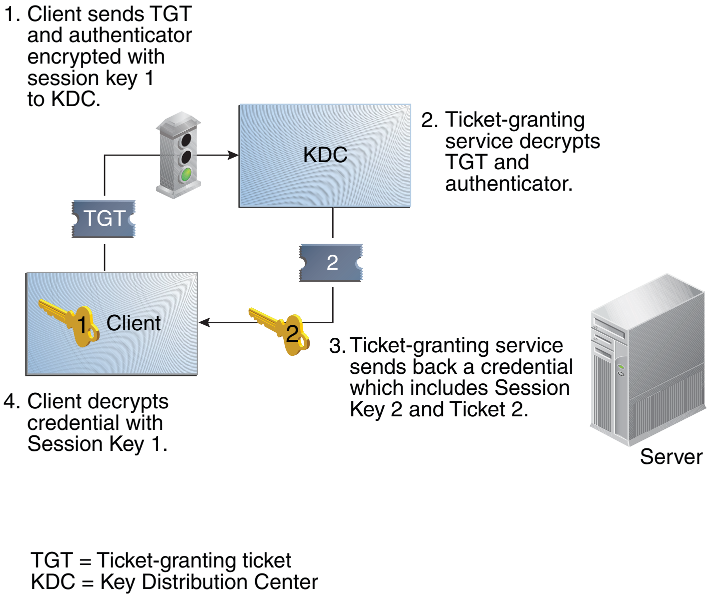

1. Client requests credential for server.
KDC
(Credential)
TGT
Client
1
3. Client decrypts credential with entered password.
2. Authentication service sends back credential, which includes Session Key 1 and TGT.
Server
TGT = Ticket-granting ticket
KDC = Key Distribution Center

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 506. 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.

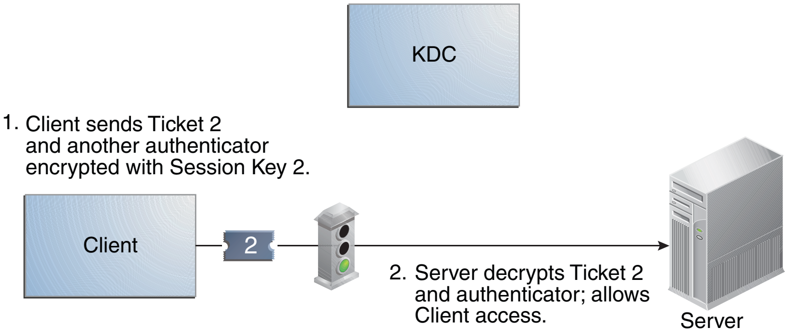
FIGURE 25–3 Obtaining a Credential for a Server


1. Client sends TGT and authenticator encrypted with session key 1 to KDC.
KDC
2. Ticket-granting service decrypts TGT and authenticator.
TGT
2
1
Client
2
3. Ticket-granting service sends back a credential which includes Session Key 2 and Ticket 2.
Server
4. Client decrypts credential with Session Key 1.
TGT = Ticket-granting ticket
KDC = Key Distribution Center

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 506 and "Obtaining a Credential for a Server" on page 507. 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.

FIGURE 25–4 Obtaining Access to a Specific Service


KDC
1. Client sends Ticket 2 and another authenticator encrypted with Session Key 2.
Client
2
2. Server decrypts Ticket 2 and authenticator; allows Client access.
Server

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/encryption-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-crc` 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 343 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 Oracle 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 Oracle 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

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.

PART VII

Auditing in Oracle Solaris

This section provides information about the configuration, management, and use of the auditing subsystem.

- Chapter 26, “Auditing (Overview)”
- Chapter 27, “Planning for Auditing”
- Chapter 28, “Managing Auditing (Tasks)”
- Chapter 29, “Auditing (Reference)”

CHAPTER 26

Auditing (Overview)

The auditing subsystem of Oracle Solaris 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 Oracle Solaris. The following is a list of the information in this chapter.

- "What Is Auditing?" on page 515
- "Audit Terminology and Concepts" on page 516
- "How Is Auditing Related to Security?" on page 524
- "How Does Auditing Work?" on page 524
- "How Is Auditing Configured?" on page 525
- "Auditing on a System With Oracle Solaris Zones" on page 526
- "About the Audit Service in This Release" on page 527

For planning suggestions, see Chapter 27, "Planning for Auditing." For procedures to configure auditing at your site, see Chapter 28, "Managing Auditing (Tasks)." For reference information, see Chapter 29, "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 PAM authentication succeeds, a unique and immutable *audit user ID* is generated and associated with the user, and 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 that login session. When a user switches to another user, all user actions are tracked with the same audit user ID. For more details about switching identity, see the su(1M) man page. Note that by default, certain actions such as booting and shutting down the system are always audited.

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

Audit Terminology and Concepts

The following terms are used to describe the audit service. Some definitions include pointers to more complete descriptions.

audit class

A grouping of audit events. Audit classes provide a way to select a group of events to be audited.

For more information, see "Audit Classes and Preselection" on page 519, and the audit_flags(5), audit_class(4), and audit_event(4) man pages.

audit file system

A repository of audit files in binary format.

For more information, see "Audit Logs" on page 521 and the audit.log(4) man page.

audit event

A security-related system action that is auditable. For ease of selection, events are grouped into audit classes.

For more information, see "Audit Events" on page 518 and the audit_event(4) man page.

audit flag

An audit class that is supplied as an argument to a command or keyword. A flag can be prefixed by a plus sign or minus sign to indicate that the class is audited for success (+) or failure (-). A preceding caret (^) indicates that a success is not to be audited (^+) or a failure is not to be audited (^-).

For more information, see the audit_flags(5) man page and "Audit Class Syntax" on page 601.

audit plugin

A module that transfers the audit records in the queue to a specified location. The audit_binfile plugin creates binary audit files. Binary files comprise the audit trail, which is stored on audit file systems. The audit_remote plugin sends binary audit records to a remote repository. The audit_syslog plugin summarizes selected audit records in the syslog logs.

For more information, see "Audit Plugin Modules" on page 520 and the module man pages, audit_binfile(5), audit_remote(5), and audit_syslog(5).

audit policy

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 queue is full.

For more information, see "Understanding Audit Policy" on page 534 and the auditconfig(1M) man page.

audit record

Audit data that is collected in the audit queue. An audit record describes a single audit event. Each audit record is composed of audit tokens.

For more information, see "Audit Records and Audit Tokens" on page 520 and the audit.log(4) man page.

audit token

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 more information, see "Audit Token Formats" on page 607 and the audit.log(4) man page.

audit trail

A collection of one or more audit files that store the audit data from all audited systems that use the default plugin, audit_binfile.

For more information, see "Audit Trail" on page 605.

post-selection

The choice of which audit events to examine in the audit trail. The default active plugin, audit_binfile, creates the audit trail. A post-selection tool, the auditreduce command, selects records from the audit trail.

For more information, see the auditreduce(1M) and praudit(1M) man pages.

preselection

The choice of which audit classes to monitor. The audit events of preselected audit classes are collected in the audit queue. Audit classes that are not preselected are not audited, so their events do not appear in the queue.

For more information, see "Audit Classes and Preselection" on page 519 and the audit_flags(5) and auditconfig(1M) man pages.

public object 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.

Audit Events

Audit events represent auditable actions on a system. Audit events are listed in the /etc/security/audit_event file. Each audit event is connected to a system call or user command, and is assigned to one or more audit classes. For a description of the format of the audit_event file, see the audit_event(4) man page.

For example, the AUE_EXECVE audit event audits the execve() system call. The command auditrecord -e execve displays this entry:

```
execve
  system call execve                 See execve(2)
  event ID    23                     AUE_EXECVE
  class       ps,ex                  (0x0000000040100000)
      header
      path
      [attribute]                    omitted on error
      [exec_arguments]               output if argv policy is set
      [exec_environment]             output if arge policy is set
      subject
      [use_of_privilege]
      return
```

When you preselect either the audit class ps or the audit class ex, then every execve() system call is recorded in the audit queue.

Auditing handles *attributable* and *non-attributable* events. Audit policy divides events into *synchronous* and *asynchronous* events, as follows:

- **Attributable events –** Events that can be attributed to a user. The execve() system call can be attributed to a user, so the call is considered an attributable event. All attributable events are synchronous events.
- **Non-attributable events –** Events that occur at the kernel-interrupt level or before a user is authenticated. The na audit class handles audit events that are non-attributable. For example, booting the system is a non-attributable event. Most non-attributable events are asynchronous events. However, non-attributable 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.

In addition to the audit events that are defined by the audit service, third-party applications can generate audit events. Audit event numbers from 32768 to 65535 are available for third-party applications. Vendors need to contact their Oracle Solaris representative to reserve event numbers and obtain access to the audit interfaces.

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, all the events in that class are recorded in the audit queue. For example, when you preselect the `ps` audit class, `execve()`, `fork()`, and other system calls are recorded.

You can preselect for events on a system and for events initiated by a particular user.

- **System-wide preselection –** Specify the system-wide defaults for auditing by using the `-setflags` and `-setnaflags` options to the `auditconfig` command.

  Note – If the `perzone` policy is set, default audit classes can be specified in every zone. For `perzone` auditing, the defaults are zone-wide, not system-wide.

- **User-specific preselection –** Specify differences from the system-wide auditing defaults for individual users by configuring the audit flags for the user. The `useradd`, `roleadd`, `usermod`, and `rolemod` commands place the `audit_flags` security attribute in the `user_attr` database. The `profiles` command places audit flags for rights profiles in the `prof_attr` database.

  The audit preselection mask determines which classes of events are audited for a user. For a description of the user preselection mask, see "Process Audit Characteristics" on page 604. For which configured audit flags are used, see "Order of Search for Assigned Security Attributes" on page 199.

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 `lo` and `ps` class definitions appear in the `audit_class` file as follows:

```
0x0000000000001000:lo:login or logout
0x0000000000100000:ps:process start/stop
```

The audit classes include the two global classes: `all` and `no`. The audit classes are described in the `audit_class(4)` man page. For the list of classes, read the `/etc/security/audit_class` file.

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 555. To view the events that are mapped to a class, use the `auditrecord -c` *class* command.

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,69,2,login - local,,example_system,2010-10-10 10:10:10.020 -07:00
subject,jdoe,jdoe,staff,jdoe,staff,1210,4076076536,69 2 example_system
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 token name. Together, the `header`, `subject`, and `return` audit tokens comprise the `login - local` audit record. To display the tokens that comprise an audit record, use the `auditrecord -e` *event* command.

For a detailed description of the structure of each audit token with an example of `praudit` output, see "Audit Token Formats" on page 607. For a description of the binary stream of audit tokens, see the `audit.log(4)` man page.

Audit Plugin Modules

You can specify which audit plugin modules handle the records that your preselection has placed in the audit queue. At least one plugin must be active. By default, the `audit_binfile` plugin is active. You configure plugins with the `auditconfig -setplugin` *plugin-name* command.

The audit service provides the following plugins:

- `audit_binfile` plugin – Handles delivery of the audit queue to the binary audit files. For more information, see the `audit.log(4)` man page.
- `audit_remote` plugin – Handles secure delivery of binary audit records from the audit queue to a configured remote server. The `audit_remote` plugin uses the `libgss()` library to authenticate the server. The transmission is protected for privacy and integrity.
- `audit_syslog` plugin – Handles delivery of selected records from the audit queue to the `syslog` logs.

To configure a plugin, see the auditconfig(1M) man page. For examples of plugin configuration, see the tasks in "Configuring Audit Logs (Tasks)" on page 556.

For information about the plugins, see the audit_binfile(5), audit_remote(5), and audit_syslog(5) man pages.

Audit Logs

Audit records are collected in audit logs. The audit service provides three output modes for audit records.

- Logs that are called *audit files* store audit records in binary format. The set of audit files from a system or site provides a complete audit record. The complete audit record is called the *audit trail*. These logs are created by the audit_binfile plugin, and can be reviewed by the praudit and auditreduce post-selection commands.
- The audit_remote plugin streams audit records to a remote repository. The repository is responsible for maintaining an audit trail and supplying post-selection tools.
- The syslog utility collects and stores text summaries of the audit record. A syslog record is not complete. The following example shows a syslog entry for a login audit record:

```
Oct 10  10:10:20 example_system auditd: [ID 6472 audit.notice] \
        login - login ok session 4076172534 by root as root:other
```

A site can configure auditing to collect audit records in all formats. You can configure the systems at your site to use binary mode locally, to send binary files to a remote repository, or to use syslog mode, or to use any combination of these modes. The following table compares binary audit records with syslog audit records.

TABLE 26–1 Comparison of Binary, Remote, and syslog Audit Records

Feature	Binary and Remote Records	syslog Records
Protocol	Binary – Writes to the file system Remote – Streams to a remote repository	Uses UDP for remote logging
Data type	Binary	Text
Record length	No limit	Up to 1024 characters per audit record
Location	Binary – Stored in a zpool on the system Remote – Remote repository	Stored in a location that is specified in the syslog.conf file

TABLE 26–1 Comparison of Binary, Remote, and syslog Audit Records *(Continued)*

Feature	Binary and Remote Records	syslog Records
How to configure	Binary – Set the p_dir attribute on the audit_binfile plugin	Make the audit_syslog plugin active and configure the syslog.conf file
	Remote – Set the p_hosts attribute on the audit_remote plugin and make the plugin active	
How to read	Binary – Typically, in batch mode, browser output in XML	In real time or searched by scripts that you have created for syslog
	Remote – Repository dictates the procedure	Plain text output
Completeness	Guaranteed to be complete and to appear in the correct order	Not guaranteed to be complete
Time stamp	Coordinated Universal Time (UTC)	Time on the system that is being audited

Binary records provide the greatest security and coverage. Binary output meets the requirements of security certifications, such as the Common Criteria (http://www.commoncriteriaportal.org/) audit requirements.

The audit_binfile plugin writes the records to a file system that you protect from snooping. On a single system, all binary records are collected and displayed in order. The UTC time stamp 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.

The audit_remote plugin writes the records to a remote repository. The repository handles storage and post-selection.

In contrast, the syslog records might 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 time stamp. 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 time stamp to construct an audit trail for several systems.

For more information about plugins and audit logs, refer to the following:

- `audit_binfile`(5) man page
- `audit_syslog`(5) man page
- `audit.log`(4) man page
- "How to Assign Audit Space for the Audit Trail" on page 559
- "How to Configure `syslog` Audit Logs" on page 563

Storing and Managing the Audit Trail

When the `audit_binfile` plugin is active, an *audit file system* holds audit files in binary format. A typical installation uses the `/var/audit` file system and can use additional file systems. The contents of all audit file systems comprise the *audit trail*. Audit records are stored in these file systems in the following order:

- **Primary audit file system**– The `/var/audit` file system, the default file system for audit files for a system
- **Secondary audit file systems** – File systems where the audit files for a system are placed at administrator discretion

The file systems are specified as arguments to the `p_dir` attribute of the `audit_binfile` plugin. A file system is not used until a file system that is earlier in the list is full. For an example with a list of file system entries, see "How to Create ZFS File Systems for Audit Files" on page 556.

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 `/var/audit`. The `-M` option to the `auditreduce` command can be used to specify the audit files from a specific machine, and the `-S` option can be used to specify a different audit file system. For more information, see the `auditreduce`(1M) man page.

The audit service provides commands to combine and filter 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.

Ensuring Reliable Time Stamps

When you merge audit logs from several systems, the date and time on those systems must be accurate. Similarly, when you send audit logs to a remote system, the recording system and the repository system must have accurate clocks. The Network Time Protocol (NTP) keeps system clocks accurate and coordinated. For more information, see Chapter 3, "Time-Related Services," in *Oracle Solaris Administration: Network Services* and the `xntpd(1M)` man page.

Managing a Remote Repository

When the `audit_remote` plugin is active, the remote repository manages the audit records.

How Is Auditing Related to Security?

Auditing helps to detect potential security breaches by revealing suspicious or abnormal patterns of system usage. 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.

Best practice requires that audit parameters be set before users log in or system processes begin, because most audit activity involves monitoring current events and reporting the events that meet the specified parameters. How the audit service monitors and reports these events is discussed in detail in Chapter 27, "Planning for Auditing," and Chapter 28, "Managing 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 user name. 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

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

After the relevant event information has been captured, the information is formatted into an audit record. Contained in each audit record is information that identifies the event, what caused the event, the time of the event, and other relevant information. This record is then placed in an audit queue for the active *plugins*. At least one plugin must be active, although all plugins can be active.

By default, one plugin is active. This is the `audit_binfile` plugin, which writes the audit records to audit files. These files are stored locally in binary format. An active `audit_remote` plugin sends these records to a remote repository. An active `audit_syslog` plugin sends text summaries to the `syslog` utility. For an illustration, see Figure 26–1.

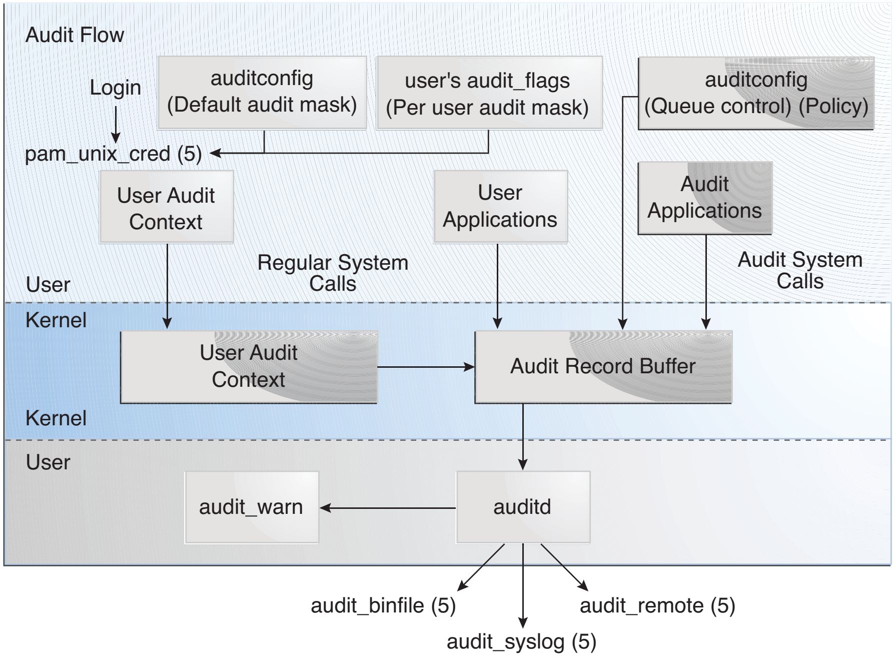
When stored locally, audit files can be in one or more ZFS pools. ZFS pools can make local storage easy to manage. These pools can be on different systems and on different but linked networks. The collection of audit files that are linked together is considered an *audit trail*.

For more information, see "How Is Auditing Configured?" on page 525, "Audit Logs" on page 521, and "Audit Plugin Modules" on page 520.

How Is Auditing Configured?

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 auditing in Oracle Solaris.

FIGURE 26–1 The Flow of Auditing



After audit data is collected in the kernel, plugins distribute the data to the appropriate locations.

- The `audit_binfile` plugin places binary audit records in the `/var/audit` file system. Post-selection tools enable you to examine interesting parts of the audit trail.
- The `audit_remote` plugin sends binary audit records across a protected link to a remote repository.
- The `audit_syslog` plugin sends text summaries of audit records to the `syslog` utility.

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 Oracle Solaris Zones" on page 600.

Auditing on a System With Oracle Solaris Zones

A zone is a virtualized operating system environment that is created within a single instance of the Oracle 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 run 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 zonename audit policy can be used to distinguish audit events that are configured 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. Each zone administrator must administer auditing. Each zone runs its own audit daemon, and has its own audit queue and audit logs. These audit logs must be managed.

About the Audit Service in This Release

The following features have been introduced to auditing:

- Auditing is a service. See "Audit Service" on page 597.
- Auditing is enabled by default.
- No reboot is required when disabling or enabling the audit service.
- The auditconfig command is used to display and change audit policy, non-attributable flags, attributable flags, plugins, and queue controls. See the auditconfig(1M) man page.
- The auditing of public objects generates less noise in the audit trail.
- The auditing of non-kernel events has no performance impact.
- By default, events in the login/logout class are audited for the system and for the root account.
- Oracle Solaris supplies three plugins, audit_binfile, audit_remote, and audit_syslog. See the audit_binfile(5), audit_remote(5), and audit_syslog(5) man pages.
- Non-global zones can be audited without the global zone having to be audited. The only requirement for auditing in non-global zones is that the perzone audit policy be set in the global zone.

- The possible number of audit classes is extended from 32 to 64. The first eight high-level bits are reserved for customers.
- The rights profiles for auditing have been reconfigured. See "Rights Profiles for Administering Auditing" on page 600.
- The audit_flags security attribute is used to configure user differences from system-wide auditing. This keyword is an argument to the useradd, usermod, roleadd, and rolemod, commands. The audit_flags value is stored in the user_attr database. See the useradd(1M), usermod(1M), roleadd(1M), rolemod(1M), and user_attr(4) man pages.

  The always_audit and never_audit keywords to the profiles command update the audit_flags security attribute in the prof_attr database. For more information, see the profiles(1) man page and "Order of Search for Assigned Security Attributes" on page 199.
- New audit classes are defined. The ft audit class contains file transfer audit events. The ftp and sftp commands are among the events that are audited by this class. The frcp audit class contains audit events that are recorded whether or not they are preselected by an administrator. The auditrecord -c *classname* command describes the audit events in these new classes.

CHAPTER 27

Planning for Auditing

This chapter describes how to customize the audit service for your Oracle Solaris installation. The following is a list of the planning information in this chapter:

- "Planning Auditing (Tasks)" on page 529
- "Understanding Audit Policy" on page 534
- "Controlling Auditing Costs" on page 537
- "Auditing Efficiently" on page 538

For an overview of auditing, see Chapter 26, "Auditing (Overview)." For procedures to configure auditing at your site, see Chapter 28, "Managing Auditing (Tasks)." For reference information, see Chapter 29, "Auditing (Reference)."

Planning 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. You also need to carefully plan who to audit and what to audit. If you are using the default `audit_binfile` plugin, audit files can quickly grow to fill the available space, so you must allocate enough disk space.

The following task map points to the major tasks that are required for planning disk space and which events to record.

Task	For Instructions
Determine auditing strategy for non-global zones	"How to Plan Auditing in Zones" on page 530
Plan storage space for the audit trail	"How to Plan Storage for Audit Records" on page 531
Determine who and what to audit	"How to Plan Who and What to Audit" on page 532

▼ How to Plan Auditing in Zones

If your system contains non-global zones, the zones can be audited as the global zone is audited, or the audit service for each non-global zone can be configured, enabled, and disabled separately. For example, you could audit only the non-global zones, and not audit the global zone.

For a discussion of the trade-offs, see "Auditing on a System With Oracle Solaris Zones" on page 526.

● **Choose one of the following options:**

- **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 you are using the `audit_binfile` or the `audit_remote` plugin, and 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 service on a system and collects audit records for every zone. You customize the `audit_class` and `audit_event` files only in the global zone, then copy these files to every non-global zone.

  a. **Use the same naming service for every zone.**

  Note – If naming service files are customized in non-global zones, and `perzone` 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.

  ---

  b. **Enable the audit records include the name of the zone.**

  To put the zone name as part of the audit record, set the `zonename` policy in the global zone. The `auditreduce` 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 532. 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 531.

- **OPTION 2 - Configure one audit service per zone.**

  Choose to configure per-zone auditing if different zones use different naming service databases, or if zone administrators want to control auditing in their zones.

Note – To audit non-global zones, the perzone policy must be set, but the audit service does not have to be enabled in the global zone. Non-global zone auditing is configured, and its audit service is enabled and disabled separately from the global zone.

- When you configure per-zone auditing, you set the perzone audit policy in the global zone. If per-zone auditing is set before a non-global zone is first booted, auditing begins at the zone's first boot. To set audit policy, see "How to Configure Per-Zone Auditing" on page 567.
- Each zone administrator configures auditing for the zone.

  A non-global zone administrator can set all policy options except perzone and ahlt.
- Each zone administrator can enable or disable auditing in the zone.
- To generate records that can be traced to their originating zone during review, set the zonename audit policy.

To plan per-zone auditing, see "How to Plan Who and What to Audit" on page 532. You can skip the first step. If the audit_binfile plugin is active, each zone administrator must also set aside storage for every zone, as described in "How to Plan Storage for Audit Records" on page 531.

▼ How to Plan Storage for Audit Records

The audit_binfile plugin creates an audit trail. The audit trail requires dedicated file space. This space must be available and secure. The system uses the /var/audit file system for initial storage. You can configure additional audit file systems for audit files. The following procedure covers the issues that you must resolve 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 530 before using this procedure.

You are using the audit_binfile plugin.

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 537 and "Auditing Efficiently" on page 538.

For practical steps, see "How to Lessen the Volume of Audit Records That Are Produced" on page 585, "How to Compress Audit Files on a Dedicated File System" on page 593, and Example 28–28.

2 **Determine which systems are to be audited and configure their audit file systems.**

Create a list of all the file systems that you plan to use. For configuration guidelines, see "Storing and Managing the Audit Trail" on page 523 and the `auditreduce(1M)` man page. To specify the audit file systems, see "How to Assign Audit Space for the Audit Trail" on page 559.

3 **Synchronize the clocks on all systems.**

For more information, see "Ensuring Reliable Time Stamps" on page 523.

▼ How to Plan Who and What to Audit

Before You Begin If you are implementing non-global zones, review "How to Plan Auditing in Zones" on page 530 before using this procedure.

1 **Determine if you want a single-system image audit trail.**

Note – This step applies only to the `audit_binfile` plugin.

Systems within a single administrative domain can create a single-system image audit trail. If your systems use different naming services, start with Step 2. Then, complete the rest of the planning steps for every system.

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 for all systems.

  For correct interpretation of the audit records, the `passwd`, `group`, and `hosts` files must be consistent.
- Configure the audit service identically on 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.

2 **Determine the audit policy.**

By default, only the `cnt` policy is enabled.

Use the `auditconfig -lspolicy` command to see a description of available policy options.

- For the effects of the policy options, see "Understanding Audit Policy" on page 534.
- For the effect of the `cnt` policy, see "Audit Policies for Asynchronous and Synchronous Events" on page 603.
- To set audit policy, see "How to Change Audit Policy" on page 549.

3 **Determine if you want to modify event-to-class mappings.**

In almost all 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 want to modify event-to-class mappings.

For an example, see "How to Change an Audit Event's Class Membership" on page 555.

4 **Determine which audit classes to preselect.**

The best time to add audit classes or to change the default classes is before users log in to the system.

The audit classes that you preselect with the `-setflags` and `-setnaflags` options to the `auditconfig` command apply to all users and processes. You can preselect a class for success, for failure, or for both.

For the list of audit classes, read the `/etc/security/audit_class` file.

5 **Determine user modifications to the system-wide preselections.**

If you decide that some users should be audited differently from the system, use the `audit_flags` security attribute to the `useradd`, `usermod`, `roleadd`, or `rolemod` command. You can also use the `profiles` command to add this attribute to a rights profile in the `prof_attr` database. The user preselection mask is modified for users who use a rights profile with explicit audit flags.

For the procedure, see "How to Configure a User's Audit Characteristics" on page 546. For which audit flag values are in effect, see "Order of Search for Assigned Security Attributes" on page 199.

6 **Decide how to manage the `audit_warn` email alias.**

The `audit_warn` script is run whenever the audit system detects 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 553.

7 **Decide in which format and where to collect audit records.**

You have three choices.

- By default, store binary audit records locally. The default storage directory is `/var/audit`. To further configure the `audit_binfile` plugin, see "How to Create ZFS File Systems for Audit Files" on page 556.
- Stream binary audit records to a remote protected repository by using the `audit_remote` plugin. You must have a receiver for the files. For the procedure, see "How to Send Audit Files to a Remote Repository" on page 562.
- Send audit record summaries to `syslog` by using the `audit_syslog` plugin. For the procedure, see "How to Configure `syslog` Audit Logs" on page 563.

For a comparison of binary and syslog formats, see "Audit Logs" on page 521.

8 Determine when to warn the administrator about shrinking disk space.

Note – This step applies only to the audit_binfile plugin.

When disk space on an audit file system drops below the minimum free space percentage, or soft limit, the audit service switches to the next available audit directory. The service then sends a warning that the soft limit has been exceeded.

To set a minimum free space percentage, see Example 28–17.

9 Decide what action to take when all the audit directories are full.

Note – This step applies only to the audit_binfile plugin.

In the default configuration, the audit_binfile plugin is active, and the cnt policy is set. In this configuration, when the kernel audit queue is full, 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 603. To configure these policy options, see Example 28–6.

However, if the audit_binfile queue is full, and the queue for another active plugin is not full, then the kernel queue will continue to send records to the plugin that is not full. When the audit_binfile queue can again accept records, the audit service will resume sending records to it.

Note – The cnt or ahlt policy is not triggered if the queue for at least one plugin is accepting audit records.

Understanding 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 properties of the audit service and determine the policies that are in effect at system boot. For more information, see the auditconfig(1M) man page.

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 27–1 Effects of Audit Policy Options

Policy Name	Description	Why Change the Policy Option?
ahlt	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 queue is 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.	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. For a fuller discussion, see "Audit Policies for Asynchronous and Synchronous Events" on page 603.
arge	When disabled, this policy omits environment variables of an executed program from the execve audit record. When enabled, this policy adds the environment variables of an executed program to the execve audit record. The resulting audit records contain much more detail than when this policy is disabled.	The disabled option collects much less information than the enabled option. For a comparison, see "How to Audit All Commands by Users" on page 587. 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 programs in the ex audit class.
argv	When disabled, this policy omits the arguments of an executed program from the execve audit record. When enabled, this policy adds the arguments of an executed program to the execve audit record. The resulting audit records contain much more detail than when this policy is disabled.	The disabled option collects much less information than the enabled option. For a comparison, see "How to Audit All Commands by Users" on page 587. 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 programs in the ex audit class are being run.
cnt	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 the audit queue is full. 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.	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. For a fuller discussion, see "Audit Policies for Asynchronous and Synchronous Events" on page 603.
group	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.	The disabled option usually satisfies requirements for site security. The enabled option makes sense when you need to audit which supplemental groups the subject belongs to.

TABLE 27–1 Effects of Audit Policy Options *(Continued)*

Policy Name	Description	Why Change the Policy Option?
`path`	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.	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.
`perzone`	When disabled, this policy maintains a single audit configuration for a system. One audit service runs in the global zone. Audit events in specific zones can be located in the audit record if the `zonename` audit token was preselected. When enabled, this policy maintains a separate audit configuration, audit queue, and audit logs for each zone. An audit service runs in each zone. This policy can be enabled in the global zone only.	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 examining audit records with the `zonename` audit token.
`public`	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 `cl`. When enabled, this policy records every read-only audit event of public objects if an appropriate audit class is preselected.	The disabled option usually satisfies requirements for site security. The enabled option is rarely useful.
`seq`	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.	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. Alternatively, you can use the `auditstat` command to view dropped records.
`trail`	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.	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 with the `sequence` token. The `trailer` token aids in the recovery of damaged audit trails.
`zonename`	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.	The disabled option is useful when you do not need to track audit behavior per zone. The enabled option is useful when you want to isolate and compare audit behavior across zones by post-selecting records according to zone.

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

If you are using the default plugin, `audit_binfile`, you must also consider the storage cost 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. If you are using the `audit_binfile` plugin, another reason is that audit administrators can move the post-selection tasks from the audited system to systems that are dedicated to analyzing audit data. Finally, unless kernel events are preselected, the kernel has no measurable impact on system performance beyond the audit service impact.

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.

For records collected by the `audit_binfile` plugin, cost also includes the time that is required to archive the records and their supporting name service databases, and to keep the records in a safe place. Supporting databases include `groups`, `hosts`, and `passwd`.

The fewer records that you generate, the less time that is required to analyze the audit trail. The sections "Cost of Storage of Audit Data" on page 537 and "Auditing Efficiently" on page 538 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

If you are using the `audit_binfile` plugin, 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:

- 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.
- Preselect audit classes judiciously to reduce the volume of records that are generated.

  Full auditing, that is, with the `all` class, fills disk space 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.
- If you are using the `audit_binfile` plugin, efficient audit file management is also important. For example, you can compress a ZFS file system that is dedicated to audit files.
- 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.

- For as many audit classes as possible, only preselect those classes for users and roles, not system-wide.
- Randomly audit only a certain percentage of users at any one time.
- If the `audit_binfile` plugin is active, reduce the disk storage requirements for audit files by filtering, merging, 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.
  - `audit_syslog` plugin – You can extend management and analysis tools that you have already developed to handle the audit records in `syslog` files.
  - `audit_binfile` plugin – You can 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 the audited systems.
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 file systems for the appearance of new `not_terminated` audit files.
5. Terminates outstanding `tail` processes when their files are no longer being written to.

CHAPTER 28

Managing Auditing (Tasks)

This chapter provides procedures to help you configure and manage auditing on an Oracle Solaris system. This chapter also includes instructions for administering the audit trail and troubleshooting the audit service. The following is a list of the information in this chapter.

- "Managing Auditing (Task Map)" on page 541
- "Configuring the Audit Service (Tasks)" on page 542
- "Configuring Audit Logs (Tasks)" on page 556
- "Configuring the Audit Service in Zones (Tasks)" on page 565
- "Enabling and Disabling the Audit Service (Tasks)" on page 568
- "Managing Audit Records on Local Systems (Tasks)" on page 572
- "Troubleshooting the Audit Service (Tasks)" on page 582

For an overview of the audit service, see Chapter 26, "Auditing (Overview)." For planning suggestions, see Chapter 27, "Planning for Auditing." For reference information, see Chapter 29, "Auditing (Reference)."

Managing Auditing (Task Map)

The following task map points to the major tasks that are required to manage auditing. With the exception of the troubleshooting section, the tasks are ordered.

Task	Description	For Instructions
1. Plan for auditing.	Contains configuration issues to decide before you configure the audit service.	"Planning Auditing (Tasks)" on page 529

Task	Description	For Instructions
2. Configure auditing.	Sets which audit events will be recorded for users and systems. Optionally, modifies audit policy, audit class-event mappings, and queue controls.	"Configuring the Audit Service (Task Map)" on page 542
	Configures plugins, which determine where audit records are stored and their format.	"Configuring Audit Logs (Tasks)" on page 556
3. Enable auditing.	Starts the audit service. Stops the audit service.	"Enabling and Disabling the Audit Service (Tasks)" on page 568
	On a host that has installed non-global zones, one audit service runs per zone. Alternatively, zones use the global zone audit service.	"Configuring the Audit Service in Zones (Tasks)" on page 565
4. Manage audit records.	Collects and analyzes audit data from the audit trail.	"Managing Audit Records on Local Systems (Task Map)" on page 572
Troubleshoot auditing.	Debugs and resolves audit service issues.	"Troubleshooting the Audit Service (Tasks)" on page 582

Configuring the Audit Service (Tasks)

Before you enable auditing on your network, you can modify the defaults to satisfy your site auditing requirements. Best practice is to customize your audit configuration as much as possible before the first users log in.

If you have implemented zones, you can choose to audit all zones from the global zone or to audit non-global zones individually. For an overview, see "Auditing and Oracle Solaris Zones" on page 600. For planning, see "How to Plan Auditing in Zones" on page 530. For procedures, see "Configuring the Audit Service in Zones (Tasks)" on page 565.

Configuring the Audit Service (Task Map)

The following task map points to the procedures for configuring auditing. All tasks are optional.

Task	Description	For Instructions
Display auditing defaults.	Before configuring auditing, displays the default policy, queue controls, flags, and plugin usage.	"How to Display Audit Service Defaults" on page 543
Select which events are audited.	Preselects system-wide audit classes. If an event is attributable, then all users are audited for this event.	"How to Preselect Audit Classes" on page 545
Select which events are audited for specific users.	Sets user-specific exceptions to the system-wide audit classes.	"How to Configure a User's Audit Characteristics" on page 546

Task	Description	For Instructions
Specify audit policy.	Defines additional audit data that your site requires.	"How to Change Audit Policy" on page 549
Specify queue controls.	Modifies the default buffer size, audit records in the queue, and interval between writing audit records to the buffer.	"How to Change Audit Queue Controls" on page 551
Create the `audit_warn` email alias.	Defines who receives email warnings when the audit service needs attention.	"How to Configure the `audit_warn` Email Alias" on page 553
Configure audit logs.	Configures the location of audit records for each plugin.	"Configuring Audit Logs (Tasks)" on page 556
Add audit classes.	Reduces the number of audit records by creating a new audit class to hold critical events.	"How to Add an Audit Class" on page 554
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 555

▼ How to Display Audit Service Defaults

The commands in this procedure display the current audit configuration. The output in this procedure is taken from an unconfigured system.

Before You Begin You must be assigned the Audit Configuration or Audit Control rights profile.

1. **Become an administrator with the required security attributes.**

   For more information, see "How to Obtain Administrative Rights" on page 160.

2. **Display the preselected classes for attributable events.**

   ```
   # auditconfig -getflags
   active user default audit flags = lo(0x1000,0x1000)
   configured user default audit flags = lo(0x1000,0x1000)
   ```

   `lo` is the flag for the `login/logout` audit class. The format of the mask output is (*success*,*failure*).

3. **Display the preselected classes for non-attributable events.**

   ```
   # auditconfig -getnaflags
   active non-attributable audit flags = lo(0x1000,0x1000)
   configured non-attributable audit flags = lo(0x1000,0x1000)
   ```

Note – To see which events are assigned to a class, and therefore which events are being recorded, run the `auditrecord -c` *class* command.

4 Display the audit policy.

```
$ auditconfig -getpolicy
configured audit policies = cnt
active audit policies = cnt
```

The *active* policy is the current policy, but the policy value is not being stored by the audit service. The *configured* policy is stored by the audit service, so the policy is restored when you restart the audit service.

5 Display information about the audit plugins.

```
$ auditconfig -getplugin
Plugin: audit_binfile (active)
    Attributes: p_dir=/var/audit;p_fsize=0;p_minfree=1;

Plugin: audit_syslog (inactive)
    Attributes: p_flags=;

Plugin: audit_remote (inactive)
    Attributes: p_hosts=;p_retries=3;p_timeout=5;
```

The `audit_binfile` plugin is active by default.

6 Display the audit queue controls.

```
$ 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) = 100
  active audit queue lowater mark (records) = 10
  active audit queue buffer size (bytes) = 8192
  active audit queue delay (ticks) = 20
```

The *active* queue control is the queue control that is currently used by the kernel. The string `no configured` indicates that the system is using the default values.

7 Display the audit classes that are preselected for existing users.

Find the users, then display each user's `audit_flags` attribute value.

```
# who
adoe    pts/1        Oct 10 10:20    (:0.0)
adoe    pts/2        Oct 10 10:20    (:0.0)
jdoe    pts/5        Oct 12 12:20    (:0.0)
jdoe    pts/6        Oct 12 12:20    (:0.0)
...
# useratt audit_flags adoe
# useratt audit_flags jdoe
```

By default, users are audited for the system-wide settings only.

For a description of the `userattr` command, see the `userattr(1)` man page. For a description of the `audit_flags` keyword, see the `user_attr(4)` man page.

▼ How to Preselect Audit Classes

Preselect audit classes that contain the events that you want to monitor. Events that are not in preselected classes are not recorded.

Before You Begin You must be assigned the Audit Configuration rights profile.

1. **Become an administrator with the required security attributes.**

   For more information, see "How to Obtain Administrative Rights" on page 160.

2. **Determine the current preselected classes.**

   ```
   # auditconfig -getflags
   ...

   # auditconfig -getnaflags
   ,,,
   ```

   For an explanation of the output, see "How to Display Audit Service Defaults" on page 543.

3. **Preselect the attributable classes.**

   ```
   # auditconfig -setflags lo,ps,fw
   user default audit flags = ps,lo,fw(0x101002,0x101002)
   ```

   This command audits the events in the login/logout, process start/stop, and file write classes for success and for failure.

   Note – The `auditconfig -setflags` command does not *add* classes to the current system defaults. This command *replaces* the system defaults, so you must specify all classes that you want to preselect.

   ---

4. **Preselect the non-attributable classes.**

   The na class contains PROM, boot, and non-attributable mounts, among other events.

   ```
   # auditconfig -setnaflags lo,na
   non-attributable audit flags = lo,na(0x1400,0x1400)
   ```

   `lo` and na are the only useful arguments to the `-setnaflags` option.

   Note – The `auditconfig -setnaflags` command *replaces* the system defaults, so you must specify all classes that you want to preselect.

▼ How to Configure a User's Audit Characteristics

By preselecting classes on a per user basis rather than on a per system basis, you can sometimes reduce the impact of auditing on system performance. Also, you might want to audit specific users slightly differently from the system.

Audit class preselections for each user are specified by the `audit_flags` security attribute. These user-specific values, plus the preselected classes for the system, determine the user's audit mask, as described in "Process Audit Characteristics" on page 604.

Before You Begin You must be in the `root` role.

● **Set the audit flags in the `user_attr` or in the `prof_attr` database.**

- **To set audit flags for a user, use the `usermod` command.**

  ```
  # usermod -K audit_flags=fw:no jdoe
  ```

  The format of the `audit_flags` keyword is *always-audit*:*never-audit*.

  always-audit Lists the audit classes that are audited for this user. Modifications to the system-wide classes are prefixed by a caret (^). Classes that are added to the system-wide classes are not prefixed by a caret.

  never-audit Lists the audit classes that are never audited for the user, even if these audit events are audited system-wide. Modifications to the system-wide classes are prefixed by a caret (^).

  To specify multiple audit classes, separate the classes with commas. For more information, see the `audit_flags(5)` man page.

- **To set audit flags for a rights profile, use the `profiles` command.**

  ```
  # profiles -p "System Administrator"
  profiles:System Administrator> set name="Audited System Administrator"
  profiles:Audited System Administrator> set always_audit=fw,as
  profiles:Audited System Administrator> end
  profiles:Audited System Administrator> exit
  ```

  When you assign the Audited System Administrator rights profile to a user or a role, that user or role is audited for those flags, subject to search order as described in "Order of Search for Assigned Security Attributes" on page 199.

Example 28–1 Changing Which Events Are Audited for One User

In this example, the audit preselection mask for all users is the following:

```
# auditconfig -getflags
active user default audit flags = ss,lo(0x11000,0x11000)
configured user default audit flags = ss,lo(0x11000,0x11000)
```

No user except the administrator is logged in.

To lessen the impact of the AUE_PFEXEC audit event on system resources, the administrator does not audit this event at the system level. Rather, the administrator preselects the pf class for a user, jdoe. The pf class is created in Example 28–10.

```
# usermod -K audit_flags=pf:no jdoe
```

The userattr command shows the addition.

```
# userattr audit_flags jdoe
pf:no
```

When the user jdoe logs in, jdoe's audit preselection mask is a combination of the audit_flags values with the system default values. 289 is the PID of jdoe's login shell.

```
# auditconfig -getpinfo 289
audit id = jdoe(1234)
process preselection mask = ss,pf,lo(0x0100000000000000,0x0100000008011000)
terminal id (maj,min,host) = 242,511,example1(192.168.160.171)
audit session id = 103203403
```

Example 28–2 Modifying Audit Preselection Exception for One User

In this example, the audit preselection mask for all users is the following:

```
# auditconfig -getflags
active user default audit flags = ss,lo(0x11000,0x11000)
configured user default audit flags = ss,lo(0x11000,0x11000)
```

No users except the administrator are logged in.

The administrator decides not to collect failed ss events for the jdoe user.

```
# usermod -K audit_flags=^-ss:no jdoe
```

The userattr command shows the exception.

```
# userattr audit_flags jdoe
^-ss:no
```

When the user jdoe logs in, jdoe's audit preselection mask is a combination of the audit_flags values with the system default values. 289 is the PID of jdoe's login shell.

```
# auditconfig -getpinfo 289
audit id = jdoe(1234)
process preselection mask = +ss,lo(0x11000,0x1000)
terminal id (maj,min,host) = 242,511,example1(192.168.160.171)
audit session id = 103203403
```

Example 28–3 Auditing Selected Users, No System-Wide Auditing

In this example, the login and role activities of four selected users are audited on the system. No audit classes are preselected for the system.

First, the administrator removes all system-wide flags.

```
# auditconfig -setflags no
user default audit flags = no(0x0,0x0)
```

Then, the administrator preselects two audit classes for the four users. The pf class is created in Example 28–10.

```
# usermod -K audit_flags=lo,pf:no jdoe
# usermod -K audit_flags=lo,pf:no kdoe
# usermod -K audit_flags=lo,pf:no pdoe
# usermod -K audit_flags=lo,pf:no zdoe
```

Then, the administrator preselects the pf class for the root role.

```
# userattr audit_flags root
# rolemod -K audit_flags=lo,pf:no root
# userattr audit_flags root
lo,pf:no
```

To continue to record unwarranted intrusion, the administrator does not change the auditing of non-attributable logins.

```
# auditconfig -getnaflags
active non-attributable audit flags = lo(0x1000,0x1000)
configured non-attributable audit flags = lo(0x1000,0x1000)
```

Example 28–4 Removing a User's Audit Flags

In the following example, the administrator removes all user-specific audit flags. Existing processes of users who are currently logged in continue to be audited.

The administrator runs the usermod command with the audit_flags keyword set to no value.

```
# usermod -K audit_flags= jdoe
# usermod -K audit_flags= kdoe
# usermod -K audit_flags= ldoe
```

Then, the administrator verifies the removal.

```
# userattr audit_flags jdoe
# userattr audit_flags kdoe
# userattr audit_flags ldoe
```

Example 28–5 Creating a Rights Profile for a Group of Users

The administrator wants all administrative rights profiles at the site to explicitly audit the `pf` class. For every rights profile that is going to be assigned, the administrator creates a site-specific version in LDAP that includes audit flags.

First, the administrator clones an existing rights profile, then changes the name and adds audit flags.

```
# profiles -p "Network Wifi Management" -S ldap
profiles: Network Wifi Management> set name="Wifi Management"
profiles: Wifi Management> set desc="Audited wifi management"
profiles: Wifi Management> set audit_always=pf
profiles: Wifi Management> exit
```

After repeating this procedure for every rights profile that is going to be used, the administrator lists the information in the Wifi Management profile.

```
# profiles -p "Wifi Management" -S ldap info
name=Wifi Management
desc=Audited wifi management
auths=solaris.network.wifi.config
help=RtNetWifiMngmnt.html
always_audit=pf
```

▼ How to Change Audit Policy

Audit policy determines the characteristics of the audit records for the local system. You might change audit policy to record detailed information about audited commands, to add a zone name to every record, or to satisfy other site security requirements.

Before You Begin You must be assigned the Audit Configuration rights profile.

1 **Become an administrator with the required security attributes.**

For more information, see "How to Obtain Administrative Rights" on page 160.

2 **View the current audit policy.**

```
$ auditconfig -getpolicy
...
```

For an explanation of the output, see "How to Display Audit Service Defaults" on page 543.

3 **View the available policy options.**

```
$ auditconfig -lspolicy
policy string    description:
ahlt             halt machine if it can not record an async event
all              all policies for the zone
```

```
arge             include exec environment args in audit recs
argv             include exec command line args in audit recs
cnt              when no more space, drop recs and keep a cnt
group            include supplementary groups in audit recs
none             no policies
path             allow multiple paths per event
perzone          use a separate queue and auditd per zone
public           audit public files
seq              include a sequence number in audit recs
trail            include trailer token in audit recs
windata_down     include downgraded window information in audit recs
windata_up       include upgraded window information in audit recs
zonename         include zonename token in audit recs
```

Note – The `perzone` and `ahlt` policy options can only be set in the global zone. For the trade-offs to using a particular policy option, see "Understanding Audit Policy" on page 534.

4 Enable or disable selected audit policy options.

```
# auditconfig [ -t ] -setpolicy [prefix]policy[,policy...]
```

`-t` Optional. Creates a temporary, or *active*, policy. You might set a temporary policy for debugging or testing purposes.

A temporary policy is in effect until the audit service is refreshed, or until the policy is modified by the `auditconfig -setpolicy` command.

prefix A *prefix* value of + adds the list of policies to the current policy. A *prefix* value of - removes the list of policies from the current policy. Without a prefix, audit policy is reset. This option enables you to retain current audit policies.

policy Selects the policy to be enabled or to be disabled.

Example 28–6 Setting the ahlt Audit Policy Option

In this example, the `cnt` policy is disabled, and the `ahlt` policy is enabled. With this configuration, system use is halted when the audit queues are full, and an asynchronous event occurs. When a synchronous event occurs, the process that created the thread hangs. This configuration is appropriate when security is more important than availability. For more information, see "Audit Policies for Asynchronous and Synchronous Events" on page 603.

```
# auditconfig -setpolicy -cnt
# auditconfig -setpolicy +ahlt
```

The plus sign (+) before the `ahlt` policy adds the policy to current policy settings. Without the plus sign, the `ahlt` policy replaces current policy settings.

Example 28–7 Setting a Temporary Audit Policy

In this example, the audit service is enabled, and the ahlt audit policy is configured. The administrator adds the trail audit policy to the active policy (+trail), but does not configure the audit service to use the trail audit policy permanently (-t). The trail policy aids in the recovery of damaged audit trails.

```
$ auditconfig -setpolicy ahlt
$ auditconfig -getpolicy
  configured audit policies = ahlt
  active audit policies = ahlt
$ auditconfig -t -setpolicy +trail
  configured audit policies = ahlt
  active audit policies = ahlt,trail
```

The administrator disables the trail policy when the debugging is completed.

```
$ auditconfig -setpolicy -trail
$ auditconfig -getpolicy
  configured audit policies = ahlt
  active audit policies = ahlt
```

Refreshing the audit service by running the audit -s command also removes this temporary policy, plus any other temporary values in the audit service. For examples of other temporary values, see "How to Change Audit Queue Controls" on page 551.

Example 28–8 Setting the perzone Audit Policy

In this example, the perzone audit policy is added to the existing policy in the global zone. The perzone policy setting is stored as a permanent property, so perzone policy is in effect during the session and when the audit service is restarted.

```
$ auditconfig -getpolicy
  configured audit policies = cnt
  active audit policies = cnt
$ auditconfig -setpolicy +perzone
$ auditconfig -getpolicy
  configured audit policies = perzone,cnt
  active audit policies = perzone,cnt
```

▼ How to Change Audit Queue Controls

The audit service provides default values for audit queue parameters. You can inspect, change, and temporarily change these values with the auditconfig command.

Before You Begin You must be assigned the Audit Configuration rights profile.

1 **Become an administrator with the required security attributes.**

For more information, see "How to Obtain Administrative Rights" on page 160.

2 **View the current values of the audit queue controls.**

```
$ auditconfig -getqctrl
...
```

For an explanation of the output, see "How to Display Audit Service Defaults" on page 543.

3 **Modify selected audit queue controls.**

For examples and a description of the audit queue controls, see the auditconfig(1M) man page.

- To modify some or all audit queue controls, use the `-setqctrl` option.

  ```
  # auditconfig [ -t ] -setqctrl hiwater lowater bufsz interval
  ```

  For example, set the *interval* value to `10` without setting the other controls.

  ```
  # auditconfig -setqctrl 0 0 0 10
  ```

- To modify a specific audit queue control, specify its option. The `-setqdelay` option is the equivalent of `-setqctrl 0 0 0` *interval*, as in # **auditconfig -setqdelay 10**.

  ```
  # auditconfig [ -t ] -setqhiwater value
  # auditconfig [ -t ] -setqlowater value
  # auditconfig [ -t ] -setqbufsz value
  # auditconfig [ -t ] -setqdelay value
  ```

Example 28–9 Resetting an Audit Queue Control to the Default

The administrator sets all audit queue controls, then changes the *lowater* value in the repository back to the default.

```
# auditconfig -setqctrl 200 5 10216 10
# auditconfig -setqctrl 200 0 10216 10
configured audit queue hiwater mark (records) = 200
no configured audit queue lowater mark
configured audit queue buffer size (bytes) = 10216
configured audit queue delay (ticks) = 10
active audit queue hiwater mark (records) = 200
active audit queue lowater mark (records) = 5
active audit queue buffer size (bytes) = 10216
active audit queue delay (ticks) = 10
```

Later, the administrator sets the *lowater* value to the default for the current session.

```
# auditconfig -setqlowater 10
# auditconfig -getqlowater
configured audit queue lowater mark (records) = 10
active audit queue lowater mark (records) = 10
```

▼ How to Configure the audit_warn Email Alias

The /etc/security/audit_warn script generates mail to notify the administrator of audit incidents that might need attention. You can customize the script and you can send the mail to an account other than root.

If the perzone policy is set, the non-global zone administrator must configure the audit_warn email alias in the non-global zone.

Before You Begin You must be in the root role.

● **Configure the audit_warn email alias.**

Choose one of the following options:

- **OPTION 1** – Replace the audit_warn email alias with another email account in the audit_warn script.

  Change the audit_warn email alias in the ADDRESS line of the script to another address:

  ```
  #ADDRESS=audit_warn            # standard alias for audit alerts
  ADDRESS=audadmin            # role alias for audit alerts
  ```

  Caution – When you upgrade to a new release of the Oracle Solaris OS, you must manually merge your customized audit_warn file with the audit_warn.new file. This new file might contain important changes. For a description of the preserve=renamenew file action on upgrade, see the pkg(5) man page.

- **OPTION 2** – Redirect the audit_warn email to another mail account.

  In this case, you would add the 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 /etc/mail/aliases entry would resemble the following if the root and audadmin email accounts were added as members of the audit_warn email alias:

  ```
  audit_warn: root,audadmin
  ```

  Then, run the newaliases command to rebuild the random access database for the aliases file.

  ```
  # newaliases
  /etc/mail/aliases: 14 aliases, longest 10 bytes, 156 bytes total
  ```

▼ 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, copy the change to all systems that are being audited. Best practice is to create audit classes before enabling the audit service.

Caution – When you upgrade to a new release of the Oracle Solaris OS, you must manually merge your customized audit_class file with the audit_class.new file. This new file might contain important changes. For a description of the preserve=renamenew file action on upgrade, see the pkg(5) man page.

Before You Begin The entry must be unique. You must choose free bits. The available bits for customer use are described in the /etc/security/audit_class file.

You must be in the root role.

1. **(Optional) Save a backup copy of the audit_class file.**

```
# cp /etc/security/audit_class /etc/security/audit_class.orig
```

2. **Add new entries to the audit_class file.**

   Each entry has the following format:

   0x*64bitnumber*:*flag*:*description*

   For a description of the fields, see the audit_class(4) man page. For the list of existing classes, read the /etc/security/audit_class file.

Example 28–10 Creating a New Audit Class

This example creates a class to hold administrative commands that are executed in a role. The added entry to the audit_class file is as follows:

```
0x0100000000000000:pf:profile command
```

The entry creates the new pf audit class. Example 28–11 populates the new audit class.

Troubleshooting If you have customized the audit_class file, make sure that any user exceptions to the system's audit preselection mask are consistent with the new audit classes. Errors occur when an audit_flags value is not a subset of the audit_class file.

▼ 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.

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.

When you reconfigure audit event-class mappings on one system, copy the change to all systems that are being audited. Best practice is to change event-class mappings before users log in.

Caution – When you upgrade to a new release of the Oracle Solaris OS, you must manually merge your customized `audit_event` file with the `audit_event.new` file. This new file might contain important changes. For a description of the `preserve=renamenew` file action on upgrade, see the `pkg`(5) man page.

Before You Begin You must be in the `root` role.

1. **(Optional) Save a backup copy of the `audit_event` file.**

   ```
   # cp /etc/security/audit_event /etc/security/audit_event.orig
   ```

2. **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 28–11 Mapping Existing Audit Events to a New Class

This example maps an existing audit event to the new class that was created in Example 28–10. By default, the `AUE_PFEXEC` audit event is mapped to four classes, `ps`, `ex`, `ua`, and `as`. By creating the new class, the administrator can audit `AUE_PFEXEC` events without auditing the events in any of the other four classes.

```
# grep pf /etc/security/audit_class
0x0100000000000000:pf:profile command
# vi /etc/security/audit_event
116:AUE_PFEXEC:execve(2) with pfexec enabled:pf
# auditconfig -setflags lo,pf
user default audit flags = pf,lo(0x0100000000001000,0x0100000000001000)
```

Configuring Audit Logs (Tasks)

Two audit plugins, `audit_binfile` and `audit_syslog`, send audit logs to locations that you can configure. The following tasks help you configure these logs.

Configuring Audit Logs (Task Map)

The following task map points to the procedures for configuring audit logs for the various plugins. All tasks are optional.

Task	Description	For Instructions
Add local storage for the `audit_binfile` plugin.	Creates local disk space for the audit files and protects them with file permissions.	"How to Create ZFS File Systems for Audit Files" on page 556
Assign storage for the `audit_binfile` plugin.	Identifies directories for binary audit records.	"How to Assign Audit Space for the Audit Trail" on page 559
Configure storage for the `audit_remote` plugin.	Enables you to send audit records to a remote repository through a protected mechanism.	"How to Send Audit Files to a Remote Repository" on page 562
Configure storage for the `audit_syslog` plugin.	Enables you to stream audit events in text format to `syslog`.	"How to Configure `syslog` Audit Logs" on page 563

▼ How to Create ZFS File Systems for Audit Files

The following procedure shows how to create a ZFS pool for audit files, as well as the corresponding file systems and mount point. By default, the `/var/audit` file system holds audit files for the `audit_binfile` plugin.

Before You Begin You must be assigned the ZFS File System Management and ZFS Storage Management rights profiles. The latter profile enables you to create storage pools.

1. **Become an administrator with the required security attributes.**

   For more information, see "How to Obtain Administrative Rights" on page 160.

2 Determine the amount of disk space that is required.

Assign at least 200 MB 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.

Note – The default class preselection creates files in `/var/audit` that grow by about 80 bytes for every recorded instance of an event in the `lo` class, such as a login, logout, or role assumption.

---

3 Create a mirrored ZFS storage pool.

The `zpool create` command creates a storage pool that is a container for the ZFS file systems. For more information, see Chapter 1, "Oracle Solaris ZFS File System (Introduction)," in *Oracle Solaris Administration: ZFS File Systems*.

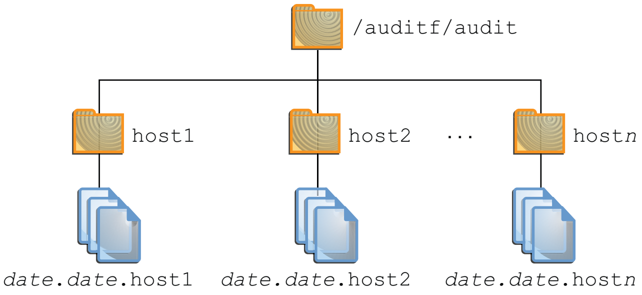
```
# zpool create audit-pool mirror disk1 disk2
```

For example, create the `auditp` pool from two disks, `c3t1d0` and `c3t2d0`, and mirror them.

```
# zpool create auditp mirror c3t1d0 c3t2d0
```

4 Create a ZFS file system and mount point for the audit files.

You create the file system and mount point with one command. At creation, the file system is mounted. For example, the following illustration shows audit trail storage that is stored by host name.



Note – If you plan to encrypt the file system, you must encrypt the file system at creation. For an example, see Example 28–12.

Encryption requires management. For example, a passphrase is required at mount time. For more information, see "Encrypting ZFS File Systems" in *Oracle Solaris Administration: ZFS File Systems*.

```
# zfs create -o mountpoint=/mountpoint audit-pool/mountpoint
```

For example, create the /audit mount point for the auditf file system.

```
# zfs create -o mountpoint=/audit auditp/auditf
```

5 Create a ZFS file system for the audit files.

```
# zfs create -p auditp/auditf/system
```

For example, create an unencrypted ZFS file system for the sys1 system.

```
# zfs create -p auditp/auditf/sys1
```

6 (Optional) Create additional file systems for audit files.

One reason to create additional file systems is to prevent audit overflow. You can set a ZFS quota per file system, as shown in Step 9. The audit_warn email alias notifies you when each quota is reached. To free space, you can move the closed audit files to a remote server.

```
# zfs create -p auditp/auditf/sys1.1
# zfs create -p auditp/auditf/sys1.2
```

7 Protect the parent audit file system.

The following ZFS properties are set to off for all file systems in the pool:

```
# zfs set devices=off auditp/auditf
# zfs set exec=off auditp/auditf
# zfs set setuid=off auditp/auditf
```

8 Compress the audit files in the pool.

Typically, compression is set in ZFS at the file system level. However, because all the file systems in this pool contain audit files, compression is set at the top-level dataset for the pool.

```
# zfs set compression=on auditp
```

See also "Interactions Between ZFS Compression, Deduplication, and Encryption Properties" in *Oracle Solaris Administration: ZFS File Systems*.

9 Set quotas.

You can set quotas at the parent file system, the descendant file systems, or both. If you set a quota on the parent audit file system, quotas on the descendant file systems impose an additional limit.

a. Set a quota on the parent audit file system.

In the following example, when both disks in the auditp pool reach the quota, the audit_warn script notifies the audit administrator.

```
# zfs set quota=510G auditp/auditf
```

b. **Set a quota on the descendant audit file systems.**

In the following example, when the quota for the `auditp/auditf/`*system* file system is reached, the `audit_warn` script notifies the audit administrator.

```
# zfs set quota=170G auditp/auditf/sys1
# zfs set quota=170G auditp/auditf/sys1.1
# zfs set quota=165G auditp/auditf/sys1.2
```

10 **For a large pool, limit the size of the audit files.**

By default, an audit file can grow to the size of the pool. For manageability, limit the size of the audit files. See Example 28–14.

Example 28–12 Creating an Encrypted File System for Audit Files

To comply with site security requirements, the administrator creates the audit file system with encryption turned on. Then, the administrator sets the mount point.

```
# zfs create -o encryption=on auditp/auditf
Enter passphrase for auditp/auditf': /** Type 8-character minimum passphrase**/
Enter again: /** Confirm passphrase **/
# zfs set -o mountpoint=/audit auditp/auditf
```

When the administrator creates additional file systems under the `auditf` file system, these descendant file systems are also encrypted.

Example 28–13 Setting a Quota on the /var/audit Directory

In this example, the administrator sets a quota on the default audit file system. When this quota is reached, the `audit_warn` script warns the audit administrator.

```
# zfs set quota=252G rpool/var/audit
```

▼ How to Assign Audit Space for the Audit Trail

In this procedure, you use attributes to the `audit_binfile` plugin to assign additional disk space to the audit trail.

Before You Begin You must be assigned the Audit Configuration rights profile.

1 **Become an administrator with the required security attributes.**

For more information, see "How to Obtain Administrative Rights" on page 160.

2 **Determine the attributes to the `audit_binfile` plugin.**

Read the OBJECT ATTRIBUTES section of the audit_binfile(5) man page.

```
# man audit_binfile
...
OBJECT ATTRIBUTES
     The p_dir attribute specifies where the audit files will be
     created. The directories are listed in the order in which
     they are to be used.

     The p_minfree attribute defines the percentage of free space
     that the audit system requires before the audit daemon invokes
     the audit_warn script.

     The p_fsize attribute defines the maximum size in bytes that
     an  audit  file can become before it is automatically closed
     and a new audit file  opened.  ...
```

3 **To add directories to the audit trail, specify the `p_dir` attribute.**

The default file system is `/var/audit`.

```
# auditconfig -setplugin audit_binfile active p_dir=/audit/sys1.1,/var/audit
```

The preceding command sets the `/audit/sys1.1` file system as the primary directory for audit files and the default `/var/audit` file system as the secondary directory. In this scenario, `/var/audit` is the directory of last resort. For this configuration to succeed, the `/audit/sys1.1` file system must exist.

You created a similar file system in "How to Create ZFS File Systems for Audit Files" on page 556.

4 **Refresh the audit service.**

The `auditconfig -setplugin` command sets the *configured* value. This value is a property of the audit service, so is restored when the service is refreshed or restarted. The configured value becomes *active* when the audit service is refreshed or restarted. For information about configured and active values, see the auditconfig(1M) man page.

```
# audit -s
```

Example 28–14 Limiting File Size for the audit_binfile Plugin

In the following example, the size of a binary audit file is set to a specific size. The size is specified in megabytes.

```
# auditconfig -setplugin audit_binfile active p_fsize=4M
# auditconfig -getplugin audit_binfile
Plugin: audit_binfile (active)
    Attributes: p_dir=/var/audit;p_fsize=4M;p_minfree=1;
```

By default, an audit file can grow without limit. To create smaller audit files, the administrator specifies a file size limit of 4MB. The audit service creates a new file when the size limit is reached. The file size limit goes into effect after the administrator refreshes the audit service.

```
# audit -s
```

Example 28–15 Specifying Several Changes to an Audit Plugin

In the following example, the administrator on a system with high throughput and a large ZFS pool changes the queue size, the binary file size, and the soft limit warning for the audit_binfile plugin. The administrator allows audit files to grow to 4GB, is warned when 2 percent of the ZFS pool remains, and doubles the allowed queue size. The default queue size is the high water mark for the kernel audit queue, 100, as in active audit queue hiwater mark (records) = 100.

```
# auditconfig -getplugin audit_binfile
Plugin: audit_binfile (active)
    Attributes: p_dir=/var/audit;p_fsize=2G;p_minfree=1;
# auditconfig -setplugin audit_binfile active "p_minfree=2;p_fsize=4G" 200
# auditconfig -getplugin audit_binfile
Plugin: audit_binfile (active)
    Attributes: p_dir=/var/audit;p_fsize=4G;p_minfree=2;
    Queue size: 200
```

The changed specifications go into effect after the administrator refreshes the audit service.

```
# audit -s
```

Example 28–16 Removing Queue Size for an Audit Plugin

In the following example, the queue size for the audit_binfile plugin is removed.

```
# auditconfig -getplugin audit_binfile
Plugin: audit_binfile (active)
    Attributes: p_dir=/var/audit;p_fsize=4G;p_minfree=2;
    Queue size: 200
# auditconfig -setplugin audit_binfile active "" ""
# auditconfig -getplugin audit_binfile
 Plugin: audit_binfile (active)
    Attributes: p_dir=/var/audit;p_fsize=4G;p_minfree=2;
```

The final empty quotation marks ("") set the queue size for the plugin to the default.

The change in qsize specification for the plugin goes into effect after the administrator refreshes the audit service.

```
# audit -s
```

Example 28–17 Setting a 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 two percent of the file system is still available.

```
# auditconfig -setplugin audit_binfile active p_minfree=2
```

The default percentage is one (1). For a large ZFS pool, choose a reasonably low percentage. For example, 10 percent of a 16 TB pool is around 16 GB, which would warn the audit administrator when plenty of disk space remains. A value of 2 sends the audit_warn message when about two GB of disk space remains.

The audit_warn email alias receives the warning. To set up the alias, see "How to Configure the audit_warn Email Alias" on page 553.

For a large pool, the administrator also limits the file size to 3 GB.

```
# auditconfig -setplugin audit_binfile active p_fsize=3G
```

The p_minfree and p_fsize specifications for the plugin go into effect after the administrator refreshes the audit service.

```
# audit -s
```

▼ How to Send Audit Files to a Remote Repository

In this procedure, you use attributes to the audit_remote plugin to send the audit trail to a remote audit repository.

Before You Begin You must have a receiver of audit files at your remote repository. You must be assigned the Audit Configuration rights profile.

1. **Become an administrator with the required security attributes.**

   For more information, see "How to Obtain Administrative Rights" on page 160.

2. **Determine the attributes to the audit_remote plugin.**

   Read the OBJECT ATTRIBUTES section of the audit_remote(5) man page.

```
# man audit_remote
...
OBJECT ATTRIBUTES
     The p_hosts attribute specifies the remote servers.
     You can also specify the port number and the GSS-API
     mechanism.

     The p_retries attribute specifies the number of retries for
     connecting and sending data. The default is 3.

     The p_timeout attribute specifies the number of seconds
     in which a connection times out.
```

The default port is the solaris_audit IANA-assigned port, 16162/tcp. The default mechanism is kerberos_v5. The timeout default is 5 seconds. You can also specify a queue size for the plugin.

3 **To specify the remote hosts, use the `p_hosts` attribute.**

```
# auditconfig -setplugin audit_remote active p_hosts=rhost1:16088:kerberos_v5
```

4 **To specify the number of retries, use the `p_retries` attribute.**

```
# auditconfig -setplugin audit_remote active p_retries=5
```

5 **To specify the length of a connection timeout, use the `p_timeout` attribute.**

```
# auditconfig -setplugin audit_remote active p_timeout=3
```

6 **Refresh the audit service.**

The audit service reads the audit plugin change upon refresh.

```
# audit -s
```

▼ How to Configure syslog Audit Logs

You can instruct the audit service to copy some or all of the audit records in the audit queue to the `syslog` utility. If you record both binary audit data and text summaries, the binary data provide a complete audit record, while the summaries filter the data for real-time review.

Before You Begin To configure the `audit_syslog` plugin, you must be assigned the Audit Configuration rights profile. To configure the `syslog` utility, you must be in the `root` role.

1 **Select audit classes to be sent to the `audit_syslog` plugin, and make the plugin active.**

Note – `p_flags` audit classes must be preselected as either system defaults or in a user's or a rights profile's audit flags. Records are not collected for a class that is not preselected.

```
# auditconfig -setplugin audit_syslog active p_flags=lo,+as,-ss
```

2 **Configure the `syslog` utility.**

a. **Add an `audit.notice` entry to the `syslog.conf` file.**

The entry includes the location of the log file.

```
# cat /etc/syslog.conf
...
audit.notice       /var/adm/auditlog
```

b. **Create the log file.**

```
# touch /var/adm/auditlog
```

c. **Refresh the configuration information for the `syslog` service.**

```
# svcadm refresh system/system-log
```

3 **Refresh the audit service.**

The audit service reads the changes to the audit plugin upon refresh.

```
# audit -s
```

4 **Regularly archive the `syslog` log files.**

The audit service can generate extensive output. To manage the logs, see the `logadm`(1M) man page.

Example 28–18 Specifying Audit Classes for syslog Output

In the following example, the `syslog` utility collects a subset of the preselected audit classes. The `pf` class is created in Example 28–10.

```
# auditconfig -setnaflags lo,na
# auditconfig -setflags lo,ss
# usermod -K audit_flags=pf:no jdoe
# auditconfig -setplugin audit_syslog active p_flags=lo,+na,-ss,+pf
```

The arguments to the `auditconfig` command instruct the system to collect all login/logout, non-attributable, and change of system state audit records. The `audit_syslog` plugin entry instructs the `syslog` utility to collect all logins, successful non-attributable events, and failed changes of system state.

For the `jdoe` user, the binary audit record includes all uses of a call to the `pfexec` command. For these events to be available for post-selection, either the `audit_binfile` or the `audit_remote` plugin must be active. The `syslog` utility collects successful calls to the `pfexec` command.

Example 28–19 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 `sys1.1`. The remote system is `remote1`.

```
sys1.1 # 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.

```
remote1 # cat /etc/syslog.conf
...
audit.notice       /var/adm/auditlog
```

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 configure auditing per zone. For background, see "Auditing on a System With Oracle Solaris Zones" on page 526. To plan, see "How to Plan Auditing in Zones" on page 530.

When you audit the non-global zones exactly as the global zone is audited, the audit service runs in the global zone. The service collects audit records from the global zone and all the non-global zones. The non-global zone administrators might not have access to the audit records.

Note – The global zone administrator can choose to modify the audit masks of users in non-global zones.

When you audit the non-global zones individually, a separate audit service runs in each zone that is audited. Each zone collects its own audit records. The records are visible to the non-global zone and to the global zone from the non-global zone root.

▼ 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.

Before You Begin You must be in the root role.

1 **Configure the global zone for auditing.**

Complete the tasks in "Configuring the Audit Service (Task Map)" on page 542, with the following exceptions:

- Do not enable perzone audit policy.
- Do not enable the audit service. You enable the audit service after you have configured the non-global zones for auditing.
- Set the zonename policy. This policy adds the name of the zone to every audit record.

```
# auditconfig -setpolicy +zonename
```

2 **If you modified audit configuration files, copy them from the global zone to every non-global zone.**

If you modified the `audit_class` or `audit_event` file, copy it in one of two ways:

- You can loopback mount the files.
- You can copy the files.

The non-global zone must be running.

- **Mount the changed `audit_class` and `audit_event` files as a loopback file system (`lofs`).**

  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]
      commit
      end
   exit
  ```

  c. **To make the changes effective, boot the non-global zone.**

  ```
  # zoneadm -z non-global-zone boot
  ```

  Later, if you modify an audit configuration file in the global zone, you reboot the zone to refresh the loopback-mounted files in the non-global zones.

- **Copy the files.**

  a. **From the global zone, list the `/etc/security` directory in the non-global zone.**

  ```
  # ls /zone/zonename/root/etc/security/
  ```

  b. **Copy the changed `audit_class` and `audit_event` files to the zone's `/etc/security` directory.**

  ```
  # cp /etc/security/audit-file /zone/zonename/root/etc/security/audit-file
  ```

  Later, if you change one of these files in the global zone, you must re-copy the file to the non-global zones.

The non-global zones are audited when the audit service is enabled in the global zone.

Example 28–20 Mounting Audit Configuration Files as Loopback Mounts in a Zone

In this example, the system administrator has modified the `audit_class`, `audit_event`, and `audit_warn` files.

The audit_warn file is read in the global zone only, so does not have to be 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 modifying the audit configuration files. If the administrator later modifies the files, the zone must be rebooted to re-read the loopback mounts.

```
# 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]
    commit
    end
 add fs
    set special=/etc/security/audit_event
    set dir=/etc/security/audit_event
    set type=lofs
    add options [ro,nodevices,nosetuid]
    commit
    end
  exit
# zonecfg -z machine1-appserver
 add fs
    set special=/etc/security/audit_class
    set dir=/etc/security/audit_class
    set type=lofs
    add options [ro,nodevices,nosetuid]
    commit
    end
...
 exit
```

When the non-global zones are rebooted, the audit_class and audit_event 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.

Before You Begin You must be assigned the Audit Configuration rights profile to configure auditing. You must be assigned the Audit Control rights profile to enable the audit service.

1 **Become an administrator with the required security attributes.**

For more information, see "How to Obtain Administrative Rights" on page 160.

2 **In the global zone, configure auditing.**

a. **Complete the tasks in "Configuring the Audit Service (Task Map)" on page 542.**

b. **Add the `perzone` audit policy. For the command, see Example 28–8.**

Note – You are not required to enable the audit service in the global zone.

---

3 **In each non-global zone that you plan to audit, configure the audit files.**

a. **Complete the tasks in "Configuring the Audit Service (Task Map)" on page 542.**

b. **Do not configure system-wide audit settings.**

Specifically, do not add the `perzone` or `ahlt` policy to the non-global zone.

4 **Enable auditing in your zone.**

```
myzone# audit -s
```

Example 28–21 Disabling Auditing in a Non-Global Zone

This example works if the global zone has set the `perzone` audit policy. The zone administrator of the `noaudit` zone disables auditing for that zone.

```
noauditzone # auditconfig -getcond
audit condition = auditing
noauditzone # audit -t
noauditzone # auditconfig -getcond
audit condition = noaudit
```

Enabling and Disabling the Audit Service (Tasks)

The audit service is enabled by default and configured by the `auditconfig` command. If the `perzone` audit policy is set in the global zone, zone administrators can enable, refresh, and disable the service in their non-global zones.

▼ How to Refresh the Audit Service

This procedure updates the audit service when you have changed the configuration of an audit plugin after the audit service is enabled.

Before You Begin You must be assigned the Audit Control rights profile.

1 **Become an administrator with the required security attributes.**

For more information, see “How to Obtain Administrative Rights” on page 160.

2 **Refresh the audit service.**

```
# audit -s
```

Note – When you refresh the audit service, all temporary configuration settings are lost. Audit policy and queue controls allow temporary settings. For more information, see the auditconfig(1M) man page.

---

3 **Update the preselection masks of users who are currently being audited.**

Audit records are generated based on the audit preselection mask that is associated with each process. Refreshing the audit service does *not* change the masks of existing processes. To explicitly reset the preselection mask for an existing process, see “How to Update the Preselection Mask of Logged In Users” on page 591.

Example 28–22 Refreshing an Enabled Audit Service

In this example, the administrator reconfigures auditing, verifies the changes, then refreshes the audit service.

- First, the administrator adds a temporary policy.

  ```
  # auditconfig -t -setpolicy +zonename
  # auditconfig -getpolicy
  configured audit policies = ahlt,arge,argv,perzone
  active audit policies = ahlt,arge,argv,perzone,zonename
  ```

- Then, the administrator specifies queue controls.

  ```
  # auditconfig -setqctrl 200 20 0 0
  # auditconfig -getqctrl
  configured audit queue hiwater mark (records) = 200
  configured audit queue lowater mark (records) = 20
  configured audit queue buffer size (bytes) = 8192
  configured audit queue delay (ticks) = 20
  active audit queue hiwater mark (records) = 200
  active audit queue lowater mark (records) = 20
  active audit queue buffer size (bytes) = 8192
  active audit queue delay (ticks) = 20
  ```

- Then, the administrator specifies plugin attributes.
  - For the audit_binfile plugin, the administrator removes the qsize value.

    ```
    # auditconfig -getplugin audit_binfile
    Plugin: audit_binfile (active)
        Attributes: p_dir=/audit/sys1.1,/var/audit;
        p_minfree=2;p_fsize=4G;
    Queue size: 200
    # auditconfig -setplugin audit_binfile active  "" ""
    # auditconfig -getplugin audit_binfile
    ```

```
        Plugin: audit_binfile (active)
            Attributes: p_dir=/audit/sys1.1,/var/audit
         p_minfree=2;p_fsize=4G;
        ```

        The final empty quotation marks ("") set the queue size for the plugin to the default.

    - For the audit_syslog plugin, the administrator specifies that successful login and logout events and failed executables be sent to syslog. The qsize for this plugin is set to 50.

        ```
        # auditconfig -setplugin audit_syslog active p_flags=+lo,-ex 50
        # auditconfig -getplugin audit_syslog
        auditconfig -getplugin audit_syslog
        Plugin: audit_syslog (active)
            Attributes: p_flags=+lo,-ex;
            Queue size: 50
        ```

    - The administrator does not configure or use the audit_remote plugin.

- Then, the administrator refreshes the audit service and verifies the configuration.
    - The temporary zonename policy is no longer set.

        ```
        # audit -s
        # auditconfig -getpolicy
        configured audit policies = ahlt,arge,argv,perzone
        active audit policies = ahlt,arge,argv,perzone
        ```

    - The queue controls remain the same.

        ```
        # auditconfig -getqctrl
        configured audit queue hiwater mark (records) = 200
        configured audit queue lowater mark (records) = 20
        configured audit queue buffer size (bytes) = 8192
        configured audit queue delay (ticks) = 20
        active audit queue hiwater mark (records) = 200
        active audit queue lowater mark (records) = 20
        active audit queue buffer size (bytes) = 8192
        active audit queue delay (ticks) = 20
        ```

    - The audit_binfile plugin does not have a specified queue size. The audit_syslog plugin has a specified queue size.

        ```
        # auditconfig -getplugin
        Plugin: audit_binfile (active)
            Attributes: p_dir=/var/audit;p_fsize=4G;p_minfree=2;

        Plugin: audit_syslog (active)
            Attributes: p_flags=+lo,-ex;
            Queue size: 50
        ...
        ```

▼ How to Disable the Audit Service

This procedure shows how to disable auditing in the global zone and in a non-global zone when the `perzone` audit policy is set.

- If the `perzone` audit policy is not set, auditing is disabled for all zones.
- If the `perzone` audit policy is set in the global zone, the policy remains in effect in the non-global zones that have enabled auditing.

  Because the `perzone` policy is set in the global zone, the non-global zone continues to collect audit records across global zone reboots and non-global zone reboots.

Before You Begin You must be assigned the Audit Control rights profile.

1 **Become an administrator with the required security attributes.**

For more information, see "How to Obtain Administrative Rights" on page 160.

2 **Run the `audit -t` command to disable the audit service.**

For more information, see the `audit(1M)` and `auditd(1M)` man pages.

- **In the global zone, disable the audit service.**

  ```
  # audit -t
  ```

  If the `perzone` audit policy is not set, this command disables auditing in all zones.

- **In a non-global zone, disable the audit service.**

  If the `perzone` audit policy is set, the non-global zone administrator must disable the service in the non-global zone.

  ```
  zone1 # audit -t
  ```

▼ How to Enable the Audit Service

This procedure enables the audit service for all zones after the service is disabled by an administrator. To start the audit service in a non-global zone, see Example 28–23.

Before You Begin To enable or disable the audit service, you must be assigned the Audit Control rights profile.

1 **Become an administrator with the required security attributes.**

For more information, see "How to Obtain Administrative Rights" on page 160.

2 **Use the `audit -s` command to enable the audit service.**

```
# audit -s
```

For more information, see the audit(1M) man page.

3 **Verify that auditing is enabled.**

```
# auditconfig -getcond
audit condition = auditing
```

Example 28–23 Enabling Auditing in a Non-Global Zone

In this example, the zone administrator enables the audit service for zone1 after taking the following actions are taken:

- The global zone administrator sets the perzone policy in the global zone.
- The zone administrator of the non-global zone configures the audit service and per-user customizations.

Then, the zone administrator enables the audit service for the zone.

```
zone1# audit -s
```

Managing Audit Records on Local Systems (Tasks)

The default plugin, audit_binfile, creates an audit trail. 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.

Managing Audit Records on Local Systems (Task Map)

The following task map points to procedures for selecting, analyzing, and managing audit records.

Task	Description	For Instructions
Display the formats of audit records.	Shows the kind of information that is collected for an audit event, and the order in which the information is presented.	"How to Display Audit Record Definitions" on page 573
Merge audit records.	Combines audit files from several machines into one audit trail.	"How to Merge Audit Files From the Audit Trail" on page 574
Select records to examine.	Selects particular events for study.	"How to Select Audit Events From the Audit Trail" on page 576
Display audit records.	Enables you to view binary audit records.	"How to View the Contents of Binary Audit Files" on page 578
Clean up incorrectly named audit files.	Provides an end timestamp to audit files that were inadvertently left open by the audit service.	"How to Clean Up a not_terminated Audit File" on page 580

Task	Description	For Instructions
Prevent audit trail overflow.	Prevents the audit file systems from becoming full.	“How to Prevent Audit Trail Overflow” on page 581

▼ How to Display Audit Record Definitions

The `auditrecord` command displays audit record definitions. The definitions provide the audit event number, audit class, selection mask, and record format of an audit event.

- **Put the definitions of all audit event records in an HTML file.**

  The `-a` option lists all audit event definitions. The `-h` option puts the list in HTML format.

  ```
  % auditrecord -ah > audit.events.html
  ```

Tip – When you display the HTML file in a browser, use the browser's Find tool to find specific audit record definitions.

For more information, see the `auditrecord(1M)` man page.

Example 28–24 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`, and the Secure Shell feature of Oracle Solaris.

```
% auditrecord -p login
...
login: logout
  program     various              See login(1)
  event ID    6153                 AUE_logout
  class       lo                  (0x0000000000001000)
...
newgrp
  program     newgrp               See newgrp login
  event ID    6212                 AUE_newgrp_login
  class       lo                  (0x0000000000001000)
...
rlogin
  program     /usr/sbin/login      See login(1) - rlogin
  event ID    6155                 AUE_rlogin
  class       lo                   (0x0000000000001000)
...
/usr/lib/ssh/sshd
  program     /usr/lib/ssh/sshd    See login - ssh
  event ID    6172                 AUE_ssh
  class       lo                   (0x0000000000001000)
...
telnet login
  program     /usr/sbin/login      See login(1) - telnet
```

```
  event ID    6154                    AUE_telnet
  class       lo                      (0x0000000000001000)
  ...
```

Example 28–25 Displaying the Audit Record Formats of an Audit Class

In this example, the format of all audit records in the pf class that was created in Example 28–10 is displayed.

```
% auditrecord -c pf

pfexec
  system call pfexec                  See execve(2) with pfexec enabled
  event ID    116                     AUE_PFEXEC
  class       pf                      (0x0100000000000000)
      header
      path                       pathname of the executable
      path                       pathname of working directory
      [privileges]               privileges if the limit or inheritable set are changed
      [privileges]               privileges if the limit or inheritable set are changed
      [process]                  process if ruid, euid, rgid or egid is changed
      exec_arguments
      [exec_environment]         output if arge policy is set
      subject
      [use_of_privilege]
      return
```

The use_of_privilege token is recorded whenever privilege is used. The privileges tokens are recorded if the limit or inheritable set is changed. The process token is recorded if an ID is changed. No policy option is required for these tokens to be included in the record.

▼ 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. If no path is specified, the auditreduce command uses the /var/audit file system.

Note – Because the time stamps in the audit trail are in Coordinated Universal Time (UTC), the date and hour must be translated to the current time zone to be meaningful. Be aware of this point whenever you manipulate these files with standard file commands rather than with the auditreduce command.

Before You Begin You must be assigned the Audit Review rights profile.

1 **Become an administrator with the required security attributes.**

For more information, see "How to Obtain Administrative Rights" on page 160.

2 Create a file system for storing merged audit files.

This file system should be in a *different zpool* from the file systems that you created in "How to Create ZFS File Systems for Audit Files" on page 556 to store the original files.

3 Merge the audit records in the audit trail.

Change directories to the directory for storing merged audit files. From this directory, merge the audit records into a file with a named suffix. All directories in the audit trail on the local system are merged.

```
# cd audit-storage-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.

`-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.

`-R` *pathname* Specifies to read audit files in *pathname*, an alternate audit root directory.

`-S` *server* Specifies to read audit files from the specified server.

For the full list of options, see the `auditreduce`(1M) man page.

4 Move the merged file to the file system in the different zpool.

To move the file to a different system, use the `sftp` command. For instructions, see the `sftp`(1) man page.

Example 28–26 Copying Audit Files to a Summary File

In the following example, an administrator who is assigned the System Administrator rights profile copies all files from the audit trail into a merged file on a different file system. The `/var/audit/storage` file system is on a separate disk from the `/var/audit` file system, the audit root file system.

```
$ cd /var/audit/storage
$ auditreduce -A -O All
$ ls /var/audit/storage/*All
20100827183214.20100827215318.All
```

In the following example, only complete files are copied from the audit trail into a merged file. The complete path is specified as the value of the -O option. The last component of the path, Complete, is used as the suffix.

```
$ auditreduce -C -O /var/audit/storage/Complete
$ ls /var/audit/storage/*Complete
20100827183214.20100827214217.Complete
```

In the following example, only complete files are copied from the sys1.1 system into a merged file.

```
$ cd /var/audit/storage
$ auditreduce -M sys1.1 -O example1summ
$ ls /var/audit/storage/*summ
20100827183214.20100827214217.example1summ
```

Example 28–27 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 for the sys1.1 system are copied to the audit_summary file system for later examination.

```
$ cd /var/audit/audit_summary
$ auditreduce -C -O daily_sys1.1 -D sys1.1
$ ls *sys1.1
20100827183214.20100827214217.daily_sys1.1
```

The audit files from the sys1.1 system that were the input to the *daily_sys1.1 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.

Before You Begin You must be assigned the Audit Review rights profile.

1. **Become an administrator with the required security attributes.**

   For more information, see "How to Obtain Administrative Rights" on page 160.

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 *yyymmdd*. 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.

`-m` Selects all of the instances of a particular audit event. The *argument* is an audit event.

optional-file Is the name of an audit file.

For the full list of options, see the `auditreduce`(1M) man page.

Example 28–28 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
# auditreduce -O lo.summary -b 20100827 -c lo; compress *lo.summary
```

Example 28–29 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 login user.

```
$ cd /var/audit/audit_summary
$ 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, 2010, 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 *yyyymmdd*.

```
# auditreduce -M tamiko -O tamikolo -d 20100907 -u tamiko -c lo
```

Example 28–30 Copying Selected Records to a Single File

In this example, login and logout records for a particular day are selected from the audit trail. The records are merged into a target file. The target file is written in a file system other than the file system that contains the audit root directory.

```
# auditreduce -c lo -d 20100827 -O /var/audit/audit_summary/logins
# ls /var/audit/audit_summary/*logins
/var/audit/audit_summary/20100827183936.20100827232326.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.

Before You Begin You must be assigned the Audit Review rights profile.

1 **Become an administrator with the required security attributes.**

For more information, see "How to Obtain Administrative Rights" on page 160.

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.

```
$ auditreduce -c lo | praudit -s
header,69,2,AUE_screenlock,,mach1,2010-10-14 08:02:56.348 -07:00
subject,jdoe,root,staff,jdoe,staff,856,50036632,82 0 mach1
return,success,0
sequence,1298
```

- 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
21,69,2,6222,0x0000,10.132.136.45,1287070091,698391050
36,26700,0,10,26700,10,856,50036632,82 0 10.132.136.45
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. The following listing divides two lines of output to fit on this printed page:

```
$ auditreduce -c lo | praudit -x
<record version="2" event="screenlock - unlock" host="mach1"
     iso8601="2010-10-14 08:28:11.698 -07:00">
<subject audit-uid="jdoe" uid="root" gid="staff" ruid="jdoe
    rgid="staff" pid="856" sid="50036632" tid="82 0 mach1"/>
<return errval="success" retval="0"/>
<sequence seq-num="1298"/>
</record>
```

Example 28–31 Printing the Entire Audit Trail

With a pipe to the print command, the output for the entire audit trail goes to the printer. For security reasons, the printer has limited access.

```
# auditreduce | praudit | lp -d example.protected.printer
```

Example 28–32 Viewing a Specific Audit File

In this example, a summary login file is examined in a terminal window.

```
# cd /var/audit/audit_summary/logins
# praudit 20100827183936.20100827232326.logins | more
```

Example 28–33 Putting Audit Records in XML Format

In this example, the audit records are converted to XML format.

```
# praudit -x 20100827183214.20100827215318.logins > 20100827.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.

Example 28–34 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.

```
#!/bin/sh
#
## This script takes an argument of a user-specified string.
#  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.

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.sys1.1 to stdin.
```

Run the `praudit` command in a profile shell. You must be assigned the Audit Review rights profile.

▼ How to Clean Up a not_terminated Audit File

When anomalous system interruptions occur, the audit service exits while its audit file is still open. Or, a file system becomes inaccessible and forces the system to switch to a new file system. 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.

Before You Begin You must be assigned the Audit Review rights profile.

1. **Become an administrator with the required security attributes.**

   For more information, see "How to Obtain Administrative Rights" on page 160.

2. **List the files with the `not_terminated` string on your audit file system in order of creation.**

   ```
   # ls -R1t audit-directory*/* | grep not_terminated
   ```

   `-R` Lists files in subdirectories.

   `-t` Lists files from most recent to oldest.

   `-1` Lists the files in one column.

3. **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
   ```

4. **Remove the old `not_terminated` file.**

   ```
   # rm system-name old-not-terminated-file
   ```

Example 28–35 Cleaning Up Closed not_terminated Audit Files

In the following example, `not_terminated` files are found, renamed, then the originals are removed.

```
ls -R1t */* | grep not_terminated
.../egret.1/20100908162220.not_terminated.egret
```

```
.../egret.1/20100827215359.not_terminated.egret
# cd */egret.1
# auditreduce -O egret 20100908162220.not_terminated.egret
# ls -1t
20100908162220.not_terminated.egret      Current audit file
20100827230920.20100830000909.egret      Cleaned up audit file
20100827215359.not_terminated.egret      Input (old) audit file
# rm 20100827215359.not_terminated.egret
# ls -1t
20100908162220.not_terminated.egret      Current audit file
20100827230920.20100830000909.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, prevent audit record loss.

Before You Begin You must be in the `root` role.

1. **Set a minimum free size on the `audit_binfile` plugin.**

   Use the `p_minfree` attribute.

   The `audit_warn` email alias sends a warning when the disk space fills to the minimum free size. See Example 28–17.

2. **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.

3. **Set up a schedule to delete the archived audit files from the audit file system.**

4. **Save and store auxiliary information.**

   Archive information that is necessary to interpret audit records along with the audit trail. Minimally, you save the `passwd`, `group`, and `hosts` files. You also might archive the `audit_event` and `audit_class` files.

5. **Keep records of which audit files have been archived.**

6. **Store the archived media appropriately.**

7 **Reduce the amount of file system capacity that is required by enabling ZFS compression.**

On a ZFS file system that is dedicated to audit files, compression shrinks the files considerably. For an example, see "How to Compress Audit Files on a Dedicated File System" on page 593.

See also "Interactions Between ZFS Compression, Deduplication, and Encryption Properties" in *Oracle Solaris Administration: ZFS File Systems*.

8 **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 28–28 and Example 28–30.

Troubleshooting the Audit Service (Tasks)

This section covers various auditing error messages, preferences, and the auditing that is provided by other tools. These procedures can help you record required audit events and debug audit problems.

Troubleshooting the Audit Service (Task Map)

The following task map points to procedures for troubleshooting auditing.

Problem	Solution	For Instructions
Why are audit records not being logged when I have configured auditing?	Troubleshoot the audit service.	"How to Determine That Auditing Is Running" on page 583
How can I reduce the amount of audit information that is being collected?	Audit just the events that you want to audit.	"How to Lessen the Volume of Audit Records That Are Produced" on page 585
How can I audit everything that a user does on the system?	Audit one or more users for every command.	"How to Audit All Commands by Users" on page 587
How can I change the audit events that are being recorded and have the change affect existing sessions?	Update a user's preselection mask.	"How to Update the Preselection Mask of Logged In Users" on page 591
How can I locate modifications to particular files?	Audit file modifications, then use the `auditreduce` command to find particular files.	"How to Find Audit Records of Changes to Specific Files" on page 589
How can I reduce the size of my audit files?	Limit the size of the binary audit file.	"How to Limit the Size of Binary Audit Files" on page 593

Problem	Solution	For Instructions
How can I use less file system space for audit files?	Use ZFS quotas and compression.	"How to Compress Audit Files on a Dedicated File System" on page 593
How can I remove audit events from the `audit_event` file?	Correctly update the `audit_event` file.	"How to Prevent the Auditing of Specific Events" on page 592
How can I audit all logins to an Oracle Solaris system?	Audit logins from any system.	"How to Audit Logins From Other Operating Systems" on page 594
Why are auditing records not being kept for my FTP transfers?	Use the appropriate auditing tool for utilities that generate their own logs.	"How to Audit FTP and SFTP File Transfers" on page 595

▼ How to Determine That Auditing Is Running

Auditing is enabled by default. If you believe that auditing has not been disabled, but no audit records are being sent to the active plugin, use the following procedure to isolate the issue.

Before You Begin To modify a system file, you must be in the `root` role. To configure auditing, you must be assigned the Audit Configuration rights profile.

1 **Determine that auditing is running.**

Use any of the following methods:

- **Verify the current audit condition.**

  The following listing indicates that auditing is not running:

  ```
  # auditconfig -getcond
  audit condition = noaudit
  ```

  The following listing indicates that auditing is running:

  ```
  # auditconfig -getcond
  audit condition = auditing
  ```

- **Verify that the audit service is running.**

  The following listing indicates that auditing is not running:

  ```
  # svcs -x auditd
  svc:/system/auditd:default (Solaris audit daemon)
   State: disabled since Sun Oct 10 10:10:10 2010
  Reason: Disabled by an administrator.
     See: http://sun.com/msg/SMF-8000-05
     See: auditd(1M)
     See: audit(1M)
     See: auditconfig(1M)
     See: audit_flags(5)
     See: audit_binfile(5)
     See: audit_syslog(5)
  ```

```
   See: audit_remote(5)
   See: /var/svc/log/system-auditd:default.log
Impact: This service is not running.
```

The following listing indicates that the audit service is running:

```
# svcs auditd
STATE          STIME    FMRI
online         10:10:10 svc:/system/auditd:default
```

If the audit service is not running, enable it. For the procedure, see "How to Enable the Audit Service" on page 571.

2 **Verify that at least one plugin is active.**

```
# audit -v
```

If no plugin is active, make one active.

```
# auditconfig -setplugin audit_binfile active
```

3 **If you created a customized audit class, verify that you assigned events to the class.**

For example, the following list of flags contains the pf class, which Oracle Solaris software did not deliver:

```
# auditconfig -getflags
active user default audit flags = pf,lo(0x0100000000000000,00x0100000000001000)
configured user default audit flags = pf,lo(0x0100000000000000,00x0100000000001000)
```

For a description of creating the pf class, see "How to Add an Audit Class" on page 554.

a. **Verify that the class is defined in the audit_class file.**

The audit class must be defined, and its mask must be unique.

```
# grep pf /etc/security/audit_class      Verify class exists
0x0100000000000000:pf:profile
# grep 0x08000000 /etc/security/audit_class      Ensure mask is unique
0x0100000000000000:pf:profile
```

Replace a mask that is not unique. If the class is not defined, define it. Otherwise, run the auditconfig -setflags command with valid values to reset the current flags.

b. **Verify that events have been assigned to the class.**

Use one of the following methods:

```
# auditconfig -lsevent | egrep " pf|,pf|pf,"
AUE_PFEXEC      116 pf execve(2) with pfexec enabled

# auditrecord -c pf
    List of audit events assigned to pf class
```

If events are not assigned to the class, assign the appropriate events to this class.

4 **If the previous steps did not indicate a problem, review your email and the log files.**

a. **Read the email sent to the `audit_warn` alias.**

The `audit_warn` script sends alert messages to the `audit_warn` email alias. In the absence of a correctly configured alias, the messages are sent to the `root` account.

b. **Review the log files for the audit service.**

The output from the `svcs -s auditd` command lists the full path to the audit logs that the audit service produces. For an example, see the listing in Step 1.

c. **Review the system log files.**

The `audit_warn` script writes `daemon.alert` messages to the `/var/log/syslog` file.

The `/var/adm/messages` file might contain information.

5 **After you locate and fix the problems, enable or restart the audit service.**

```
# audit -s
```

▼ 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.

Before You Begin To preselect audit classes and set audit policy, you must be assigned the Audit Configuration rights profile. To modify system files and to assign audit flags to users, roles, and rights profiles, you must be in the `root` role.

1 **Use the default audit policy.**

Specifically, avoid adding events and audit tokens to the audit trail. The following policies grow the size of the audit trail.

- `arge` policy – Adds environment variables to `execv` audit events.
- `argv` policy – Adds command parameters to `execv` 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 object. 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 516.
- `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 `newgroups` 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 the string 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,129,2,AUE_EXECVE,,mach1,2010-10-14 11:39:22.480 -07:00
path,/usr/bin/ls
attribute,100555,root,bin,21,320271,18446744073709551615
subject,jdoe,root,root,root,root,2404,50036632,82 0 mach1
return,success,0
```

The following is the same record when all policies are turned on:

```
header,1578,2,AUE_EXECVE,,mach1,2010-10-14 11:45:46.658 -07:00
path,/usr/bin/ls
attribute,100555,root,bin,21,320271,18446744073709551615
exec_args,2,ls,/etc/security
exec_env,49,MANPATH=/usr/share/man,USER=jdoe,GDM_KEYBOARD_LAYOUT=us,EDITOR=gedit,
  LANG=en_US.UTF-8,GDM_LANG=en_US.UTF-8,PS1=#,GDMSESSION=gnome,SESSIONTYPE=1,SHLVL=2,
  HOME=/home/jdoe,LOGNAME=jdoe,G_FILENAME_ENCODING=@locale,UTF-8, PRINTER=example-dbl,
...
path,/lib/ld.so.1
attribute,100755,root,bin,21,393073,18446744073709551615
subject,jdoe,root,root,root,root,2424,50036632,82 0 mach1
group,root,other,bin,sys,adm,uucp,mail,tty,lp,nuucp,daemon
return,success,0
zone,global
sequence,197
trailer,1578
```

2 Use the audit_syslog plugin to send some audit events to syslog.

And do not send those audit events to the audit_binfile or audit_remote plugin. This strategy works only if you are not required to keep binary records of the audit events that you send to the syslog logs.

3 Set fewer system-wide audit flags and audit individual users.

Reduce the amount of auditing for all users by reducing the number of audit classes that are audited system-wide.

Use the audit_flags keyword to the roleadd, rolemod, useradd, and usermod commands to audit events for specific users and roles. For examples, see Example 28–18 and the usermod(1M) man page.

Use the always_audit and never_audit properties of the profiles command to audit events for specific rights profiles. For information, see the profiles(1) man page.

Note – Like other security attributes, audit flags are affected by search order. For more information, see "Order of Search for Assigned Security Attributes" on page 199.

---

4 **Create your own customized audit class.**

You can create audit classes at your site. Into these classes, put only those audit events that you need to monitor. For the procedure, see "How to Add an Audit Class" on page 554.

Caution – If you modify existing audit class assignments, your modifications might be kept when you upgrade to a newer version of the Oracle Solaris OS. However, the newer version of the file from Oracle Solaris might include changes that you must manually incorporate into the installation. Carefully review the installation logs. For more information, see the description of preserve=renamenew in the pkg(5) man page.

▼ 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 account and administrative roles. Some sites can require audit records of all commands by all users. Additionally, sites can require that the command arguments and environment be recorded.

Before You Begin To preselect audit classes and set audit policy, you must be assigned the Audit Configuration rights profile. To assign audit flags to users, roles, and rights profiles, you must be in the root role.

1 **Become an administrator with the required security attributes.**

For more information, see "How to Obtain Administrative Rights" on page 160.

2 **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.

```
% auditconfig -lsevent | grep " lo "
AUE_login                        6152 lo login - local
AUE_logout                       6153 lo logout
AUE_telnet                       6154 lo login - telnet
AUE_rlogin                       6155 lo login - rlogin
AUE_rshd                         6158 lo rsh access
AUE_su                           6159 lo su
AUE_rexecd                       6162 lo rexecd
```

```
AUE_passwd                      6163 lo passwd
AUE_rexd                        6164 lo rexd
AUE_ftpd                        6165 lo ftp access
AUE_ftpd_logout                 6171 lo ftp logout
AUE_ssh                         6172 lo login - ssh
AUE_role_login                  6173 lo role login
AUE_newgrp_login                6212 lo newgrp login
AUE_admin_authenticate          6213 lo admin login
AUE_screenlock                  6221 lo screenlock - lock
AUE_screenunlock                6222 lo screenlock - unlock
AUE_zlogin                      6227 lo login - zlogin
AUE_su_logout                   6228 lo su logout
AUE_role_logout                 6229 lo role logout
AUE_smbd_session                6244 lo smbd(1m) session setup
AUE_smbd_logoff                 6245 lo smbd(1m) session logoff
AUE_ClientConnect               9101 lo client connection to x server
AUE_ClientDisconnect            9102 lo client disconn. from x server
% auditconfig -lsevent | egrep " ex |,ex |ex,"
AUE_EXECVE                        23 ex,ps execve(2)
```

- **To audit these classes for administrative roles, modify the roles' security attributes.**

  In the following example, `root` is a role. The site has created three roles, `sysadm`, `auditadm`, and `netadm`. All roles are audited for the success and failure of events in the `ex` and `lo` classes.

  ```
  # rolemod -K audit_flags=lo,ex:no root
  # rolemod -K audit_flags=lo,ex:no sysadm
  # rolemod -K audit_flags=lo,ex:no auditadm
  # rolemod -K audit_flags=lo,ex:no netadm
  ```

- **To audit these classes for all users, set the system-wide flags.**

  ```
  # auditconfig -setflags lo,ex
  ```

  The output appears similar to the following:

  ```
  header,129,2,AUE_EXECVE,,mach1,2010-10-14 12:17:12.616 -07:00
  path,/usr/bin/ls
  attribute,100555,root,bin,21,320271,18446744073709551615
  subject,jdoe,root,root,root,root,2486,50036632,82 0 mach1
  return,success,0
  ```

3. **To record the arguments to commands, add the `argv` policy.**

   ```
   # auditconfig -setpolicy +argv
   ```

   The `exec_args` token records the command arguments:

   ```
   header,151,2,AUE_EXECVE,,mach1,2010-10-14 12:26:17.373 -07:00
   path,/usr/bin/ls
   attribute,100555,root,bin,21,320271,18446744073709551615
   exec_args,2,ls,/etc/security
   subject,jdoe,root,root,root,root,2494,50036632,82 0 mach1
   return,success,0
   ```

4. **To record the environment in which the command is run, add the `arge` policy.**

   ```
   # auditconfig -setpolicy +arge
   ```

The exec_env token records the command environment:

```
header,1460,2,AUE_EXECVE,,mach1,2010-10-14 12:29:39.679 -07:00
path,/usr/bin/ls
attribute,100555,root,bin,21,320271,18446744073709551615
exec_args,2,ls,/etc/security
exec_env,49,MANPATH=/usr/share/man,USER=jdoe,GDM_KEYBOARD_LAYOUT=us,EDITOR=gedit,
LANG=en_US.UTF-8,GDM_LANG=en_US.UTF-8,PS1=#,GDMSESSION=gnome,SESSIONTYPE=1,SHLVL=2,
HOME=/home/jdoe,LOGNAME=jdoe,G_FILENAME_ENCODING=@locale,UTF-8,
PRINTER=example-dbl,...,_=/usr/bin/ls
subject,jdoe,root,root,root,root,2502,50036632,82 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.

Before You Begin You must be assigned the Audit Configuration rights profile to use the auditconfig command. You must be assigned the Audit Review rights profile to use the auditreduce command. To assign audit flags to users and roles, you must be in the root role.

1. **Become an administrator with the required security attributes.**

   For more information, see "How to Obtain Administrative Rights" on page 160.

2. **Audit the fw class.**

   Adding the class to the audit flags of a user or role generates fewer records than adding the class to the system-wide audit preselection mask. Perform one of the following steps:

   - **Add the fw class to specific roles.**

     ```
     # rolemod -K audit_flags=fw:no root
     # rolemod -K audit_flags=fw:no sysadm
     # rolemod -K audit_flags=fw:no auditadm
     # rolemod -K audit_flags=fw:no netadm
     ```

   - **Add the fw class to the system-wide flags.**

     ```
     # auditconfig -getflags
     active user default audit flags = lo(0x1000,0x1000)
     configured user default audit flags = lo(0x1000,0x1000)
     # auditconfig -setflags lo,fw
     user default audit flags = lo,fw(0x1002,0x1002)
     ```

3 **Or, audit successful file-writes.**

Auditing successes generates fewer records than auditing failures and successes. Perform one of the following steps:

- **Add the `+fw` flag to specific roles.**

  ```
  # rolemod -K audit_flags=+fw:no root
  # rolemod -K audit_flags=+fw:no sysadm
  # rolemod -K audit_flags=+fw:no auditadm
  # rolemod -K audit_flags=+fw:no netadm
  ```

- **Add the `+fw` flag to the system-wide flags.**

  ```
  # auditconfig -getflags
  active user default audit flags = lo(0x1000,0x1000)
  configured user default audit flags = lo(0x1000,0x1000)
  # auditconfig -setflags lo,+fw
  user default audit flags = lo,+fw(0x1002,0x1000)
  ```

- **If the system-wide flags are auditing for success and for failure, set exceptions for specific users and roles.**

  ```
  # auditconfig -getflags
  active user default audit flags = lo,fw(0x1002,0x1002)
  configured user default audit flags = lo,fw(0x1002,0x1002)
  # rolemod -K audit_flags=^-fw:no root
  # rolemod -K audit_flags=^-fw:no sysadm
  # rolemod -K audit_flags=^-fw:no auditadm
  # rolemod -K audit_flags=^-fw:no netadm
  ```

  The system-wide flags are still unchanged, but the preselection mask for these four roles is changed.

  ```
  # auditconfig -getflags
  active user default audit flags = lo,fw(0x1002,0x1000)
  configured user default audit flags = lo,fw(0x1002,0x1000)
  ```

4 **To find the audit records for specific files, use the `auditreduce` command.**

```
# 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.

5 **To read the `filechg` file, use the `praudit` command.**

```
# praudit *filechg
```

▼ How to Update the Preselection Mask of Logged In Users

You want the users who are already logged in to be audited for changes to the system-wide audit preselection mask.

Before You Begin You must be assigned the Audit Configuration rights profile. To terminate user sessions, you must be assigned the Process Management rights profile.

1. **Become an administrator with the required security attributes.**

   For more information, see "How to Obtain Administrative Rights" on page 160.

2. **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 preselection masks.

   - **Terminate the users' existing sessions.**

     Users can log out and log back in. Or, you in a role that is assigned the Process Management rights profile can manually terminate (kill) active sessions. The new sessions will inherit the new preselection mask. However, terminating users could be impractical.

   - **Dynamically change each logged-in user's preselection mask.**

     In a role that includes the Audit Configuration rights profile, assume that you changed the system-wide audit preselection mask from `lo` to `lo,ex`.

     ```
     # auditconfig -setflags lo,ex
     ```

     a. **List the regular users who are logged in and their process IDs.**

     ```
     # who -a
     jdoe  - vt/2        Jan 25 07:56  4:10   1597   (:0)
     jdoe  + pts/1       Jan 25 10:10   .     1706   (:0.0)
     ...
     jdoe  + pts/2       Jan 25 11:36  3:41   1706   (:0.0)
     ```

     b. **For later comparison, display each user's preselection mask.**

     ```
     # auditconfig -getpinfo 1706
     audit id = jdoe(1234)
     process preselection mask = lo(0x1000,0x1000)
     terminal id (maj,min,host) = 9426,65559,mach1(192.168.123.234)
     audit session id = 103203403
     ```

c. **Modify the user's preselection mask.**

```
# auditconfig -setumask jdoe lo,ex      /* for this user */

# auditconfig -setsmask 103203403 lo,ex      /* for this session */

# auditconfig -setpmask 1706 lo,ex      /* for this process */
```

d. **Verify that the preselection mask for the user has changed.**

For example, check a process that existed before you changed the mask.

```
# auditconfig -getpinfo 1706
audit id = jdoe(1234)
process preselection mask = ex,lo(0x40001000,0x40001000)
terminal id (maj,min,host) = 9426,65559,mach1(192.168.123.234)
audit session id = 103203403
```

▼ How to Prevent the Auditing of Specific Events

For maintenance purposes, sometimes a site wants to prevent events from being audited.

Before You Begin You must be in the root role.

1 **Change the class of the event to the no class.**

For example, events 26 and 27 belong to the pm class.

```
## 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.

```
## 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 by following the instructions in "How to Update the Preselection Mask of Logged In Users" on page 591.

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 **Refresh the kernel events.**

```
# auditconfig -conf
Configured 283 kernel events.
```

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.

Before You Begin You must be assigned the Audit Configuration rights profile to set the p_fsize attribute. You must be assigned the Audit Review rights profile to use the auditreduce command.

1 **Become an administrator with the required security attributes.**

For more information, see "How to Obtain Administrative Rights" on page 160.

2 **Use the p_fsize attribute to limit the size of individual binary audit files.**

For a description of the p_fsize attribute, see the OBJECT ATTRIBUTES section of the audit_binfile(5) man page.

For an example, see Example 28–14.

3 **Use the auditreduce command to select records and write those records to a smaller 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 Compress Audit Files on a Dedicated File System

Audit files can grow large. You can set an upper limit to the size of a file, as shown in Example 28–14. In this procedure, you use compression to reduce the size.

Before You Begin You must be assigned the ZFS File System Management and ZFS Storage Management rights profiles. The latter profile enables you to create storage pools.

1. **Become an administrator with the required security attributes.**

   For more information, see "How to Obtain Administrative Rights" on page 160.

2. **Dedicate a ZFS file system for audit files.**

   For the procedure, see "How to Create ZFS File Systems for Audit Files" on page 556.

3. **Compress the ZFS storage pool by using one of the following options.**

   With both options, the audit file system is compressed. After the audit service is refreshed, the compression ratio is displayed.

   To set compression, use the `zfs set compression=on` *dataset* command. In the following examples, the ZFS pool `auditp/auditf` is the dataset.

   - **Use the default compression algorithm.**

     ```
     # zfs set compression=on auditp/auditf
     # audit -s
     # zfs get compressratio auditp/auditf
     NAME           PROPERTY       VALUE  SOURCE
     auditp/auditf  compressratio  4.54x  -
     ```

   - **Use a higher compression algorithm.**

     ```
     # zfs set compression=gzip-9 auditp/auditf
     # zfs get compression auditp/auditf
     NAME           PROPERTY     VALUE     SOURCE
     auditp/auditf  compression  gzip-9    local
     # audit -s
     # zfs get compressratio auditp/auditf
     NAME           PROPERTY       VALUE  SOURCE
     auditp/auditf  compressratio  16.89x  -
     ```

     The `gzip-9` compression algorithm results in files that occupy one-third less space than the default compression algorithm, `lzjb`. For more information, see Chapter 6, "Managing Oracle Solaris ZFS File Systems," in *Oracle Solaris Administration: ZFS File Systems*.

▼ How to Audit Logins From Other Operating Systems

The Oracle Solaris OS can audit all logins, independent of source.

Before You Begin You must be assigned the Audit Configuration rights profile.

1. **Become an administrator with the required security attributes.**

   For more information, see "How to Obtain Administrative Rights" on page 160.

2 **Audit the `lo` class for attributable events and non-attributable events.**

This class audits logins, logouts, and screen locks. These classes are audited by default.

```
# auditconfig -getflags
active user default audit flags = lo(0x1000,0x1000)
configured user default audit flags = lo(0x1000,0x1000)
# auditconfig -getnaflags
active non-attributable audit flags = lo(0x1000,0x1000)
configured non-attributable audit flags = lo(0x1000,0x1000)
```

3 **If the values have been changed, add the `lo` flag.**

```
# auditconfig -getflags
active user default audit flags = as,st(0x20800,0x20800)
configured user default audit flags = as,st(0x20800,0x20800)
# auditconfig -setflags lo,as,st
user default audit flags = as,lo,st(0x21800,0x21800)
# auditconfig -getnaflags
active non-attributable audit flags = na(0x400,0x400)
configured non-attributable audit flags = na(0x400,0x400)
# auditconfig -setnaflags lo,na
non-attributable audit flags = lo,na(0x1400,0x1400)
```

Note – To audit `ssh` logins, your system must be running the `ssh` daemon from Oracle Solaris. This daemon is modified for the audit service on an Oracle Solaris system. For more information, see "Secure Shell and the OpenSSH Project" on page 295.

▼ 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 preselecting the `ft` audit class. Logins to both services can be audited.

Before You Begin You must be assigned the Audit Configuration rights profile.

1 **Become an administrator with the required security attributes.**

For more information, see "How to Obtain Administrative Rights" on page 160.

2 **To log commands and file transfers of the FTP service, see the `proftpd`(8) 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.

3 **To log `sftp` access and file transfers, audit the `ft` class.**

The `ft` class includes the following SFTP transactions:

```
% auditrecord -c ft
file transfer: chmod ...
file transfer: chown ...
file transfer: download ...
file transfer: mkdir ...
file transfer: upload ...
file transfer: remove ...
file transfer: rename ...
file transfer: rmdir ...
file transfer: session start ...
file transfer: session end ...
file transfer: symlink ...
file transfer: utimes
```

4 To record access to the FTP server, audit the `lo` class.

As the following output indicates, logging in to and out of the `ftpd` daemon generates audit records.

```
% auditrecord -c lo | more
...
in.ftpd
  program     /usr/sbin/in.ftpd    See ftp access
  event ID    6165                 AUE_ftpd
  class       lo                   (0x0000000000001000)
      subject
      [text]                       error message
      return

in.ftpd
  program     /usr/sbin/in.ftpd    See ftp logout
  event ID    6171                 AUE_ftpd_logout
  class       lo                   (0x0000000000001000)
      subject
      return
...
```

CHAPTER 29

Auditing (Reference)

This chapter describes the important components of auditing. The following is a list of the reference information in this chapter.

- "Audit Service" on page 597
- "Audit Service Man Pages" on page 598
- "Rights Profiles for Administering Auditing" on page 600
- "Auditing and Oracle Solaris Zones" on page 600
- "Audit Classes" on page 601
- "Audit Plugins" on page 602
- "Audit Policy" on page 603
- "Process Audit Characteristics" on page 604
- "Audit Trail" on page 605
- "Conventions for Binary Audit File Names" on page 605
- "Audit Record Structure" on page 606
- "Audit Token Formats" on page 607

For an overview of auditing, see Chapter 26, "Auditing (Overview)." For planning suggestions, see Chapter 27, "Planning for Auditing." For procedures to configure auditing at your site, see Chapter 28, "Managing Auditing (Tasks)."

Audit Service

The audit service, `auditd`, is enabled by default. To enable, refresh, or disable the service, see "Enabling and Disabling the Audit Service (Tasks)" on page 568.

Without customer configuration, the following defaults are in place:

- All login events are audited.

  Both successful and unsuccessful login attempts are audited.
- All users are audited for login and logout events, including role assumption and screenlock.

- The `audit_binfile` plugin is active. The `/var/audit` directory stores audit records, the size of an audit file is not limited, and the queue size is 100 records.
- The `cnt` policy is set.

  When audit records fill the available disk space, the system tracks the number of dropped audit records. A warning is issued when one percent of available disk space remains.
- The following audit queue controls are set:
  - Maximum number of records in the audit queue before generating the records locks - 100
  - Minimum number of records in the audit queue before blocked auditing processes unblock - 10
  - Buffer size for the audit queue - 8192 bytes
  - Interval between writing audit records to the audit trail - 20 seconds

To display the defaults, see "How to Display Audit Service Defaults" on page 543.

The audit service enables you to set temporary, or active, values. These values can differ from configured, or property, values.

- Temporary values are not restored when you refresh or restart the audit service.

  Audit policy and audit queue controls accept temporary values. Audit flags do not have a temporary value.
- Configured values are stored as property values of the service, so they are restored when you refresh or restart the audit service.

Rights profiles control who can administer the audit service. For more information, see "Rights Profiles for Administering Auditing" on page 600.

By default, all zones are audited identically. See "Auditing and Oracle Solaris Zones" on page 600.

Audit Service Man Pages

The following table summarizes the major administrative man pages for the audit service.

Man Page	Summary
audit(1M)	Command that controls the actions of the audit service audit -n starts a new audit file for the audit_binfile plugin. audit -s enables and refreshes auditing. audit -t disables auditing. audit -v verifies that at least one plugin is active.
audit_binfile(5)	Default audit plugin, which sends audit records to a binary file. See also "Audit Plugins" on page 602.
audit_remote(5)	Audit plugin that sends audit records to a remote receiver.
audit_syslog(5)	Audit plugin that sends text summaries of audit records to the syslog utility.
audit_class(4)	File that contains the definitions of audit classes. The eight high-order bits are available for customers to create new audit classes. For the effect of modifying this file on system upgrade, see "How to Add an Audit Class" on page 554.
audit_event(4)	File that contains the definitions of audit events and maps the events to audit classes. The mapping can be modified. For the effect of modifying this file on system upgrade, see "How to Change an Audit Event's Class Membership" on page 555.
audit_flags(5)	Describes the syntax of audit class preselection, the prefixes for selecting only failed events or only successful events, and the prefixes that modify an existing preselection.
audit.log(4)	Describes the naming of binary audit files, the internal structure of a file, and the structure of every audit token.
audit_warn(1M)	Script that notifies an email alias when the audit service 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.
auditconfig(1M)	Command that retrieves and sets audit configuration parameters. Type auditconfig with no options for a list of parameters that can be retrieved and set.
auditrecord(1M)	Command that displays the definition of audit events in the /etc/security/audit_event file. For sample output, see "How to Display Audit Record Definitions" on page 573.
auditreduce(1M)	Command that post-selects and merges audit records that are stored in binary format. The command can merge audit records from one or more input audit files. The records remain in binary format. Uppercase options affect file selection. Lowercase options affect record selection.

Man Page	Summary
auditstat(1M)	Command that displays kernel audit statistics. For example, the command can display the number of records in the kernel audit queue, the number of dropped records, and the number of audit records that user processes produced in the kernel as a result of system calls.
praudit(1M)	Command that 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 or a list of audit files. Input can also be produced with the tail -0f command for a current audit file. For sample output, see "How to View the Contents of Binary Audit Files" on page 578.
syslog.conf(4)	File that is configured to send text summaries of audit records to the syslog utility for the audit_syslog plugin.

Rights Profiles for Administering Auditing

Oracle Solaris provides rights profiles for configuring the audit service, for enabling and disabling the service, and for analyzing the audit trail. To edit an audit configuration file requires the privileges of root.

- **Audit Configuration –** Enables an administrator to configure the parameters of the audit service and to run the auditconfig command.
- **Audit Control –** Enables an administrator to start, refresh, and disable the audit service and to run the audit command to start, refresh, or stop the service.
- **Audit Review –** Enables an administrator to analyze audit records. This rights profile grants authorization to read audit records with the praudit and auditreduce commands. This administrator can also run the auditstat command.
- **System Administrator –** Includes the Audit Review rights profile. An administrator with the System Administrator rights profile can analyze audit records.

To configure roles to handle the audit service, see "Initially Configuring RBAC (Task Map)" on page 163.

Auditing and Oracle Solaris Zones

Non-global zones can be audited exactly as the global zone is audited, or non-global zones can set their own flags, storage, and audit policy.

When all zones are being audited identically, the audit_class and audit_event files in the global zone provide the class-event mappings for auditing in every zone. The +zonename policy option is useful for post-selecting records by zone name.

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 service, 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 Oracle Solaris Zones" on page 526 and "How to Plan Auditing in Zones" on page 530.

To learn about zones, see Part II, "Oracle Solaris Zones," in *Oracle Solaris Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management*.

Audit Classes

Oracle Solaris defines audit classes as convenient containers for large numbers of audit events.

You can reconfigure 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. For more information, see the audit_class(4) man page and "How to Add an Audit Class" on page 554.

Caution – The all class can generate large amounts of data and quickly fill disks. Use the all class only if you have extraordinary reasons to audit all activities.

Audit Class Syntax

Events in an audit class can be audited for success, for failure, and 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.
- With a caret (^) preceding a prefix or an audit flag, a current preselection is modified. For example,
  - If ot is preselected for the system, and a user's preselection is ^ot, that user is not audited for events in the other class.
  - If +ot is preselected for the system, and a user's preselection is ^+ot, that user is not audited for successful events in the other class.
  - If -ot is preselected for the system, and a user's preselection is ^-ot, that user is not audited for failed events in the other class.

To review the syntax of audit class preselection, see the audit_flags(5) man page.

The audit classes and their prefixes can be specified in the following commands:

- As arguments to the auditconfig command options -setflags and -setnaflags.
- As values for the p_flags attribute to the audit_syslog plugin. You specify the attribute as an option to the auditconfig -setplugin audit_syslog active command.
- As values for the -K audit_flags=*always-audit-flags*:*never-audit-flags* option to the useradd, usermod, roleadd, and rolemod commands.
- As values for the -always_audit and -never_audit properties of the profiles command.

Audit Plugins

Audit plugins specify how to handle the audit records in the audit queue. The audit plugins are specified by name: audit_binfile, audit_remote, and audit_syslog, as arguments to the auditconfig -setplugin command. The plugins can be further specified by the following attributes:

- audit_binfile plugin

  Where to send binary data - p_dir attribute

  The minimum space remaining on a disk before the administrator is warned - p_minfree attribute

  The maximum size of an audit file - p_fsize attribute
- audit_remote plugin

  A remote authenticated audit server to send the binary audit data to - p_hosts attribute

  The number of attempts to make to reach a remote authenticated audit server - p_retries attribute

  The number of seconds between attempts to reach a remote authenticated audit server - p_timeout attribute
- audit_syslog plugin

  A selection of text summaries of audit records to be sent to syslog - p_flags attribute
- For all plugins, the maximum number of audit records that are queued for the plugin - qsize attribute

Refer to the audit_binfile(5), audit_remote(5), audit_syslog(5), and auditconfig(1M) 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 `windata_down` and `windata_up` policies are used by the Trusted Extensions feature of Oracle Solaris. For more information, see Chapter 22, "Trusted Extensions Auditing (Overview)," in *Trusted Extensions Configuration and Administration*.

The remaining policies do not add tokens. The `public` policy limits auditing of public files. The `perzone` policy establishes separate audit queues for non-global zones. The `ahlt` and `cnt` policies determine what happens when audit records cannot be delivered. For details, see "Audit Policies for Asynchronous and Synchronous Events" on page 603.

The effects of the different audit policy options are described in "Understanding Audit Policy" on page 534. 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 command `auditconfig -lspolicy`. For the current policy, run the command `auditconfig -getpolicy`.

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.

Note – The `cnt` or `ahlt` policy is not triggered if the queue for at least one plugin can accept audit records.

The `cnt` and `ahlt` 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 asynchronous 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.

Process Audit Characteristics

The following audit characteristics are set at initial login:

- **Process preselection mask –** A combination of the system-wide audit mask and the user-specific audit mask, if a user audit mask has been specified. 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:

(system-wide default flags + *always-audit-classes*) - *never-audit-classes*

Add the system-wide audit classes from the results of the `auditconfig -getflags` command to the classes from the *always-audit-classes* value for the user's `always_audit` keyword. Then, from the total subtract the classes from the user's *never-audit-classes*. See also the `audit_flags`(5) man page.

- **Audit user ID –** A process acquires an immutable audit user ID when the user logs in. This ID is inherited by all child processes that were started by the user's initial process. The audit user ID helps enforce accountability. Even after a user assumes a role, the audit user ID remains the same. The audit user ID that is saved in each audit record enables you to always trace actions back to the login user.
- **Audit session ID –** The audit session ID is assigned at login. This ID is inherited by all child processes.
- **Terminal ID –** For a local login, the terminal ID consists of the local system's IP 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 `0,0`. For a remote login, the terminal ID consists of a the remote host's IP address followed by the remote port number and the local port number.

Audit Trail

The *audit trail* contains binary audit files. The trail is created by the `audit_binfile` plugin. The audit service collects the audit trail records and sends them to the plugin, which writes them to disk.

Conventions for Binary Audit File Names

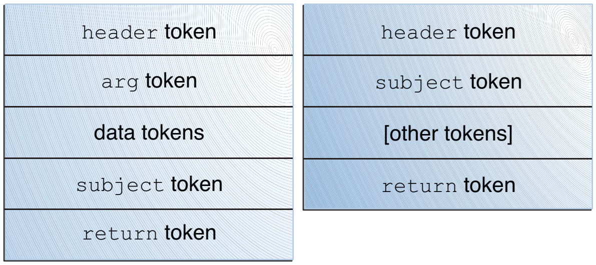
The `audit_binfile` plugin creates binary audit files. Each binary audit file is a self-contained collection of records. The file's name identifies the time span during which the records were generated and the system that generated them. The time stamps that indicate the time span are specified in Coordinated Universal Time (UTC) to ensure that they sort in proper order, even across time zones.

For more information, see the `audit.log`(4) man page. For examples of open and closed audit file names, see "How to Clean Up a `not_terminated` Audit File" on page 580.

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 kernel audit record and a typical user-level audit record.

FIGURE 29–1 Typical Audit Record Structures



Audit Record Analysis

Audit record analysis involves post-selecting records from the audit trail. You can use one of two approaches to parsing the binary data that was collected.

- You can use the `praudit` command. Options to the command provide different text output. For example, the `praudit -x` command provides XML for input into scripts and browsers. `praudit` output does not include fields whose sole purpose is to help to parse the binary data. Note that the order and format of `praudit` output is not guaranteed between Oracle Solaris releases.

  For examples of `praudit` output, see "How to View the Contents of Binary Audit Files" on page 578.

  For examples of `praudit` output for each audit token, see the individual tokens in "Audit Token Formats" on page 607.

- You can write a program to parse the binary data stream. The program must take into account the variants of an audit record. For example, the `ioctl()` system call creates an audit record for "Bad file name". This record contains different tokens from the `ioctl()` 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 manifest values, see the `/usr/include/bsm/audit.h` file.

- To view the order of tokens in an audit record, use the auditrecord command. Output from the auditrecord command includes the different tokens for different manifest values. Square brackets ([]) indicate that an audit token is optional. For more information, see the auditrecord(1M) man page.

Audit Token Formats

Each audit token has a token type identifier, which is followed by data that is specific to the token. 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 29–1 Audit Tokens for Auditing

Token Name	Description	For More Information
acl	Access Control Entry (ACE) and Access Control List (ACL) information	"acl Token" on page 609
arbitrary	Data with format and type information	See the audit.log(4) man page.
argument	System call argument value	"argument Token" on page 609
attribute	File vnode information	"attribute Token" on page 609
cmd	Command arguments and environment variables	"cmd Token" on page 609
exec_args	Exec system call arguments	"exec_args Token" on page 610
exec_env	Exec system call environment variables	"exec_env Token" on page 610
exit	Program exit information	See the audit.log(4) man page.
file	Audit file information	"file Token" on page 610
fmri	Framework Management Resource Indicator	"fmri Token" on page 611
group	Process groups information	"group Token" on page 611
header	Indicates start of audit record	"header Token" on page 611
ip	IP header information	See the audit.log(4) man page.
ip address	Internet address	"ip address Token" on page 612
ip port	Internet port address	"ip port Token" on page 612
ipc	System V IPC information	"ipc Token" on page 612
IPC_perm	System V IPC object access information	"IPC_perm Token" on page 613
opaque	Unstructured data (unspecified format)	See the audit.log(4) man page.
path	Path information	"path Token" on page 613

TABLE 29–1 Audit Tokens for Auditing *(Continued)*

Token Name	Description	For More Information
path_attr	Access path information	"path_attr Token" on page 613
privilege	Privilege set information	"privilege Token" on page 614
process	Process information	"process Token" on page 614
return	Status of system call	"return Token" on page 614
sequence	Sequence number	"sequence Token" on page 614
socket	Socket type and addresses	"socket Token" on page 615
subject	Subject information (same format as process)	"subject Token" on page 615
text	ASCII string	"text Token" on page 615
trailer	Indicates end of audit record	"trailer Token" on page 616
use of authorization	Use of authorization	"use of authorization Token" on page 616
use of privilege	Use of privilege	"use of privilege Token" on page 616
user	User ID and user name	"user Token" on page 616
xclient	X client identification	"xclient Token" on page 617
zonename	Name of zone	"zonename Token" on page 617
Trusted Extensions tokens	label and X Window System information	See "Trusted Extensions Audit Reference" in *Trusted Extensions Configuration and Administration*.

The following tokens are obsolete:

- liaison
- host
- tid

For information about obsolete tokens, see the reference material for the release that included the token.

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 non-attributable events, the process token refers to the system.

acl Token

The acl token has two forms to record information about Access Control Entries (ACEs) for a ZFS file system and Access Control Lists (ACLs) for a UFS file system.

When the acl token is recorded for a UFS file system, the praudit -x command shows the fields as follows:

```
<acl type="1" value="root" mode="6"/>
```

When the acl token is recorded for a ZFS dataset, the praudit -x command shows the fields as follows:

```
<acl who="root" access_mask="default" flags="-i,-R" type="2"/>
```

argument Token

The argument 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 argument token as follows:

```
<argument arg-num="2" value="0x5401" desc="cmd"/>
```

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 as follows:

```
<attribute mode="20620" uid="root" gid="tty" fsid="0" nodeid="9267" device="108233"/>
```

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.

```
<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 `praudit -x` command shows the fields of the `exec_args` token as follows:

```
<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 `praudit -x` command shows the fields of the `exec_env` token. The line is wrapped for display purposes.

```
<exec_env><env>_=/usr/bin/hostname</env>
<env>LANG=C</env><env>PATH=/usr/bin:/usr/ucb</env>
<env><env>LOGNAME=jdoe</env><env>USER=jdoe</env>
<env>DISPLAY=:0</env><env>SHELL=/bin/csh</env>
<env>HOME=/home/jdoe</env><env>PWD=/home/jdoe</env><env>TZ=US/Pacific</env>
</exec_env>
```

Note – The `exec_env` token is output only when the `arge` audit policy option is active.

file Token

The `file` token is a special token that 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. These tokens "link" together successive audit files into one audit trail.

The `praudit -x` command shows the fields of the `file` token. 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>
```

group Token

The group token records the group entries from the process's credential.

The praudit -x command shows the fields of the groups token as follows:

```
<group><gid>staff</gid><gid>other</gid></group>
```

Note – The group 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.

Infrequently, a header token can include one or more event modifiers:

- fe indicates a failed audit event
- fp indicates the failed use of privilege
- na indicates a non-attributable event

  ```
  header,52,2,system booted,na,mach1,2011-10-10 10:10:20.564 -07:00
  ```
- rd indicates that data is read from the object
- sp indicates the successful use of privilege

  ```
  header,120,2,exit(2),sp,mach1,2011-10-10 10:10:10.853 -07:00
  ```
- wr indicates that data is written to the object

The praudit command displays the header token as follows:

```
header,756,2,execve(2),,machine1,2010-10-10 12:11:10.209 -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="execve(2)" host="machine1"
iso8601="2010-10-10 12:11:10.209 -07:00">
```

ip address Token

The ip address token contains an Internet Protocol address (IP address). The IP address can be displayed in IPv4 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 ip address token as follows:

```
<ip_address>machine1</ip_address>
```

ip port Token

The ip port token contains the TCP or UDP port address.

The praudit command displays the ip port token as follows:

```
ip port,0xf6d6
```

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 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 29–2 Values for the IPC Object Type Field

Name	Value	Description
AU_IPC_MSG	1	IPC message object
AU_IPC_SEM	2	IPC semaphore object

TABLE 29–2 Values for the IPC Object Type Field (Continued)

Name	Value	Description
AU_IPC_SHM	3	IPC shared-memory object

The praudit -x command shows the fields of the ipc token as follows:

```
<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.

path Token

The path token contains access path information for an object.

The praudit -x command shows the content of the path token:

```
<path>/export/home/srv/.xsession-errors</path>
```

path_attr Token

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 about 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
```

privilege Token

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 about privileges, see "Privileges (Overview)" on page 145

The praudit -x command shows the fields of the privilege token.

```
<privilege set-type="Inheritable">ALL</privilege>
```

process Token

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="567" sid="0" tid="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 as follows:

```
<return errval="failure: Operation now in progress" retval="-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 includes only the remote port and remote IP address.

The praudit command displays this instance of the socket token as follows:

```
socket,0x0002,0x83b1,localhost
```

The expanded token adds information, including socket type and local port information.

The praudit -x command displays this instance of the socket token as follows. The line is wrapped for display purposes.

```
<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:

```
subject,jdoe,root,root,root,root,1631,1421584480,8243 65558 machine1
```

The praudit -x command shows the fields of the subject token. The line is wrapped for display purposes.

```
<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:

```
<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 auditreduce 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
```

use of authorization Token

The use of authorization token records the use of authorization.

The praudit command displays the use of authorization token as follows:

```
use of authorization,solaris.role.delegate
```

```
XXXX<use_of_authorization result="successful use of auth">solaris.role.delegate</use_of_auth>
```

use of privilege Token

The use of privilege token records the use of privilege.

The praudit -x command shows the fields of the use of privilege token as follows:

```
<use_of_privilege result="successful use of priv">proc_setid</use_of_privilege>
```

user Token

The user token records the user name and user ID. This token is present if the user name is different from the caller.

The praudit -x command shows the fields of the user token as follows:

```
<user uid="123456" username="tester1"/>
```

xclient Token

The `xclient` token contains the number of the client connection to the X server.

The `praudit -x` command shows the content of the `xlient` token as follows:

```
<X_client>15</X_client>
```

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 Advanced Encryption Standard. A symmetric 128-bit block data encryption technique. The U.S. government adopted the Rijndael variant of the algorithm as its encryption standard in October 2000. AES replaces user principal encryption as the government standard.

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 file system.

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 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 Cryptographic Framework feature of Oracle Solaris, 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 Cryptographic Framework feature of Oracle Solaris, 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 system 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 Oracle 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.

least privilege A security model which gives a specified process only a subset of superuser powers. The least privilege model assigns enough privilege to regular users that they can perform personal administrative tasks, such as mount file systems and change the ownership of files. On the other hand, processes run with just those privileges that they need to complete the task, rather than with the full power of superuser, that is, all privileges. Damage due to programming errors like buffer overflows can be contained to a non-root user, which has no access to critical abilities like reading or writing protected system files or halting the machine.

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 system address on a LAN. If the system 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 Cryptographic Framework feature of Oracle Solaris, 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 naming service such as NIS or LDAP.

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 Oracle 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 Cryptographic Framework feature of Oracle Solaris, 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.

primary The first part of a principal name. See also instance, principal name, realm.

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 an Oracle 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-aware Programs, scripts, and commands that turn on and off the use of privilege in their code. In a production environment, the privileges that are turned on must be supplied to the process, for example, by requiring users of the program to use a rights profile that adds the privileges to the program. For a full description of privileges, see the `privileges(5)` man page.

privilege escalation Gaining access to resources that are outside the range of resources that your assigned security attributes, including overrides, permit. The result is that a process can perform unauthorized actions.

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 set, permitted set 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 Cryptographic Framework feature of Oracle Solaris, 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 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.

public object A file that is owned by the `root` user and readable by the world, such as any file in the `/etc` directory.

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, a feature of the Oracle Solaris. 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, `solaris` is the valid policy. The `solaris` policy recognizes privileges, authorizations, and `setuid` security attributes.

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.

rights profile 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, privileges, commands with security attributes, and other rights profiles.

role A special identity for running privileged applications that only assigned users can assume.

RSA A method for obtaining digital signatures and public key cryptosystems. The method was first described in 1978 by its developers, Rivest, Shamir, and Adleman.

scan engine A third-party application, residing on an external host, that examines a file for known viruses.

SEAM 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.

secret key See private key.

Secure Shell A special protocol for secure remote login and other secure network services over an insecure network.

security attributes 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.

security flavor See flavor.

security mechanism See mechanism.

security policy See policy.

security service See service.

seed A numeric starter for generating random numbers. When the starter originates from a random source, the seed is called a *random seed*.

separation of duty Part of the notion of least privilege. Separation of duty prevents one user from performing or approving all actions that complete a transaction. For example, in RBAC, you can separate the creation of a login user from the assignment of security overrides. One role creates the user. A separate role can assign security attributes, such as rights profiles, roles, and privileges to existing users.

server A principal that provides a resource to network clients. For example, if you `ssh` to the system `central.example.com`, then that system is the server that provides the `ssh` service. See also service principal.

server principal (RPCSEC_GSS API) A principal that provides a service. The server principal is stored as an ASCII string in the form *service@host*. See also client principal.

service 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.

2. A security service (either integrity or privacy) that provides a level of protection beyond authentication. See also integrity and privacy.

service key 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.

service principal 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.

session key 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.

SHA1 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.

single-system image A single-system image is used in Oracle Solaris auditing to describe a group of audited systems that use the same naming service. These systems send their audit records to a central audit server, where the records can be compared as if the records came from one system.

slave KDC 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.

software provider In the Cryptographic Framework feature of Oracle Solaris, a kernel software module or a PKCS #11 library that provides cryptographic services. See also provider.

stash file 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 `kadmind` 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.

superuser model The typical UNIX model of security on a computer system. In the superuser model, an administrator has all-or-nothing control of the system. Typically, to administer the machine, a user becomes superuser (`root`) and can do all administrative activities.

synchronous audit event 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.

TGS Ticket-Granting Service. That portion of the KDC that is responsible for issuing tickets.

TGT Ticket-Granting Ticket. A ticket that is issued by the KDC that enables a client to request tickets for other services.

ticket 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.

ticket file See credential cache.

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, 191
[] (square brackets), auditrecord output, 607
* (asterisk)
  checking for in RBAC authorizations, 173
  device_allocate file, 92
  wildcard character
    in RBAC authorizations, 200
@ (at sign), device_allocate file, 92
\ (backslash)
  device_allocate file, 92
  device_maps file, 91
^ (caret), audit class prefix modifier, 601–602
^ (caret) in audit class prefixes, 546–549, 590
. (dot)
  authorization name separator, 200
  displaying hidden files, 122
= (equal sign), file permissions symbol, 118
- (minus sign)
  audit class prefix, 601–602
  file permissions symbol, 118
  file type symbol, 114
  sulog file, 64
+ (plus sign)
  audit class prefix, 601–602
  file permissions symbol, 118
  sulog file, 64
+ (plus sign) in audit class prefixes, 563
# (pound sign)
  device_allocate file, 92
  device_maps file, 91
; (semicolon), device_allocate file, 92
> (redirect output), preventing, 46
>> (append output), preventing, 46
~/.gkadmin file, description, 497
~/.k5login file, description, 497
~/.rhosts file, description, 321
~/.shosts file, description, 321
~/.ssh/authorized_keys file
  description, 320
  override, 322
~/.ssh/config file
  description, 321
  override, 321
~/.ssh/environment file, description, 321
~/.ssh/id_dsa file, override, 322
~/.ssh/id_rsa file, override, 322
~/.ssh/identity file, override, 322
~/.ssh/known_hosts file
  description, 320
  override, 322
~/.ssh/rc file, description, 321
32–bit executables, protecting from compromising security, 121
3des-cbc encryption algorithm, ssh_config file, 315
3des encryption algorithm, ssh_config file, 314

A

-A option, auditreduce command, 575–576
-a option
  auditrecord command, 573

- `-a` option *(Continued)*
  - `digest` command, 229
  - `encrypt` command, 232
  - Kerberized commands, 491
  - `mac` command, 230
- absolute mode
  - changing file permissions, 118, 125–126
  - changing special file permissions, 126–127
  - description, 118
  - setting special permissions, 119
- access
  - control lists
    - *See* ACL
  - getting to server
    - with Kerberos, 506–508
  - granting to your account, 488–490
  - login authentication with Secure Shell, 305–306
  - obtaining for a specific service, 508
  - restricting for
    - devices, 42–44, 76
    - system hardware, 66–68
  - restricting for KDC servers, 422–423
  - `root` access
    - displaying attempts on console, 65–66
    - monitoring `su` command attempts, 45, 64–65
    - restricting, 50, 65–66
  - Secure RPC authentication, 267
  - security
    - ACLs, 49
    - controlling system usage, 45–48
    - devices, 76
    - file access restriction, 46
    - firewall setup, 52–53, 53
    - login access restrictions, 38
    - login authentication, 305–306
    - login control, 38
    - monitoring system usage, 48
    - network control, 50–54
    - NFS client-server, 269–271
    - `PATH` variable setting, 46
    - peripheral devices, 43
    - physical security, 38
    - remote systems, 293
    - reporting problems, 54
- access, security *(Continued)*
    - `root` login tracking, 45
    - saving failed logins, 59–60
    - `setuid` programs, 47
    - system hardware, 66–68
    - UFS ACLs, 119–120
  - sharing files, 49
- access control list
  - *See* ACL
- Access Control Lists (ACLs), *See* ACL
- ACL
  - description, 49, 119–120
  - format of entries, 119–120
  - `kadm5.acl` file, 454, 456, 460
  - restrictions on copying entries, 120
- `acl` audit token, format, 609
- active audit policy, temporary audit policy, 549–551
- `add_drv` command, description, 88
- adding
  - administration principals (Kerberos), 356, 362
  - allocatable device, 80
  - audit classes, 554
  - audit file systems, 556–559
  - audit policy, 549–551
  - auditing
    - of individual users, 546–549, 586
    - of roles, 169–170
    - of zones, 529–534
  - cryptomgt role, 168–169
  - DH authentication to mounted file systems, 272
  - hardware provider mechanisms and features, 247
  - library plugin, 241
  - new rights profile, 170–171
  - PAM modules, 281
  - plugins
    - auditing, 562–563, 563–564
    - Cryptographic Framework, 240–241
    - KMF, 262–263
  - privileged users, 180
  - privileges
    - directly to role, 179
    - directly to user, 180–181
    - to command, 171

adding *(Continued)*
- RBAC properties
  - to legacy applications, 171–173
- roles, 165–167
- security attributes
  - to legacy applications, 171–173
  - to roles, 178–179
  - to users, 179–181
- security-related role, 168–169
- security to devices, 77–78, 79–84
- security to system hardware, 66–67
- service principal to keytab file (Kerberos), 474–475
- software provider, 240–241
- temporary audit policy, 551
- user-level software provider, 241

`admin_server` section
- `krb5.conf` file, 354, 361

administering
- auditing
  - `audit -s` command, 568–570, 571–572
  - `audit -t` command, 571
  - audit classes, 519–520
  - audit events, 518
  - audit files, 578–580
  - audit records, 520
  - `audit_remote` plugin, 562–563
  - `audit_syslog` plugin, 563–564
  - audit trail overflow prevention, 581–582
  - `auditconfig` command, 542–543, 545–546
  - `auditreduce` command, 574–576
  - configuring, 542–543
  - cost control, 537
  - description, 526
  - disabling, 571
  - efficiency, 538
  - enabling, 571–572
  - plugins, 562–563
  - policy, 549–551
  - `praudit` command, 578–580
  - queue controls, 551–552
  - reducing space requirements, 537–538
  - refreshing, 568–570
  - rights profiles required, 600
  - task map, 541

administering, auditing *(Continued)*
  - in zones, 526–527, 600–601
  - zones, 565–568
- auditing in zones, 530–531
- Cryptographic Framework and zones, 219
- Cryptographic Framework commands, 217
- Cryptographic Framework task map, 235
- device allocation, 79–80
- device policy, 76
- file permissions, 121–130
- Kerberos
  - keytabs, 473–479
  - policies, 461–468
  - principals, 448–461
- metaslot, 217
- NFS client-server file security, 269–271
- password algorithms, 61–64
- privileges, 191
- RBAC properties, 170–171
- remote logins with Secure Shell, 302–304
- rights profiles, 170–171
  - of a user, 184
- role password, 177–178
- roles to replace superuser, 163–165
- Secure RPC task map, 272
- Secure Shell
  - clients, 313–314
  - overview, 311–313
  - servers, 314
  - task map, 297
- security properties
  - of a legacy application, 171–173
  - of a rights profile, 170–171
  - of a role, 177–178, 178–179, 183–184
  - of a user, 179–181
- user password to assume role, 183–184
- user password to use rights profile, 184
- without privileges, 147

administrators
- restricting rights, 182–183
- restricting users' rights, 181–182

`AES` kernel provider, 236

`aes128-cbc` encryption algorithm, `ssh_config`
- file, 315

aes128-ctr encryption algorithm, ssh_config file, 315
agent daemon, Secure Shell, 305–306
ahlt audit policy
  description, 535
  setting, 550
  with cnt policy, 603–604
algorithms
  definition in Cryptographic Framework, 215
  file encryption, 232–235
  listing in the Cryptographic Framework, 236–239
  passphrase protection in ssh-keygen, 295
  password
    configuration, 61–62
  password encryption, 40, 61–64
All (RBAC), rights profile, 198
all audit class, caution for using, 601
allocate command
  allocate error state, 91
  authorizations required, 90, 205
  tape drive, 85
  user authorization, 80
  using, 84–85
allocate error state, 90–91
allocating devices
  by users, 84–85
  forcibly, 82
  troubleshooting, 85
AllowGroups keyword, sshd_config file, 314
AllowTcpForwarding keyword
  changing, 300
  sshd_config file, 314
AllowUsers keyword, sshd_config file, 314
ALTSHELL in Secure Shell, 319
always-audit classes, process preselection mask, 604
antivirus software, *See* virus scanning
appending arrow (>>), preventing appending, 46
application server, configuring, 372–375
arcfour encryption algorithm, ssh_config file, 315
ARCFOUR kernel provider, 236
Archive tape drive device-clean script, 93
archiving, audit files, 581–582
arge audit policy
  and exec_env token, 610
arge audit policy *(Continued)*
  description, 535
  setting, 588
argument audit token, format, 609
argv audit policy
  and exec_args token, 610
  description, 535
  setting, 588
assigning
  privileges to commands in a rights profile, 171
  privileges to commands in a script, 194–195
  privileges to role, 179
  privileges to user, 180–181
  rights profile
    to a role, 178–179
  role to a user locally, 167–169
assuming role
  how to, 163–176
  in a terminal window, 159–160
  root, 160
asterisk (*)
  checking for in RBAC authorizations, 173
  device_allocate file, 92
  wildcard character
    in RBAC authorizations, 200
asynchronous audit events, 603–604
at command, authorizations required, 205
at sign (@), device_allocate file, 92
atq command, authorizations required, 205
attribute audit token, 609
attributes, keyword in BART, 110
audio devices, security, 94
audit -s command, 568–570, 571–572
audit -t command, 571
audit_binfile plugin, 520–521
  getting attributes, 560–561, 561
  limiting audit file size, 560–561
  removing queue size, 561
  setting attributes, 559–562
  setting free space warning, 561–562
audit characteristics
  audit user ID, 605
  processes, 604–605
  session ID, 605

audit characteristics *(Continued)*
- terminal ID, 605
- user process preselection mask, 604

audit_class file
- adding a class, 554
- troubleshooting, 554

audit classes
- adding, 554
- configuration, 601–602
- description, 516, 518
- displaying defaults, 543–544
- exceptions to system-wide settings, 519
- mapping events, 520
- modifying default, 554
- overview, 519–520
- post-selection, 517
- prefixes, 601–602
- preselecting
  - effect on public objects, 518
  - for failure, 547, 563, 564
  - for success, 547, 563, 564
  - for success and failure, 545–546
- preselection, 517
- process preselection mask, 604
- replacing, 545–546
- syntax, 601
- user exceptions, 546–549

audit command
- disabling audit service, 571
- options, 599
- refreshing audit service, 568–570

Audit Configuration rights profile, 600
- auditing a role, 169–170
- configuring audit policy, 549–551
- displaying auditing defaults, 543–544
- preselecting audit classes, 545–546

Audit Control rights profile, 600
- disabling audit service, 571
- enabling audit service, 571–572
- refreshing audit service, 568–570

audit directory, creating file systems for, 556–559

audit_event file
- changing class membership, 555–556
- description, 518

audit_event file *(Continued)*
- removing events safely, 592–593

audit event-to-class mappings, changing, 555–556

audit events
- asynchronous, 603–604
- audit_event file, 518
- changing class membership, 555–556
- description, 518
- mapping to classes, 520
- removing from audit_event file, 592–593
- selecting from audit trail, 576–578
- selecting from audit trail in zones, 601
- summary, 516
- synchronous, 603–604
- viewing from binary files, 578–580

audit file system, description, 516

audit files
- combining, 574–576
- compressing on disk, 593–594
- copying messages to single file, 577–578
- creating summary files, 577
- effects of Coordinated Universal Time (UTC), 574
- limiting size of, 593
- managing, 581–582
- printing, 579
- reading with praudit, 578–580
- reducing, 574–576
- reducing space requirements, 537–538
- reducing storage-space requirements, 538
- setting aside disk space for, 556–559
- time stamps, 605
- ZFS file systems, 556–559, 593–594

audit flags, summary, 516

audit_flags keyword, 544
- specifying user exceptions to audit preselection, 546–549
- use, 602
- using caret (^) prefix, 547

audit logs
- *See also* audit files
- comparing binary and text summaries, 521
- configuring, 556–564
- configuring text summary audit logs, 563–564
- modes, 521

audit.notice entry, syslog.conf file, 563
audit plugins
  audit_binfile plugin, 551–552, 559–562
  audit_remote plugin, 562–563
  audit_syslog plugin, 563–564
  description, 516
  qsize attribute, 551–552
  summary, 598–600, 602
audit policy
  audit tokens from, 603
  defaults, 534–537
  description, 517
  displaying defaults, 543–544
  effects of, 534–537
  public, 536
  setting, 549–551
  setting ahlt, 550
  setting arge, 588
  setting argv, 588
  setting in global zone, 526–527, 600–601
  setting perzone, 551
  that does not affect tokens, 603
  tokens added by, 603
audit preselection mask
  modifying for existing users, 591–592
  modifying for individual users, 546–549
audit queue, events included, 519
audit queue controls
  displaying defaults, 543–544
  getting, 551–552
audit records
  converting to readable format, 579
  copying to single file, 577–578
  description, 517
  displaying, 578–580
  displaying definitions of
    procedure, 573–574
  displaying formats of a program, 573–574
  displaying formats of an audit class, 574
  displaying in XML format, 579
  event modifiers, 611
  events that generate, 524
  format, 606
  formatting example, 573
audit records *(Continued)*
  merging, 574–576
  overview, 520
  reducing audit files, 574–576
  sequence of tokens, 606
  /var/adm/auditlog file, 563
audit_remote plugin, 520–521
  getting attributes, 562–563
  setting attributes, 562–563
Audit Review rights profile, 600
audit service
  See also auditing
  audit trail creation, 605
  configuring policy, 549–551
  configuring queue controls, 551–552
  defaults, 597–598
  disabling, 571
  enabling, 571–572
  policy, 534
  refreshing the kernel, 568
  troubleshooting, 583–585
audit session ID, 605
  overview, 515–516
audit_syslog plugin, 520–521
  setting attributes, 563–564
audit tokens
  See also individual audit token names
  added by audit policy, 603
  audit record format, 606
  description, 517, 520
  format, 607
  list of, 607
  xclient token, 617
audit trail
  adding disk space, 559–562
  analysis costs, 537
  cleaning up not terminated files, 580–581
  creating
    summary files, 577
  description, 517
  effect of audit policy, 534
  monitoring in real time, 538
  no public objects, 518
  overview, 525

audit trail *(Continued)*
preventing overflow, 581–582
reducing size of, 585–587, 593–594
selecting events from, 576–578
sending files to remote repository, 562–563
viewing events from, 578–580
viewing events from different zones, 601
audit user ID
mechanism, 605
overview, 515–516
`audit_warn` script
configuring, 553
description, 599
`auditconfig` command
adding audit file systems, 559–562
audit classes as arguments, 519
configuring policy, 549–551
configuring queue controls, 551–552
description, 599
displaying audit defaults, 543–544
`-getplugin` option, 562–563, 563–564
policy options, 549–551
preselecting audit classes, 545–546
queue control options, 551–552
sending files to remote repository, 562–563
`-setflags` option, 545–546
`-setnaflags` option, 545–546
`-setplugin` option, 562–563, 563–564
setting active audit policy, 551
setting `audit_binfile` attributes, 559–562
setting audit policy, 588
setting audit policy temporarily, 551
setting `audit_remote` attributes, 562–563
setting system-wide audit parameters, 519
viewing default audit preselection, 545–546
`auditd` daemon
refreshing audit service, 569, 570
auditing
adding audit flags to a group of users, 549
all commands by users, 587–589
changes in current release, 527–528
changes in device policy, 78
configuring
all zones, 542–556

auditing, configuring *(Continued)*
global zone, 550
identically for all zones, 565–567
per zone, 567–568
configuring in global zone, 530
defaults, 597–598
determining if running, 583–585
device allocation, 83–84
disabling, 571
enabling, 571–572
finding changes to specific files, 589–590
getting queue controls, 551–552
logins, 594–595
man page summaries, 598–600
planning, 529–534
planning in zones, 530–531
plugin modules, 520–521
post-selection definition, 517
preselection definition, 517
privileges and, 207–208
removing user-specific audit flags, 548
rights profiles for, 600
roles, 169–170
setting queue controls, 551–552
`sftp` file transfers, 595–596
troubleshooting, 582–583
troubleshooting `praudit` command, 580
updating information, 568–570
users only, 548
zones and, 526–527, 600–601
auditing task map, 541
`auditlog` file, text audit records, 563
`auditrecord` command
`[]` (square brackets) in output, 607
description, 599
displaying audit record definitions, 573–574
example, 573
listing all formats, 573
listing formats of class, 574
listing formats of program, 573–574
optional tokens (`[]`), 607
`auditreduce` command
`-A` option, 575–576
`-b` option, 577

auditreduce command *(Continued)*
- -C option, 576
- -c option, 577–578
- cleaning up audit files, 580–581
- -D option, 576
- -d option, 577–578
- description, 599
- -e option, 577
- examples, 574–576
- filtering options, 576
- -M option, 576
- merging audit records, 574–576
- -O option, 574–576, 576, 577
- selecting audit records, 576–578
- time stamp use, 574
- trailer tokens, and, 616
- using lowercase options, 576
- using uppercase options, 575

auditstat command, description, 600

auth_attr database
- description, 202–203
- summary, 201

AUTH_DES authentication, *See* AUTH_DH authentication

AUTH_DH authentication, and NFS, 267

authentication
- AUTH_DH client-server session, 269–271
- configuring cross-realm, 370–372
- description, 51–52
- DH authentication, 268–271
- disabling with -X option, 492
- Kerberos and, 327
- naming services, 267
- network security, 51–52
- NFS-mounted files, 275
- overview of Kerberos, 505–506
- Secure RPC, 267
- Secure Shell
  - methods, 294–295
  - process, 312–313
- terminology, 501–502
- types, 51–52
- use with NFS, 267

authentication methods
- GSS-API credentials in Secure Shell, 294

authentication methods *(Continued)*
- host-based in Secure Shell, 294, 297–300
- password in Secure Shell, 295
- public keys in Secure Shell, 295
- Secure Shell, 294–295

authenticator
- in Kerberos, 501, 507

authlog file, saving failed login attempts, 60–61

authorizations
- device allocation, 89
- Kerberos and, 327
- troubleshooting, 173–176
- types, 51–52

authorizations (RBAC)
- checking for wildcards, 173
- checking in privileged application, 142
- commands that require authorizations, 205–206
- database, 201–204
- definition, 140–141
- delegating, 201
- description, 137, 200–201
- for allocating device, 80–81
- for device allocation, 90
- granularity, 200
- naming convention, 200
- not requiring for device allocation, 83
- solaris.device.allocate, 80, 90
- solaris.device.revoke, 90

authorized_keys file, description, 320

AuthorizedKeysFile keyword, sshd_config file, 314

auths command, description, 204

AUTHS_GRANTED keyword, policy.conf file, 203

auto_transition option, SASL and, 291

automatic login
- disabling, 492
- enabling, 491

automatically configuring
- Kerberos
  - master KDC server, 351–352
  - slave KDC server, 364–365

automating principal creation, 449

auxprop_login option, SASL and, 291

B

-b option, auditreduce command, 577

backup
  Kerberos database, 403–405
  slave KDCs, 342–343

Banner keyword, sshd_config file, 314

BART
  components, 96–98
  overview, 95–98
  programmatic output, 112
  security considerations, 98–99
  task map, 99
  verbose output, 111

bart command, 95

bart compare command, 97

bart create command, 96, 99

Basic Audit Reporting Tool, *See* BART

basic privilege set, 150

Basic Solaris User (RBAC), rights profile, 198

Batchmode keyword, ssh_config file, 314

BindAddress keyword, ssh_config file, 314

binding control flag, PAM, 284

blowfish-cbc encryption algorithm, ssh_config file, 315

Blowfish encryption algorithm
  allowing in heterogeneous environment, 62

Blowfish encryption algorithm, kernel provider, 236

Blowfish encryption algorithm
  policy.conf file, 62
  ssh_config file, 314

Bourne shell, privileged version, 144

C

-C option, auditreduce command, 576

C shell, privileged version, 144

-c option
  auditrecord command, 574
  auditreduce command, 577–578

cache, credential, 506

canon_user_plugin option, SASL and, 291

caret (^), audit class prefix modifier, 601–602

caret (^) in audit class prefixes, 546–549, 590

caret (^) prefix, using in audit_flags value, 547

CD-ROM drives
  allocating, 86
  security, 94

cdrw command, authorizations required, 205

certificate signing requests (CSR), *See* certificates

certificates
  exporting for use by another system, 255–256
  generating with pktool gencert command, 253–254
  importing into keystore, 254–255
  signing PKCS #10 CSR
    using the pktool command, 261–262

ChallengeResponseAuthentication keyword, *See* KbdInteractiveAuthentication keyword

changing
  allocatable devices, 82–83
  audit_class file, 554
  audit_event file, 555–556
  auditing defaults, 545–546
  default password algorithm, 61–64
  device policy, 77–78
  file ownership, 123–124
  file permissions
    absolute mode, 125–126
    special, 126–127
    symbolic mode, 124–125
  group ownership of file, 124
  NFS secret keys, 269
  passphrase for Secure Shell, 304
  password algorithm for a domain, 62–63
  password algorithm task map, 61–64
  password of role, 177–178
  properties of role, 178–179
  rights profile contents, 170–171
  root password, 56–57
  root role into user, 184–186
  special file permissions, 126–127
  your password with kpasswd, 486
  your password with passwd, 486

CheckHostIP keyword, ssh_config file, 314

chgrp command
  description, 114
  syntax, 124

chkey command, 269, 274

chmod command
 changing special permissions, 126–127, 127
 description, 114
 syntax, 127
choosing, your password, 485–486
chown command, description, 114
ChrootDirectory keyword, ssh_config file, 314
Cipher keyword, ssh_config file, 314
Ciphers keyword, Secure Shell, 315
classes, *See* audit classes
cleaning up, binary audit files, 580–581
clear protection level, 492
ClearAllForwardings keyword, Secure Shell port forwarding, 315
client names, planning for in Kerberos, 341–342
ClientAliveCountMax keyword, ssh_config file, 315
ClientAliveInterval keyword, ssh_config file, 315
clients
 AUTH_DH client-server session, 269–271
 configuring for Secure Shell, 312, 313–314
 configuring Kerberos, 381–398
 definition in Kerberos, 501
clntconfig principal
 creating, 357, 363
clock skew
 Kerberos and, 398–399
 Kerberos planning and, 344
clock synchronizing
 Kerberos master KDC and, 357, 364
 Kerberos planning and, 344
 Kerberos slave KDC and, 369
 Kerberos slave server and, 413
cmd audit token, 609–610
cnt audit policy
 description, 535
 with ahlt policy, 603–604
combining audit files
 auditreduce command, 574–576
 from different zones, 601
command execution, Secure Shell, 313
command-line equivalents of SEAM Tool, 444–445
commands
 See also individual commands
 Cryptographic Framework commands, 217
commands *(Continued)*
 determining user's privileged commands, 189–190
 device allocation commands, 90
 device policy commands, 88
 file protection commands, 113
 for administering privileges, 206
 Kerberos, 499–500
 RBAC administration commands, 204–205
 Secure RPC commands, 269
 Secure Shell commands, 322–323
 that assign privileges, 151
 that check for privileges, 142
 user-level cryptographic commands, 217–218
common keys
 calculating, 270
 DH authentication and, 268–271
components
 BART, 96–98
 device allocation mechanism, 88–89
 RBAC, 137–139
 Secure Shell user session, 313
compressing, audit files on disk, 593–594
Compression keyword, Secure Shell, 315
CompressionLevel keyword, ssh_config file, 315
Computer Emergency Response Team/Coordination Center (CERT/CC), 54
computer security, *See* system security
computing
 DH key, 273
 digest of a file, 229–230
 MAC of a file, 230–232
 secret key, 222–224, 224–228
configuration decisions
 auditing
  file storage, 531–532
  policy, 534–537
  who and what to audit, 532–534
  zones, 530–531
 Kerberos
  client and service principal names, 341–342
  clients, 344–345
  clock synchronization, 344
  database propagation, 344
  encryption types, 346–347

configuration decisions, Kerberos *(Continued)*
- KDC server, 345
- mapping host names onto realms, 341
- number of realms, 340
- ports, 342
- realm hierarchy, 341
- realm names, 340
- realms, 340–341
- slave KDCs, 342–343

password algorithm, 40

configuration files
- auditing, 598–600
- `device_maps` file, 91
- for password algorithms, 40
- `policy.conf` file, 40, 61–62, 204
- Secure Shell, 311
- `syslog.conf` file, 60–61, 207
- with privilege information, 207

configured audit policy, permanent audit policy, 549–551

configuring
- active audit policy, 551
- `ahlt` audit policy, 550
- `audit_class` file, 554
- audit classes, 545–546
- `audit_event` file, 555–556
- audit logs task map, 556
- audit policy, 549–551
- audit policy temporarily, 551
- audit queue controls, 551–552
- audit service policy, 549–551
- audit trail overflow prevention, 581–582
- `audit_warn` script, 553
- auditing, 542–556
- auditing in zones, 526–527, 600–601
- auditing task map, 542–543
- device allocation, 79–80
- device policy, 76
- devices task map, 75
- DH key for NIS user, 273–274
- DH key in NIS, 272–273
- exceptions to Secure Shell system defaults, 300–301
- hardware security, 66–68
- host-based authentication for Secure Shell, 297–300

configuring *(Continued)*
- identical auditing for non-global zones, 565–567
- Kerberos
  - adding administration principals, 356, 362
  - clients, 381–398
  - cross-realm authentication, 370–372
  - master KDC server, 351–352, 352–353, 353–357
  - master KDC server using LDAP, 358–364
  - NFS servers, 376–377
  - overview, 349–423
  - slave KDC server, 364–365, 365–366, 366–369
  - task map, 349–350
- password for hardware access, 66–67
- per-zone auditing, 567–568
- permanent audit policy, 549–551
- `perzone` audit policy, 551
- port forwarding in Secure Shell, 300
- privileged users, 180
- RBAC, 163–176
- RBAC task map, 163
- rights profiles, 170–171
- roles, 165–167, 178–179
- `root` role as user, 184–186
- Secure Shell, 297
  - clients, 313–314
  - servers, 314
- Secure Shell task map, 297
- space for audit trail, 559–562
- temporary audit policy, 549–551
- text summaries of audit records, 563–564

configuring application servers, 372–375

`ConnectionAttempts` keyword, `ssh_config` file, 315

`ConnectTimeout` keyword, `ssh_config` file, 315

console, displaying `su` command attempts, 65–66

`CONSOLE` in Secure Shell, 318

Console User (RBAC), rights profile, 198

`CONSOLE_USER` keyword, `policy.conf` file, 203

consumers, definition in Cryptographic Framework, 215

context-sensitive help, SEAM Tool, 445

control manifests (BART), 95

controlling
- system access, 55–56
- system usage, 45–48

conversation keys
  decrypting in secure RPC, 270
  generating in secure RPC, 269
converting, audit records to readable format, 579
Coordinated Universal Time (UTC)
  time stamp use in auditing, 574, 605
copying, files using Secure Shell, 308
copying audit records to single file, 577–578
cost control, and auditing, 537
crammd5.so.1 plug-in, SASL and, 290
creating
  audit trail, 605
  credential table, 378
  file digests, 229–230
  key pair, 257–261
  new device-clean scripts, 94
  new policy (Kerberos), 453, 465–466
  new principal (Kerberos), 453–455
  privileged users, 180
  rights profile for a group of users, 549
  rights profiles, 170–171
  roles, 165–167
  root user, 184–186
  secret keys
    for encryption, 222–224, 224–228
  Secure Shell keys, 302–304
  stash file, 369, 413
  storage for binary audit files, 556–559
  tickets with kinit, 482
cred database, DH authentication, 268–271
cred table
  DH authentication and, 269
  information stored by server, 271
credential
  cache, 506
  description, 270, 501
  obtaining for a server, 507–508
  obtaining for a TGS, 506–507
  or tickets, 329
credential table, adding single entry to, 378–379
credentials, mapping, 343
crontab files, authorizations required, 205
cross-realm authentication, configuring, 370–372
CRYPT_ALGORITHMS_ALLOW keyword, policy.conf file, 41
CRYPT_ALGORITHMS_DEPRECATE keyword, policy.conf file, 41
crypt_bsdbf password algorithm, 40
crypt_bsdmd5 password algorithm, 40
crypt command, file security, 49
CRYPT_DEFAULT keyword, policy.conf file, 41
CRYPT_DEFAULT system variable, 61
crypt_sha256 password algorithm, 40, 61–64
crypt_sunmd5 password algorithm, 40
crypt_unix password algorithm, 40
Crypto Management (RBAC), creating role, 168–169
cryptoadm command
  description, 217
  disabling cryptographic mechanisms, 241, 243
  disabling hardware mechanisms, 246–248
  installing PKCS #11 library, 241
  listing providers, 236
  -m option, 241, 243
  -p option, 242, 243
  restoring kernel software provider, 243
cryptoadm install command, installing PKCS #11 library, 241
Cryptographic Framework
  administering with role, 168–169
  connecting providers, 218
  consumers, 215
  cryptoadm command, 217
  definition of terms, 215
  description, 213–215
  elfsign command, 218
  error messages, 234
  hardware plugins, 215
  interacting with, 217
  listing providers, 236–239
  PKCS #11 library, 215
  providers, 215
  refreshing, 248
  registering providers, 218
  restarting, 248
  signing providers, 218
  task maps, 221
  user-level commands, 217–218

Cryptographic Framework *(Continued)*
  zones and, 219, 248
cryptographic services, *See* Cryptographic Framework
Cryptoki, *See* PKCS #11 library
csh command, privileged version, 144
customizing, manifests, 101–102
customizing a report (BART), 107–108

D

-D option
  auditreduce command, 576
  ppriv command, 193
-d option
  auditreduce command, 577
daemons
  kcfd, 217
  keyserv, 272
  nscd (name service cache daemon), 204
  running with privileges, 147
  ssh-agent, 305–306
  sshd, 311–313
  table of Kerberos, 500
Data Encryption Standard, *See* DES encryption
data forwarding, Secure Shell, 313
databases
  auth_attr, 202–203
  backing up and propagating KDC, 403–405
  creating KDC, 355
  cred for Secure RPC, 268
  exec_attr, 203
  KDC propagation, 344
  NFS secret keys, 269
  prof_attr, 203
  publickey for Secure RPC, 268
  RBAC, 201–204
  user_attr, 202
dd command, generating secret keys, 222–224
deallocate command
  allocate error state, 90–91, 91
  authorizations required, 90, 205
  device-clean scripts and, 94
  using, 87
deallocating
  devices, 87
  forcibly, 82
  microphone, 87
debugging, privileges, 193
debugging sequence number, 614–615
decrypt command
  description, 218
  syntax, 233
decrypting
  conversation keys for Secure RPC, 270
  files, 233
  NFS secret keys, 269
  secret keys, 269
default/login file, description, 320
default_realm section
  krb5.conf file, 354, 361
defaults
  audit service, 597–598
  privilege settings in policy.conf file, 207
  system-wide in policy.conf file, 40
  umask value, 117
delegating, RBAC authorizations, 201
delete_entry command, ktutil command, 478
deleting
  archived audit files, 581
  audit files, 574
  host's service, 478
  not_terminated audit files, 580–581
  policies (Kerberos), 468
  principal (Kerberos), 457–458
DenyGroups keyword, sshd_config file, 315
DenyUsers keyword, sshd_config file, 315
DES encryption, kernel provider, 236
DES encryption, Secure NFS, 268
destroying, tickets with kdestroy, 484
determining
  audit ID of a user, 591
  auditing is running, 583–585
  files with setuid permissions, 128
  privileges on a process, 191–192
  privileges task map, 187
/dev/arp device, getting IP MIB-II
  information, 78–79

- `/dev/urandom` device, 222–224
- `devfsadm` command, description, 88
- `device_allocate` file
  - description, 92–93
  - format, 92
  - sample, 82, 92
- device allocation
  - adding devices, 79–80
  - allocatable devices, 93
  - allocate error state, 90–91
  - allocating devices, 84–85
  - auditing, 83–84
  - authorizations, 89
  - authorizations for commands, 90
  - authorizing users to allocate, 80–81
  - changing allocatable devices, 82–83
  - commands, 90
  - components of mechanism, 88–89
  - configuration file, 91
  - `deallocate` command
    - device-clean scripts and, 94
    - using, 87
  - deallocating devices, 87
  - `device_allocate` file, 92–93
  - device-clean scripts
    - audio devices, 94
    - CD-ROM drives, 94
    - description, 93–94
    - diskette drives, 94
    - options, 94
    - tape drives, 93
    - writing new scripts, 94
  - `device_maps` file, 91
  - disabling, 80
  - enabling, 80
  - examples, 85
  - forcibly allocating devices, 82
  - forcibly deallocating devices, 82
  - making device allocatable, 80
  - managing devices, 79–80
  - mounting devices, 85–86
  - not requiring authorization, 83
  - preventing, 83
  - requiring authorization, 82–83
- device allocation *(Continued)*
  - rights profiles, 89
  - SMF service, 89
  - task map, 79–80
  - troubleshooting, 85, 86
  - troubleshooting permissions, 81
  - unmounting allocated device, 87
  - user procedures, 79–84
  - using, 79–84
  - using `allocate` command, 84–85
  - viewing information, 81
- device-clean scripts
  - and object reuse, 93–94
  - audio devices, 94
  - CD-ROM drives, 94
  - description, 93–94
  - diskette drives, 94
  - options, 94
  - tape drives, 93
  - writing new scripts, 94
- device management, *See* device policy
- Device Management rights profile, 89
- `device_maps` file
  - description, 91
  - format, 91
  - sample entries, 91
- device policy
  - `add_drv` command, 88
  - auditing changes, 78
  - changing, 77–78
  - commands, 88
  - configuring, 76–79
  - kernel protection, 87–94
  - managing devices, 76
  - overview, 42–44
  - removing from device, 77–78
  - task map, 76
  - `update_drv` command, 77–78, 88
  - viewing, 76–77
- Device Security rights profile, 80, 89
- devices
  - adding device policy, 77–78
  - allocating for use, 79–84
  - auditing allocation of, 83–84

devices *(Continued)*
- auditing policy changes, 78
- authorizing users to allocate, 80–81
- changing device policy, 77–78
- changing which are allocatable, 82–83
- deallocating a device, 87
- `/dev/urandom` device, 222–224
- device allocation
  - *See* device allocation
- forcibly allocating, 82
- forcibly deallocating, 82
- getting IP MIB-II information, 78–79
- listing, 76–77
- listing device names, 81
- login access control, 42
- making allocatable, 80
- managing, 76
- managing allocation of, 79–80
- mounting allocated devices, 85–86
- not requiring authorization for use, 83
- policy commands, 88
- preventing use of all, 83
- preventing use of some, 83
- privilege model and, 152–153
- protecting by device allocation, 43
- protecting in the kernel, 43
- removing policy, 77–78
- security, 42–44
- superuser model and, 152–153
- unmounting allocated device, 87
- viewing allocation information, 81
- viewing device policy, 76–77
- zones and, 43

DH authentication
- configuring in NIS, 272–273
- description, 268–271
- for NIS client, 272–273
- mounting files with, 275
- sharing files with, 274–275

Diffie-Hellman authentication, *See* DH authentication

`digest` command
- description, 217
- example, 229
- syntax, 229

`digestmd5.so.1` plug-in, SASL and, 290

digests
- computing for file, 229–230
- of files, 229–230

direct realms, 371–372

directories
- *See also* files
- displaying files and related information, 113, 122–123
- permissions
  - defaults, 117
  - description, 114–115
- public directories, 117

`DisableBanner` keyword, `ssh_config` file, 315

disabling
- 32–bit executables that compromise security, 121
- abort sequence, 67–68
- audit policy, 549–551
- audit service, 571
- cryptographic mechanisms, 241
- device allocation, 80
- executable stacks, 129–130
- hardware mechanisms, 246–248
- keyboard abort, 67–68
- keyboard shutdown, 67–68
- logging of executable stack messages, 129
- logins temporarily, 58–59
- programs from using executable stacks, 129–130
- remote `root` access, 65–66
- service on a host (Kerberos), 477–479
- system abort sequence, 67–68
- user logins, 58–59

disk space, for binary audit files, 556–559

disk space requirements, audit files, 537–538

diskette drives, device-clean scripts, 94

displaying
- allocatable devices, 81
- audit policies, 549
- audit policy defaults, 543–544
- audit queue controls, 543–544, 552
- audit record definitions, 573–574
- audit records, 578–580
- audit records in XML format, 579
- auditing defaults, 543–544

displaying *(Continued)*
 definition of audit records, 573–574
 device policy, 76–77
 exceptions to system-wide auditing, 543–544
 file information, 122–123
 files and related information, 113
 providers in the Cryptographic Framework, 236–239
 roles you can assume, 159, 204
 root access attempts, 65–66
 selected audit records, 574–576
 su command attempts, 65–66
 sublist of principals (Kerberos), 450
 user's login status, 57–58
 users with no passwords, 58
dminfo command, 91
DNS, Kerberos and, 341–342
domain_realm section
 krb5.conf file, 341, 354, 361
dot (.)
 authorization name separator, 200
 displaying hidden files, 122
double dollar sign ($$), parent shell process number, 191
DSAAuthentication keyword, *See* PubkeyAuthentication keyword
duplicating, principals (Kerberos), 456
DynamicForward keyword, ssh_config file, 315

E

-e option
 auditreduce command, 577
 ppriv command, 193
ECC kernel provider, 236
eeprom command, 38, 66–68
effective privilege set, 149
efficiency, auditing and, 538
eject command, device cleanup and, 94
elfsign command, description, 218
enabling
 audit service, 571–572
 auditing, 571–572
 cryptographic mechanisms, 242

enabling *(Continued)*
 device allocation, 80
 Kerberized applications only, 421–422
 kernel software provider use, 243
 keyboard abort, 67–68
 mechanisms and features on hardware provider, 247
encrypt command
 description, 218
 error messages, 234
 syntax, 223
 troubleshooting, 234
encrypting
 communications between hosts, 305
 encrypt command, 232–235
 files, 49, 222, 232–235
 network traffic between hosts, 293–295
 passwords, 61–64
 private key of NIS user, 274
 Secure NFS, 268
 using user-level commands, 217–218
encryption
 algorithms
  Kerberos and, 346–347
 DES algorithm, 268
 generating symmetric key
  using the dd command, 222–224
  using the pktool command, 224–228
 list of password algorithms, 40
 modes
  Kerberos and, 346–347
 password algorithm, 40
 privacy service, 327
 specifying algorithms in ssh_config file, 314
 specifying password algorithm locally, 61–64
 specifying password algorithms in policy.conf file, 40
 types
  Kerberos and, 346–347, 509–511
 with -x option, 492
environment variables
 See also variables
 audit token for, 610

environment variables *(Continued)*
- overriding proxy servers and ports, 309
- PATH, 45
- presence in audit records, 535, 607
- Secure Shell and, 318–319
- use with ssh-agent command, 322

equal sign (=), file permissions symbol, 118

error messages
- encrypt command, 234
- Kerberos, 425–439
- with kpasswd, 486

errors, allocate error state, 90–91

EscapeChar keyword, ssh_config file, 315

/etc/default/kbd file, 67–68

/etc/default/login file
- description, 320
- login default settings, 60
- restricting remote root access, 65–66
- Secure Shell and, 318–319

/etc/default/su file
- displaying su command attempts, 65–66
- monitoring access attempts, 65–66
- monitoring su command, 64–65

/etc/hosts.equiv file, description, 321

/etc/krb5/kadm5.acl file, description, 497

/etc/krb5/kadm5.keytab file, description, 498

/etc/krb5/kdc.conf file, description, 498

/etc/krb5/kpropd.acl file, description, 498

/etc/krb5/krb5.conf file, description, 498

/etc/krb5/krb5.keytab file, description, 498

/etc/krb5/warn.conf file, description, 498

/etc/logindevperm file, 42

/etc/nologin file
- description, 321
- disabling user logins temporarily, 58–59

/etc/pam.conf file, Kerberos and, 498

/etc/publickey file, DH authentication and, 268

/etc/security/audit_event file, audit events and, 518

/etc/security/device_allocate file, 92

/etc/security/device_maps file, 91

/etc/security/policy.conf file, algorithms configuration, 61–62

/etc/ssh_host_dsa_key.pub file, description, 320

/etc/ssh_host_key.pub file, description, 320

/etc/ssh_host_rsa_key.pub file, description, 320

/etc/ssh/shosts.equiv file, description, 321

/etc/ssh/ssh_config file
- configuring Secure Shell, 313–314
- description, 321
- host-specific parameters, 318
- keywords, 314–319
- override, 321

/etc/ssh/ssh_host_dsa_key file, description, 320

/etc/ssh/ssh_host_key file, override, 321

/etc/ssh/ssh_host_rsa_key file, description, 320

/etc/ssh/ssh_known_hosts file
- controlling distribution, 319
- description, 320
- override, 322
- secure distribution, 320

/etc/ssh/sshd_config file
- description, 320
- keywords, 314–319

/etc/ssh/sshrc file, description, 321

/etc/syslog.conf file
- auditing and, 563, 600
- executable stack messages and, 121
- failed logins and, 60–61
- PAM and, 282

event, description, 518

event modifiers, audit records, 611

exec_args audit token
- argv policy and, 610
- format, 610

exec_attr database
- description, 203
- summary, 201

exec_env audit token, format, 610

executable stacks
- disabling logging messages, 129
- logging messages, 121
- protecting against, 129–130
- protecting against 32–bit processes, 121

execute permissions, symbolic mode, 118

export subcommand, pktool command, 255–256

EXTERNAL security mechanism plug-in, SASL and, 290

F

-f option
  Kerberized commands, 491, 493
  st_clean script, 94
-F option
  deallocate command, 90–91
  Kerberized commands, 491, 493
failed login attempts
  loginlog file, 59–60
  syslog.conf file, 60–61
failure, audit class prefix, 601–602
FallBackToRsh keyword, ssh_config file, 315
fd_clean script, description, 94
fe audit event modifier, 611
file audit token, format, 610
file permission modes
  absolute mode, 118
  symbolic mode, 118
FILE privileges, 146
file systems
  adding a virus scan engine, 72
  enabling virus scanning, 72
  excluding files from virus scans, 73–74
  NFS, 267
  scanning for viruses, 71
  security
    authentication and NFS, 267
    TMPFS file system, 117
  sharing files, 49
  TMPFS, 117
file transfers, auditing, 595–596
file vnode audit token, 609
files
  audit_class, 599
  audit_event, 599
  auditing modifications to, 589–590
  BART manifests, 108–109
  changing group ownership, 124
  changing ownership, 114, 123–124
  changing special file permissions, 126–127
  computing a digest, 229–230
  computing digests of, 229–230
  computing MAC of, 230–232
  copying with Secure Shell, 308

files *(Continued)*
  decrypting, 233
  digest of, 229–230
  displaying file information, 122–123
  displaying hidden files, 122
  displaying information about, 113
  encrypting, 222, 232–235
  file types, 114
  finding files with setuid permissions, 128
  for administering Secure Shell, 320
  hashing, 222
  kdc.conf, 503
  Kerberos, 497–498
  manifests (BART), 108–109
  mounting with DH authentication, 275
  ownership
    and setgid permission, 116
    and setuid permission, 116
  permissions
    absolute mode, 118, 125–126
    changing, 114, 117–119, 125
    defaults, 117
    description, 114–115
    setgid, 116
    setuid, 116
    sticky bit, 116–117
    symbolic mode, 118, 124–125, 125
    umask value, 117
  PKCS #12, 256
  privileges relating to, 146
  protecting with UNIX permissions, 122
  public objects, 518
  security
    access restriction, 46
    ACL, 49
    changing ownership, 123–124
    changing permissions, 117–119, 125
    directory permissions, 114–115
    displaying file information, 113, 122–123
    encryption, 49, 222
    file permissions, 114–115
    file types, 114
    special file permissions, 119
    umask default, 117

files, security *(Continued)*
UNIX permissions, 113–119
user classes, 114
sharing with DH authentication, 274–275
special files, 115–117
symbols of file type, 114
syslog.conf, 600
verifying integrity with digest, 229–230
with privilege information, 207
find command, finding files with setuid permissions, 128
FIPS-140 support, Secure Shell using a Sun Crypto Accelerator 6000 card, 296
firewall systems
connecting from outside, 310
outside connections with Secure Shell
from command line, 310
from configuration file, 309–310
packet smashing, 53–54
packet transfers, 53–54
secure host connections, 309
security, 52–53
trusted hosts, 53
flags line, process preselection mask, 605
fmri audit token, format, 611
forced cleanup, st_clean script, 94
format of audit records, auditrecord command, 573
forwardable tickets
definition, 502
description, 328
example, 482
with -F option, 491, 493
with -f option, 491, 493
ForwardAgent keyword, Secure Shell forwarded authentication, 315
ForwardX11 keyword, Secure Shell port forwarding, 315
ForwardX11Trusted keyword, Secure Shell port forwarding, 315
fp audit event modifier, 611
FQDN (Fully Qualified Domain Name), in Kerberos, 341–342
ftp command
Kerberos and, 490–493, 499
ftp command *(Continued)*
logging file transfers, 595–596
setting protection level in, 492
ftpd daemon, Kerberos and, 500

G

GatewayPorts keyword, Secure Shell, 315
gateways, *See* firewall systems
gencert subcommand, pktool command, 253–254
generating
certificates with pktool command, 253–254
key pair
using the pktool command, 257–261
keys for Secure Shell, 302–304
NFS secret keys, 269
passphrases with pktool command, 256–257
random number
using the dd command, 222–224
using the pktool command, 224–228
Secure Shell keys, 302–304
symmetric key
using the dd command, 222–224
using the pktool command, 224–228
X.509 v3 certificate, 261–262
getdevpolicy command, description, 88
getent command, description, 204
-getflags option
auditconfig command, 543–544, 545–546
-getnaflags option
auditconfig command, 543–544, 545–546
-getplugin option
auditconfig command, 543–544, 562–563, 563–564
-getpolicy option
auditconfig command, 543–544, 549–551
-getqctrl option, auditconfig command, 543–544
getting
access to a specific service, 508
credential for a server, 507–508
credential for a TGS, 506–507
gkadmin command
See also SEAM Tool
description, 499

.gkadmin file
  description, 497
  SEAM Tool and, 445
GlobalKnownHostsFile keyword
  See GlobalKnownHostsFile keyword
  ssh_config file, 315
granting access to your account, 488–490
group audit policy
  and groups token, 535
  description, 535
  group token, and, 611
group audit token
  format, 611
  group policy, and, 611
groups
  changing file ownership, 124
  exceptions to Secure Shell defaults, 300–301
GSS-API
  authentication in Secure Shell, 294
  credentials in Secure Shell, 312
  Kerberos and, 328
gssapi.so.1 plug-in, SASL and, 290
GSSAPIAuthentication keyword, Secure Shell, 315
GSSAPIDelegateCredentials keyword, ssh_config file, 315
GSSAPIKeyExchange keyword, Secure Shell, 315
GSSAPIStoreDelegatedCredentials keyword, sshd_config file, 315
gsscred command, description, 499
gsscred table, using, 511
gssd daemon, Kerberos and, 500

H

-h option, auditrecord command, 573
hard disk, space requirements for auditing, 537–538
hardware
  listing attached hardware accelerators, 245–246
  protecting, 38, 66–68
  requiring password for access, 66–67
hardware providers
  disabling cryptographic mechanisms, 246–248
  enabling mechanisms and features on, 247
  listing, 245–246
hardware providers *(Continued)*
  loading, 245
hash
  algorithms
    Kerberos and, 346–347
hashing, files, 222
HashKnownHosts keyword, ssh_config file, 315
header audit token
  event modifiers, 611
  format, 611–612
  order in audit record, 611–612
help
  SEAM Tool, 445–446
  URL for online, 347
Help Contents, SEAM Tool, 446
hierarchical realms
  configuring, 370–371
  in Kerberos, 333, 341
hmac-md5 algorithm, ssh_config file, 316
hmac-sha1 encryption algorithm, ssh_config file, 316
host-based authentication
  configuring in Secure Shell, 297–300
  description, 294
Host keyword
  ssh_config file, 316, 318
host names, mapping onto realms, 341
host principal
  creating, 356, 363
HostbasedAuthentication keyword, Secure Shell, 316
HostbasedUsesNameFromPacketOnly keyword, sshd_config file, 316
HostKey keyword, sshd_config file, 316
HostKeyAlgorithms keyword, ssh_config file, 316
HostKeyAlias keyword, ssh_config file, 316
HostName keyword, ssh_config file, 316
hosts
  disabling Kerberos service on, 477–479
  exceptions to Secure Shell defaults, 300–301
  Secure Shell hosts, 294
  trusted hosts, 53
hosts.equiv file, description, 321

I

-I option
  bart create command, 99
  st_clean script, 94
-i option
  bart create command, 99, 102
  encrypt command, 232
  st_clean script, 94
identity files (Secure Shell), naming conventions, 320
IDs
  audit
    mechanism, 605
    overview, 515–516
  audit session, 605
  mapping UNIX to Kerberos principals, 511
IgnoreIfUnknown keyword, ssh_config file, 316
IgnoreRhosts keyword, sshd_config file, 316
IgnoreUserKnownHosts keyword, sshd_config file, 316
import subcommand, pktool command, 254–255
in.ftpd daemon, Kerberos and, 500
in.rlogind daemon, Kerberos and, 500
in.rshd daemon, Kerberos and, 500
in.telnetd daemon, Kerberos and, 500
include control flag, PAM, 284
inheritable privilege set, 149
initial ticket, definition, 502
install subcommand, cryptoadm command, 241
installing, Secure by Default, 47
instance, in principal names, 332–333
integrity
  Kerberos and, 327
  security service, 334
interactively configuring
  Kerberos
    master KDC server, 352–353
    slave KDC server, 365–366
INTERNAL plug-in, SASL and, 290
Internet firewall setup, 52–53
Internet-related tokens
  ip address token, 612
  ip port token, 612
  socket token, 615
invalid ticket, definition, 502
ioctl() system calls, AUDIO_SETINFO(), 94
ip address audit token, format, 612
IP addresses
  exceptions to Secure Shell defaults, 300–301
  Secure Shell checking, 314
IP MIB-II, getting information from /dev/arp, 78–79
ip port audit token, format, 612
ipc audit token, 612–613
  format, 612–613
IPC_perm audit token, format, 613
IPC privileges, 147
ipc type field values (ipc token), 612–613

K

-k option
  encrypt command, 232
  Kerberized commands, 492
  mac command, 230
-K option
  encrypt command, 232
  mac command, 230
  Kerberized commands, 492
  rolemod command, 179
  usermod command, 180–181
.k5.*REALM* file, description, 498
.k5login file
  description, 488–490, 497
  rather than revealing password, 489
kadm5.acl file
  description, 497
  format of entries, 460
  master KDC entry, 355, 362, 402
  new principals and, 454, 456
kadm5.keytab file, description, 498
kadmin command
  creating host principal, 356, 363
  description, 499
  ktadd command, 474–475
  ktremove command, 476
  removing principals from keytab with, 476
  SEAM Tool and, 443
kadmin.local command
  adding administration principals, 356, 362

kadmin.local command *(Continued)*
automating creation of principals, 449
description, 499
kadmin.log file, description, 498
kadmind daemon
Kerberos and, 500
master KDC and, 500
kbd file, 67–68
KbdInteractiveAuthentication keyword, Secure Shell, 316
kcfd daemon, 217, 248
kclient command, description, 499
kdb5_ldap_util command, description, 499
kdb5_util command
creating KDC database, 355
creating stash file, 369, 413
description, 499
KDC
backing up and propagating, 403–405
configuring master
automatic, 351–352
interactive, 352–353
manual, 353–357
with LDAP, 358–364
configuring slave
automatic, 364–365
interactive, 365–366
manual, 366–369
copying administration files from slave to master, 367, 411
creating database, 355
creating host principal, 356, 363
database propagation, 344
master
definition, 500
planning, 342–343
ports, 342
restricting access to servers, 422–423
slave, 342–343
definition, 500
slave or master, 333–334, 350
starting daemon, 369, 413
swapping master and slave, 399–403

KDC *(Continued)*
synchronizing clocks
master KDC, 357, 364
slave KDC, 369, 413
kdc.conf file
description, 498
ticket lifetime and, 503
kdc.log file, description, 498
kdcmgr command
configuring master
automatic, 352
interactive, 352
configuring slave
automatic, 364
interactive, 365
server status, 353
kdestroy command
example, 484
Kerberos and, 499
KeepAlive keyword, Secure Shell, 316
Kerberos
administering, 443–479
Administration Tool
See SEAM Tool
commands, 490–495, 499–500
components of, 335–336
configuration decisions, 339–347
configuring KDC servers, 350–370
daemons, 500
enabling Kerberized applications only, 421–422
encryption types
overview, 346–347
using, 509–511
error messages, 425–439
examples of using Kerberized commands, 494–495
files, 497–498
gaining access to server, 506–508
granting access to your account, 488–490
Kerberos V5 protocol, 327
online help, 347
options to Kerberized commands, 491
overview
authentication system, 328–334, 505–506
Kerberized commands, 490–493

Kerberos *(Continued)*
- password management, 485–490
- planning for, 339–347
- realms
  - *See* realms (Kerberos)
- reference, 497–511
- remote applications, 332
- table of network command options, 492
- terminology, 500–505
- troubleshooting, 439
- using, 481–495

Kerberos authentication, and Secure RPC, 268

Kerberos commands, 490–495
- enabling only Kerberized, 421–422
- examples, 494–495

`kern.notice` entry, `syslog.conf` file, 121

kernel providers, listing, 236

Key Distribution Center, *See* KDC

key management framework (KMF), *See* KMF

key pairs
- creating, 257–261
- generating
  - using the `pktool` command, 257–261

`KEYBOARD_ABORT` system variable, 67–68

`keylogin` command, use for Secure RPC, 269

`KeyRegenerationInterval` keyword, `sshd_config` file, 316

keys
- creating DH key for NIS user, 273–274
- creating for Secure Shell, 302–304
- definition in Kerberos, 501
- generating for Secure Shell, 302–304
- generating key pair
  - using the `pktool` command, 257–261
- generating symmetric key
  - using the `dd` command, 222–224
  - using the `pktool` command, 224–228
- service key, 473–479
- session keys
  - Kerberos authentication and, 505–506
- using for MAC, 231

`keyserv` daemon, 272

keyserver
- description, 269

keyserver *(Continued)*
- starting, 272

keystores
- exporting certificates, 255–256
- importing certificates, 254–255
- listing contents, 253
- managed by KMF, 250
- protecting with password in KMF, 256–257
- supported by KMF, 250, 251

keytab file
- adding master KDC's host principal to, 357, 363
- adding service principal to, 473, 474–475
- administering, 473–479
- administering with `ktutil` command, 474
- disabling a host's service with `delete_entry` command, 478
- read into keytab buffer with `read_kt` command, 477, 478
- removing principals with `ktremove` command, 476
- removing service principal from, 476
- viewing contents with `ktutil` command, 476
- viewing keylist buffer with `list` command, 477, 478

`keytab` option, SASL and, 291

keywords
- *See also* specific keyword
- attribute in BART, 110
- command-line overrides in Secure Shell, 323
- Secure Shell, 314–319

`kgcmgr` command, description, 499

`kinit` command
- example, 482
- `-F` option, 482
- Kerberos and, 499
- ticket lifetime, 503

`klist` command
- example, 483–484
- `-f` option, 483–484
- Kerberos and, 499

KMF
- adding plugin, 262–263
- creating
  - passphrases for keystores, 251
  - password for keystore, 256–257
  - self-signed certificate, 253–254

KMF *(Continued)*
- exporting certificates, 255–256
- importing certificates into keystore, 254–255
- keystores, 250, 251
- library, 250
- listing plugins, 262–263
- managing
  - keystores, 251
  - PKI policy, 250
  - plugins, 251
  - public key technologies (PKI), 249
- removing plugin, 262–263
- utilities, 250

kmfcfg command
- list plugin subcommand, 262–263
- plugin subcommands, 249, 251

known_hosts file
- controlling distribution, 319
- description, 320

Korn shell, privileged version, 144

kpasswd command
- error message, 486
- example, 487
- Kerberos and, 499
- passwd command and, 486

kprop command, description, 499

kpropd.acl file, description, 498

kpropd daemon, Kerberos and, 500

kproplog command, description, 499

krb5.conf file
- description, 498
- domain_realm section, 341
- editing, 354, 360
- ports definition, 342

krb5.keytab file, description, 498

krb5cc_*uid* file, description, 498

krb5kdc daemon
- Kerberos and, 500
- master KDC and, 500
- starting, 369, 413

ksh command, privileged version, 144

ktadd command
- adding service principal, 473, 474–475
- syntax, 475

ktkt_warnd daemon, Kerberos and, 500

ktremove command, 476

ktutil command
- administering keytab file, 474
- delete_entry command, 478
- Kerberos and, 499
- list command, 477, 478
- read_kt command, 477, 478
- viewing list of principals, 476

kvno command, Kerberos and, 499

L

-L option, ssh command, 307–308

-l option
- digest command, 229
- encrypt command, 223
- mac command, 230

LDAP, configuring master KDC using, 358–364

LDAP naming service
- passwords, 39
- specifying password algorithm, 63–64

least privilege, principle of, 146

libraries, user-level providers, 236

lifetime of ticket, in Kerberos, 503–504

limit privilege set, 149

limiting
- audit file size, 593
- use of privileges in a rights profile, 171

list command, 477, 478

list_devices command
- authorizations required, 90, 205

list plugin subcommand, kmcfg command, 262–263

list privilege, SEAM Tool and, 472

list subcommand, pktool command, 253

ListenAddress keyword, sshd_config file, 316

listing
- all RBAC security attributes, 156–157
- available providers in Cryptographic Framework, 236–239
- contents of keystore, 253
- Cryptographic Framework providers, 245–246
- device policy, 76–77
- hardware providers, 245–246

listing *(Continued)*
providers in the Cryptographic Framework, 236–239
roles you can assume, 159, 204
users with no passwords, 58
your RBAC rights, 157–159
LocalForward keyword, ssh_config file, 316
log files
audit records, 521, 579
BART
programmatic output, 111–112
verbose output, 111–112
configuring for audit service, 563–564
failed login attempts, 60–61
monitoring su command, 64–65
syslog audit records, 600
/var/adm/messages, 585
/var/log/syslog, 585
log_level option, SASL and, 291
logadm command, archiving text summary audit files, 581
logging, ftp file transfers, 595–596
logging in
and AUTH_DH, 269
auditing logins, 594–595
disabling temporarily, 58–59
displaying user's login status, 57–58
log of failed logins, 60–61
monitoring failures, 59–60
root login
restricting to console, 65–66
tracking, 45
security
access control on devices, 42
access restrictions, 38
saving failed attempts, 59–60
system access control, 38
tracking root login, 45
task map, 56
users' basic privilege set, 150
with Secure Shell, 304–305
login access restrictions, svc:/system/name-service/switch:default, 38
login environment variables, Secure Shell and, 318–319
login file
login default settings, 60
restricting remote root access, 65–66
LoginGraceTime keyword, sshd_config file, 316
loginlog file, saving failed login attempts, 59–60
logins command
displaying user's login status, 57–58
displaying users with no passwords, 58
syntax, 57
LogLevel keyword, Secure Shell, 316
LookupClientHostnames keyword, sshd_config file, 316
-lspolicy option, auditconfig command, 549–551

M

-M option, auditreduce command, 576
-m option
cryptoadm command, 241, 243
Kerberized commands, 492
mac command
description, 218
syntax, 230
machine security, *See* system security
MACS keyword, Secure Shell, 316
mail, using with Secure Shell, 307
man pages, audit service, 598–600
managing
See also administering
audit files, 574–576, 581–582
audit records task map, 572–573
audit trail overflow, 581–582
auditing, 541
auditing in zones, 526–527, 600–601
device allocation task map, 79–80
devices, 79–80
file permissions, 121–130
keystores with KMF, 251
passwords with Kerberos, 485–490
privileges task map, 191
RBAC task map, 176–177

manifests
  See also bart create
  control, 95
  customizing, 101–102
  file format, 108–109
  test, 97
manually configuring
  Kerberos
    master KDC server, 353–357
    master KDC server using LDAP, 358–364
    slave KDC server, 366–369
mapping
  host names onto realms (Kerberos), 341
  UIDs to Kerberos principals, 511
mapping GSS credentials, 343
mappings, events to classes (auditing), 520
mask (auditing), description of process preselection, 604
master KDC
  automatically configuring, 351–352
  configuring with LDAP, 358–364
  definition, 500
  interactively configuring, 352–353
  manually configuring, 353–357
  slave KDCs and, 333–334, 350
  swapping with slave KDC, 399–403
Match blocks, exceptions to Secure Shell defaults, 300–301
Match keyword, sshd_config file, 316
max_life value, description, 503
max_renewable_life value, description, 504
MaxStartups keyword, sshd_config file, 316
MD4 encryption algorithm, kernel provider, 236
MD5 encryption algorithm
  allowing in heterogeneous environment, 62
MD5 encryption algorithm, kernel provider, 236
MD5 encryption algorithm
  policy.conf file, 61–62, 62
mech_dh mechanism, GSS-API credentials, 312
mech_krb mechanism, GSS-API credentials, 312
mech_list option, SASL and, 291
mechanism, definition in Cryptographic Framework, 216
mechanisms
  disabling all on hardware provider, 246–248
  enabling some on hardware provider, 247
Media Backup rights profile
  assigning to trusted users, 135, 165
Media Restore rights profile, assigning to trusted users, 165
merging, binary audit records, 574–576
message authentication code (MAC), computing for file, 230–232
messages file, executable stack messages, 121
metaslot
  administering, 217
  definition in Cryptographic Framework, 216
microphone
  allocating, 84
  deallocating, 87
minus sign (-)
  audit class prefix, 601–602
  entry in sulog file, 64
  file permissions symbol, 118
  symbol of file type, 114
mode, definition in Cryptographic Framework, 216
modifying
  policies (Kerberos), 467–468
  principal's password (Kerberos), 457
  principals (Kerberos), 456–457
  roles (RBAC), 178–179
  user security attributes, 546–549
  users (RBAC), 179–181
modules, password encryption, 40
monitoring
  audit trail in real time, 538
  failed logins, 59–60
  su command attempts, 45, 64–65
  superuser, 64–66
  superuser access attempts, 65–66
  system usage, 48
  use of privileged commands, 169–170
mount command, with security attributes, 80
mounting
  allocated CD-ROM, 86
  allocated devices, 85–86
  files with DH authentication, 275

mt command, tape device cleanup and, 93

N

-n option, bart create command, 99
n2cp driver
  hardware plugin to Cryptographic Framework, 215
  listing mechanisms, 245–246
na audit event modifier, 611
names
  device names
    device_maps file, 91, 92
naming conventions
  audit files, 605
  devices, 81
  RBAC authorizations, 200
  Secure Shell identity files, 320
naming service configuration, login access restrictions, 38
naming services
  See individual naming services
  scope and RBAC, 144
ncp driver
  hardware plugin to Cryptographic Framework, 215
  listing mechanisms, 245–246
NET privileges, 147
netservices limited installation option, 47
network, privileges relating to, 147
network security
  authentication, 51–52
  authorizations, 51–52
  controlling access, 50–54
  firewall systems
    need for, 53
    packet smashing, 53–54
    trusted hosts, 53
  overview, 50
  reporting problems, 54
Network Time Protocol, *See* NTP
never-audit classes, process preselection mask, 604
new features
  auditing enhancements, 527–528
  SASL, 289
  Secure Shell and FIPS-140, 296
new features *(Continued)*
  Secure Shell enhancements, 295–296
newkey command
  creating key for NIS user, 273–274
  generating keys, 269
NFS file systems
  authentication, 267
  providing client-server security, 269–271
  secure access with AUTH_DH, 275
NFS servers, configuring for Kerberos, 376–377
NIS naming service
  authentication, 267
  passwords, 39
  specifying password algorithm, 62–63
nisaddcred command, generating keys, 269
nobody user, 50
noexec_user_stack_log variable, 121, 129
noexec_user_stack variable, 121, 129–130
NoHostAuthenticationForLocalHost keyword, ssh_config file, 317
nologin file, description, 321
nonhierarchical realms, in Kerberos, 333
nscd (name service cache daemon), use, 204
NSS, managing keystore, 251
NTP
  Kerberos planning and, 344
  master KDC and, 357, 364
  slave KDC and, 369, 413
NumberOfPasswordPrompts keyword, ssh_config file, 317

O

-O option
  auditreduce command, 574–576, 576, 577
-o option, encrypt command, 232
object reuse requirements
  device-clean scripts
    tape drives, 93
    writing new scripts, 94
  for devices, 93–94
obtaining
  access to a specific service, 508
  credential for a server, 507–508

obtaining *(Continued)*
credential for a TGS, 506–507
forwardable tickets, 482
privileged commands, 178–179
privileges, 150, 151, 179, 180–181
privileges on a process, 191–192
tickets with `kinit`, 482
online help
SEAM Tool, 445–446
URL for, 347
OpenSSH, *See* Secure Shell
OpenSSL
managing keystore, 251
version, 250
Operator (RBAC)
recommended role, 135
rights profile, 198
optional control flag, PAM, 284
options to Kerberized commands, 491
order of search
security attributes, 199
user security attributes, 199
overflow prevention, audit trail, 581–582
`ovsec_adm.`*xxxxx* file, description, 498
ownership of files
ACLs and, 49
changing, 114, 123–124
changing group ownership, 124
UFS ACLs and, 119–120

P
`-p` option
`auditrecord` command, 573–574
`bart create`, 102
`cryptoadm` command, 242, 243
`logins` command, 58
packet transfers
firewall security, 53
packet smashing, 53–54
PAM
adding a module, 281
configuration file
control flags, 284
PAM, configuration file *(Continued)*
introduction, 282
Kerberos and, 498
stacking diagrams, 285
stacking example, 287
stacking explained, 283
syntax, 283
`/etc/syslog.conf` file, 282
framework, 278
Kerberos and, 336
overview, 277
planning, 280
stack to cache authentication, 162
task map, 279
`pam.conf` file, *See* PAM configuration file
PAM configuration file, adding `su` stack, 162
PAM modules, 162
`pam_roles` command, description, 204
`pam_tty_tickets.so.1` module, PAM, 162
`PAMServiceName` keyword, `sshd_config` file, 317
`PAMServicePrefix` keyword, `sshd_config` file, 317
panels, table of SEAM Tool, 469–472
passphrases
changing for Secure Shell, 304
`encrypt` command, 232
example, 305
generating in KMF, 256–257
`mac` command, 230
storing safely, 233
using for MAC, 231
using in Secure Shell, 305–306
`PASSREQ` in Secure Shell, 319
`passwd` command
and `kpasswd` command, 486
and naming services, 39
changing password of role, 177–178
syntax, 56
password authentication, Secure Shell, 294
`PasswordAuthentication` keyword, Secure Shell, 317
passwords
authentication in Secure Shell, 294
changing role password, 177–178
changing with `kpasswd` command, 486
changing with `passwd -r` command, 39

passwords *(Continued)*
- changing with passwd command, 486
- constraining encryption algorithms in heterogeneous environment, 62
- displaying users with no passwords, 58
- eliminating in Secure Shell, 305–306
- encryption algorithms, 40
- finding users with no passwords, 58
- granting access without revealing, 488–490
- hardware access and, 66–67
- LDAP, 39
  - specifying new password algorithm, 63–64
- local, 39
- login security, 38, 39
- managing, 485–490
- modifying a principal's password, 457
- NIS, 39
  - specifying new password algorithm, 62–63
- policies and, 486
- PROM security mode, 38, 66–68
- protecting
  - keystore, 256
  - PKCS #12 file, 256
- requiring for hardware access, 66–67
- secret-key decryption for Secure RPC, 269
- specifying algorithm, 61–62
  - in naming services, 62–63
  - locally, 61–64
- suggestions on choosing, 485–486
- system logins, 39
- task map, 56
- UNIX and Kerberos, 485–490
- using Blowfish in heterogeneous environment, 62
- using MD5 encryption algorithm for, 61–62
- using new algorithm, 62
- using user's to assume role, 183–184

path_attr audit token, 613

path audit policy, description, 536

path audit token, format, 613

PATH environment variable
- and security, 46
- setting, 46

PATH in Secure Shell, 319

permanent audit policy, configured audit policy, 549–551

permissions
- ACLs and, 49
- changing file permissions
  - absolute mode, 118, 125–126
  - chmod command, 114
  - symbolic mode, 118, 124–125, 125
- defaults, 117
- directory permissions, 114–115
- file permissions
  - absolute mode, 118, 125–126
  - changing, 117–119, 125
  - description, 114–115
  - special permissions, 117, 119
  - symbolic mode, 118, 124–125, 125
- finding files with setuid permissions, 128
- setgid permissions
  - absolute mode, 119, 127
  - description, 116
  - symbolic mode, 118
- setuid permissions
  - absolute mode, 119, 127
  - description, 116
  - security risks, 116
  - symbolic mode, 118
- special file permissions, 115–117, 117, 119
- sticky bit, 116–117
- UFS ACLs and, 119–120
- umask value, 117
- user classes and, 114

PermitEmptyPasswords keyword, sshd_config file, 317

PermitRootLogin keyword, sshd_config file, 317

permitted privilege set, 149

PermitUserEnvironment keyword, sshd_config file, 317

perzone audit policy
- description, 536
- setting, 551
- using, 531, 567–568, 600–601
- when to use, 526–527

pfcsh command, description, 144

pfexec command, description, 204

pfksh command, description, 144
pfsh command, description, 144
physical security, description, 38
PidFile keyword, Secure Shell, 317
PKCS #10 CSR
  signing
    using the pktool command, 261–262
PKCS #11 library
  adding provider library, 241
  in Cryptographic Framework, 215
PKCS #11 softtokens, managing keystore, 251
PKCS #12 files, protecting, 256
pkcs11_kernel.so user-level provider, 236
pkcs11_softtoken.so user-level provider, 236
PKI
  managed by KMF, 249
  policy managed by KMF, 250
pktool command
  creating self-signed certificate, 253–254
  export subcommand, 255–256
  gencert subcommand, 253–254
  generating key pairs, 257–261
  generating secret keys, 224–228
  import subcommand, 254–255
  list subcommand, 253
  managing PKI objects, 249
  setpin subcommand, 256–257
  signing PKCS #10 CSR, 261–262
plain.so.1 plug-in, SASL and, 290
planning
  auditing, 529–534
  auditing in zones, 530–531
  auditing task map, 529–534
  Kerberos
    client and service principal names, 341–342
    clock synchronization, 344
    configuration decisions, 339–347
    database propagation, 344
    number of realms, 340
    ports, 342
    realm hierarchy, 341
    realm names, 340
    realms, 340–341
    slave KDCs, 342–343
planning *(Continued)*
  PAM, 280
  RBAC, 163–165
pluggable authentication module, *See* PAM
plugin_list option, SASL and, 291
plugins
  adding to KMF, 262–263
  auditing, 520–521
  Cryptographic Framework, 215
  managed in KMF, 251
  removing from KMF, 262–263
  SASL and, 290
plus sign (+)
  audit class prefix, 601–602
  entry in sulog file, 64
  file permissions symbol, 118
plus sign (+) in audit class prefixes, 563
policies
  administering, 443–479
  creating (Kerberos), 453
  creating new (Kerberos), 465–466
  deleting, 468
  for auditing, 534–537
  modifying, 467–468
  on devices, 76–77
  overview, 32–33
  passwords and, 486
  SEAM Tool panels for, 469–472
  specifying password algorithm, 61–64
  task map for administering, 461
  viewing attributes, 463–465
  viewing list of, 461–463
policy
  definition in Cryptographic Framework, 216
  definition in Oracle Solaris, 32–33
policy.conf file
  description, 203–204, 204
  keywords
    for password algorithms, 41
    for privileges, 203, 207
    for RBAC authorizations, 203
    for rights profiles, 203
    for workstation owner, 203
  specifying encryption algorithms in, 61–62

policy.conf file *(Continued)*
- specifying password algorithm
  - in naming services, 62–63
- specifying password algorithms, 61–62

port forwarding
- configuring in Secure Shell, 300
- Secure Shell, 307

Port keyword, Secure Shell, 317

ports, for Kerberos KDC, 342

post-selection in auditing, 517

postdated ticket
- definition, 502
- description, 328

pound sign (#)
- device_allocate file, 92
- device_maps file, 91

ppriv command
- for debugging, 193
- listing privileges, 191

praudit command
- converting audit records to readable format, 579
- description, 600
- piping auditreduce output to, 579
- use in a script, 579
- viewing audit records, 578–580
- XML format, 579

PreferredAuthentications keyword, ssh_config file, 317

prefixes for audit classes, 601–602

preselecting, audit classes, 545–546

preselection in auditing, 517

preselection mask (auditing), description, 604

PreUserauthHook keyword, ssh_config file, 317

preventing
- audit trail overflow, 581–582
- kernel software provider use, 243–245
- use of hardware mechanism, 246–248

primary, in principal names, 332–333

principal
- adding administration, 356, 362
- adding service principal to keytab, 473, 474–475
- administering, 443–479
- automating creation of, 449
- creating, 453–455

principal *(Continued)*
- creating clntconfig, 357, 363
- creating host, 356, 363
- deleting, 457–458
- duplicating, 456
- Kerberos, 332–333
- modifying, 456–457
- principal name, 332–333
- removing from keytab file, 476
- removing service principal from keytab, 476
- SEAM Tool panels for, 469–472
- service principal, 332
- setting up defaults, 458–459
- task map for administering, 448–449
- user ID comparison, 378
- user principal, 332
- viewing attributes, 451–453
- viewing list of, 449–451
- viewing sublist of principals, 450

principal file, description, 498

principal.kadm5 file, description, 498

principal.kadm5.lock file, description, 498

principal.ok file, description, 498

principal.ulog file, description, 498

principle of least privilege, 146

Printer Management (RBAC), rights profile, 198

printing, audit log, 579

PrintLastLog keyword, ssh_config file, 317

PrintMotd keyword, sshd_config file, 317

priv.debug entry, syslog.conf file, 207

PRIV_DEFAULT keyword
- policy.conf file, 203, 207

PRIV_LIMIT keyword
- policy.conf file, 203, 207

PRIV_PROC_LOCK_MEMORY privilege, 148–149

privacy
- availability, 492
- Kerberos and, 327
- security service, 334

private keys
- *See also* secret keys
- definition in Kerberos, 501
- Secure Shell identity files, 320

private protection level, 492

privilege audit token, 614
privilege checking, in applications, 142
privilege sets
  adding privileges to, 151
  basic, 150
  effective, 149
  inheritable, 149
  limit, 149
  listing, 150
  permitted, 149
  removing privileges from, 152
privileged application
  authorization checking, 142
  description, 137
  ID checking, 141
  privilege checking, 142
privileged ports, alternative to Secure RPC, 52
privileges
  adding to command, 171
  administering, 191
  assigning to a command, 151
  assigning to a script, 152
  assigning to a user, 151
  assigning to role, 179
  assigning to user, 180–181
  auditing and, 207–208
  categories, 146
  commands, 206
  compared to superuser model, 145–153
  debugging, 153, 193
  description, 137, 146, 147
  determining directly assigned ones, 188–189
  devices and, 152–153
  differences from superuser model, 147
  effects on SEAM Tool, 472
  escalation, 208
  executing commands with privilege, 151
  files, 207
  finding missing, 193–194
  how to use, 187
  implemented in sets, 149
  inherited by processes, 150
  limiting use in a rights profile, 171
  listing, 187–188
privileges *(Continued)*
  listing on a process, 191–192
  PRIV_PROC_LOCK_MEMORY, 148–149
  processes with assigned privileges, 150
  programs aware of privileges, 150
  protecting kernel processes, 145
  removing from a user, 152
  removing from basic set, 171
  removing from limit set, 171, 180
  task map, 186
  troubleshooting
    to users, 173–176
  troubleshooting requirements for, 193–194
  using in shell script, 194–195
PROC privileges, 147
process audit characteristics
  audit session ID, 605
  audit user ID, 605
  process preselection mask, 604
  terminal ID, 605
process audit token, format, 614
process preselection mask, description, 604
process privileges, 147
process rights management, *See* privileges
processing time costs, of audit service, 537
prof_attr database
  description, 203
  summary, 201
profile shells
  description, 144
  opening, 160–162
  restricting rights, 182–183
  restricting users to desktop applications, 181–182
profiles, *See* rights profiles
profiles command, description, 204
PROFS_GRANTED keyword, policy.conf file, 203
programs
  checking for RBAC authorizations, 172
  privilege-aware, 149, 150
project.max-locked-memory resource
  control, 148–149
PROM security mode, 66–68
propagation
  KDC database, 344

propagation *(Continued)*
  Kerberos database, 403–405
protecting
  32–bit executables from compromising security, 121
  BIOS, pointer to, 66–67
  by using passwords with Cryptographic Framework, 252–253
  contents of keystore, 256
  files with Cryptographic Framework, 222
  PROM, 66–67
  system from risky programs, 127–128
protecting files
  user procedures, 122
  with UFS ACLs, 119–120
  with UNIX permissions, 113–119, 122
  with UNIX permissions task map, 122
protection level
  clear, 492
  private, 492
  safe, 492
  setting in `ftp`, 492
`Protocol` keyword, Secure Shell, 317
providers
  adding library, 241
  adding software provider, 240–241
  adding user-level software provider, 241
  connecting to Cryptographic Framework, 218
  definition as plugins, 215
  definition in Cryptographic Framework, 216
  disabling hardware mechanisms, 246–248
  listing hardware providers, 245–246
  listing in Cryptographic Framework, 236–239
  preventing use of kernel software provider, 243–245
  registering, 218
  restoring use of kernel software provider, 243
  signing, 218
proxiable ticket, definition, 503
proxy ticket, definition, 503
`ProxyCommand` keyword, `ssh_config` file, 317
pseudo-tty, use in Secure Shell, 313
`PubkeyAuthentication` keyword, Secure Shell, 317
`public` audit policy
  description, 536

`public` audit policy *(Continued)*
  read-only events, 536
public directories
  auditing, 518
  sticky bit and, 117
public key authentication, Secure Shell, 294
public key cryptography
  `AUTH_DH` client-server session, 269–271
  changing NFS public keys and secret keys, 269
  common keys
    calculation, 270
  database of public keys for Secure RPC, 269
  generating keys
    conversation keys for Secure NFS, 269
    using Diffie-Hellman, 269
  NFS secret keys, 269
public key technologies, *See* PKI
public keys
  changing passphrase, 304
  DH authentication and, 268–271
  generating public-private key pair, 302–304
  Secure Shell identity files, 320
public objects, auditing, 518
`publickey` map, DH authentication, 268–271
`pwcheck_method` option, SASL and, 291

Q

`qsize` attribute, audit plugins, 551–552
quoting syntax in BART, 110–111

R

`-R` option
  `bart create`, 99, 102
  `ssh` command, 307–308
`-r` option
  `bart create`, 102
  `passwd` command, 39
random numbers
  `dd` command, 222–224
  `pktool` command, 224–228

RBAC
- adding privileged users, 180
- adding roles, 165–167
- administration commands, 204–205
- audit profiles, 600
- auditing roles, 169–170
- authorization database, 202–203
- authorizations, 140–141
- basic concepts, 137–139
- changing role passwords, 177–178
- checking scripts or programs for authorizations, 172
- commands for managing, 204–205
- compared to superuser model, 133–137
- configuring, 163–176
- creating rights profiles, 170–171
- databases, 201–204
- defaults, 156–162
- elements, 137–139
- gaining administrative rights, 160–162
- modifying roles, 178–179
- modifying users, 179–181
- naming services and, 201
- planning, 163–165
- profile shells, 144
- restricting rights, 182–183
- restricting users to desktop applications, 181–182
- rights profile database, 203
- rights profiles, 142–143
- securing scripts, 172
- troubleshooting, 173–176
- using user password to assume role, 183–184
- using user password to use rights profile, 184
- viewing all RBAC security attributes, 156–157
- viewing your rights, 157–159

RC4, *See* ARCFOUR kernel provider

`rcp` command
- Kerberos and, 490–493, 499

`rd` audit event modifier, 611

`read_kt` command, 477, 478

read permissions, symbolic mode, 118

readable audit record format, converting audit records to, 579

realms (Kerberos)
- configuration decisions, 340–341
- configuring cross-realm authentication, 370–372
- contents of, 334
- direct, 371–372
- hierarchical, 370–371
- hierarchical or nonhierarchical, 333
- hierarchy, 341
- in principal names, 332–333
- mapping host names onto, 341
- names, 340
- number of, 340
- requesting tickets for specific, 492
- servers and, 333–334

`reauth_timeout` option, SASL and, 291

redirecting arrow (>), preventing redirection, 46

reducing
- audit files, 574–576
- disk space required for audit files, 593–594
- storage-space requirements for audit files, 538

refreshing
- audit service, 568–570, 569–570
- cryptographic services, 248

registering providers, Cryptographic Framework, 218

`RekeyLimit` keyword, `ssh_config` file, 317

`rem_drv` command, description, 88

remote logins
- authentication, 51–52
- authorization, 51–52
- preventing superuser from, 65–66
- security and, 271

`RemoteForward` keyword, `ssh_config` file, 317

removing
- audit events from `audit_event` file, 592–593
- cryptographic providers, 242, 243
- device policy, 77–78
- plugins from KMF, 262–263
- policy from device, 77–78
- principals with `ktremove` command, 476
- privileges from basic set, 171
- privileges from limit set, 171, 180
- service principal from keytab file, 476
- software providers
  - permanently, 244, 245

removing, software providers *(Continued)*
        temporarily, 244
    user-specific auditing, 548
renewable ticket, definition, 503
replacing
    preselected audit classes, 545–546
    superuser with roles, 163–165
replayed transactions, 271
reporting tool, *See* `bart compare`
reports, BART, 95
repository, installing third-party providers, 240
required control flag, PAM, 284
requisite control flag, PAM, 285
resource controls
    privileges, and, 148–149
    `project.max-locked-memory`, 148–149
    `zone.max-locked-memory`, 148–149
restarting
    cryptographic services, 248
    `ssh` service, 300
    `sshd` daemon, 300
restoring, cryptographic providers, 243
restricted shell (`rsh`), 46
restricting
    remote superuser access, 65–66
    superuser, 64–66
    user privileges, 171
restricting access for KDC servers, 422–423
`RETRIES` in Secure Shell, 319
`return` audit token, format, 614
`rewoffl` option
    `mt` command
        tape device cleanup and, 93
`.rhosts` file, description, 321
`RhostsAuthentication` keyword, Secure Shell, 317
`RhostsRSAAuthentication` keyword, Secure Shell, 317
right, *See* rights profiles
rights
    restricting administrator to explicitly assigned, 182–183
    restricting users to desktop applications, 181–182
rights profiles
    All, 198
    assigning to trusted users, 135, 165

rights profiles *(Continued)*
    for audit service, 600
    authenticating with user's password, 184
    Basic Solaris User, 198
    changing contents of, 170–171
    Console User, 198, 199
    contents of typical, 197
    databases
        See `prof_attr` database and `exec_attr` database
    description, 137, 142–143
    Device Management, 89
    Device Security, 80, 89
    major rights profiles descriptions, 197
    modifying, 170–171
    Operator, 198
    order of search, 199
    preventing privilege escalation, 135, 165
    Printer Management, 198
    Stop, 198, 199
    System Administrator, 198
    troubleshooting, 173–176
    using the System Administrator profile, 66
    viewing contents, 199
`rlogin` command
    Kerberos and, 490–493, 499
`rlogind` daemon, Kerberos and, 500
role-based access control, *See* RBAC
`roleadd` command
    description, 204
    using, 166
`roleauth` keyword, passwords for roles, 183–184
`rolemod` command
    changing properties of role, 178, 183
    description, 204
    passwords for roles, 183–184
roles
    adding to a user, 180
    assigning privileges to, 179
    assigning with `usermod` command, 167–169
    assuming, 159–160
    assuming after login, 143
    assuming in a terminal window, 144, 159–160
    assuming `root` role, 160
    auditing, 169–170

roles *(Continued)*
  authenticating with user's password, 183–184
  changing password of, 177–178
  changing properties of, 178–179
  creating, 165–167
    Crypto Management role, 168–169
  description, 143
  determining directly assigned privileges, 189
  determining role's privileged commands, 190
  listing local roles, 159, 204
  making root role into user, 184–186
  modifying, 178–179
  recommended roles, 134
  summary, 137
  use in RBAC, 134
  using an assigned role, 159–160
  using to access the hardware, 66–67
  using user password, 139
roles command
  description, 204
  using, 159
root account, description, 42
root principal, adding to host's keytab, 473
root role
  changing from root user, 186
root role
  changing password, 56–57
root role
  changing to root user, 184–186
root role
  provided role, 135
root role (RBAC)
  assuming role, 160
  troubleshooting, 186
root user
  changing into root role, 186
  displaying access attempts on console, 65–66
  monitoring su command attempts, 45, 64–65
  replacing in RBAC, 143
  restricting access, 50
  restricting remote access, 65–66
  tracking logins, 45
RSA kernel provider, 236
RSAAuthentication keyword, Secure Shell, 317
rsh command
  Kerberos and, 490–493, 499
rsh command (restricted shell), 46
rshd daemon, Kerberos and, 500
rstchown system variable, 124
rules file (BART), 97–98
rules file attributes, *See* keywords
rules file format (BART), 109–111
rules file specification language, *See* quoting syntax

S

-S option, st_clean script, 94
-s option
  audit command, 568–570, 571–572
safe protection level, 492
SASL
  environment variable, 290
  options, 291
  overview, 289
  plug-ins, 290
saslauthd_path option, SASL and, 291
saving, failed login attempts, 59–60
scope (RBAC), description, 144
scp command
  copying files with, 308
  description, 322
scripts
  audit_warn script, 553, 599
  checking for RBAC authorizations, 172
  device-clean scripts
    See also device-clean scripts
  for cleaning devices, 93–94
  monitoring audit files example, 538
  processing praudit output, 579
  running with privileges, 152
  securing, 172
  use of privileges in, 194–195
SCSI devices, st_clean script, 93
SEAM Tool
  and limited administration privileges, 472–473
  and list privileges, 472
  and X Window system, 444–445
  command-line equivalents, 444–445

SEAM Tool *(Continued)*
- context-sensitive help, 445
- creating a new policy, 453, 465–466
- creating a new principal, 453–455
- default values, 447
- deleting a principal, 457–458
- deleting policies, 468
- displaying sublist of principals, 450
- duplicating a principal, 456
- files modified by, 445
- Filter Pattern field, 450
- `gkadmin` command, 443
- `.gkadmin` file, 445
- help, 445–446
- Help Contents, 446
- how affected by privileges, 472
- `kadmin` command, 443
- login window, 447
- modifying a policy, 467–468
- modifying a principal, 456–457
- online help, 445–446
- or `kadmin` command, 444
- overview, 444–447
- panel descriptions, 469–472
- privileges, 472
- setting up principal defaults, 458–459
- starting, 447
- table of panels, 469–472
- viewing a principal's attributes, 451–453
- viewing list of policies, 461–463
- viewing list of principals, 449–451
- viewing policy attributes, 463–465

secret keys
- creating, 222–224, 224–228
- generating
  - using the `dd` command, 222–224
  - using the `pktool` command, 224–228
- generating for Secure RPC, 269

Secure by Default installation option, 47

secure connection
- across a firewall, 309
- logging in, 304–305

Secure NFS, 268

Secure RPC
- alternative, 52
- and Kerberos, 268
- description, 267
- implementation of, 269–271
- keyserver, 269
- overview, 51–52

Secure Shell
- administering, 311–313
- administrator task map, 297
- authentication
  - requirements for, 294–295
- authentication methods, 294–295
- authentication steps, 312–313
- basis from OpenSSH, 295–296
- changes in current release, 295–296
- changing passphrase, 304
- command execution, 313
- configuring clients, 313–314
- configuring port forwarding, 300
- configuring server, 314
- connecting across a firewall, 309
- connecting outside firewall
  - from command line, 310
  - from configuration file, 309–310
- copying files, 308
- creating keys, 302–304
- data forwarding, 313
- description, 293
- files, 320
- forwarding mail, 307
- generating keys, 302–304
- keywords, 314–319
- local port forwarding, 307
- logging in fewer prompts, 305–306
- logging in to remote host, 304–305
- login environment variables and, 318–319
- naming identity files, 320
- protocol versions, 293
- public key authentication, 294
- remote port forwarding, 307–308
- `scp` command, 308
- specifying exceptions to system defaults, 300–301
- TCP and, 300

Secure Shell *(Continued)*
typical session, 311–313
user procedures, 301–302
using port forwarding, 307–308
using without password, 305–306
securing
logins task map, 56
network at installation, 47
passwords task map, 56
scripts, 172
security
across insecure network, 309
auditing, 515–528
auditing and, 524
BART, 95–112
computing digest of files, 229–230
computing MAC of files, 230–232
Cryptographic Framework, 213–219
device allocation, 75–94
devices, 42–44
DH authentication, 269–271
encrypting files, 232–235
installation options, 47
key management framework, 249–263
`netservices limited` installation option, 47
NFS client-server, 269–271
password encryption, 40
policy overview, 32–33
preventing remote login, 65–66
protecting against denial of service, 47
protecting against Trojan horse, 46
protecting devices, 93–94
protecting hardware, 66–68
protecting PROM, 66–68
Secure by Default, 47
Secure Shell, 293–310
system hardware, 66–68
systems, 37
security attributes
checking for, 141
considerations when directly assigning, 144–145
description, 137
listing all RBAC, 156–157
Network Security rights profile, 139
security attributes *(Continued)*
order of search, 199
privileges on commands, 142
special ID on commands, 142
usability considerations when directly assigning, 145
using to mount allocated device, 80
security mechanism, specifying with `-m` option, 492
security modes, setting up environment with multiple, 380–381
security policy, default (RBAC), 201
security service, Kerberos and, 334
selecting
audit classes, 545–546
audit records, 576–578
events from audit trail, 576–578
semicolon (;), `device_allocate` file, 92
`sendmail` command, authorizations required, 206
`seq` audit policy
and `sequence` token, 536, 614
description, 536
`sequence` audit token
and `seq` audit policy, 614
format, 614–615
`ServerAliveCountMax` keyword, `ssh_config` file, 317
`ServerAliveInterval` keyword, `ssh_config` file, 317
`ServerKeyBits` keyword, `sshd_config` file, 318
servers
`AUTH_DH` client-server session, 269–271
configuring for Secure Shell, 314
definition in Kerberos, 501
gaining access with Kerberos, 506–508
obtaining credential for, 507–508
realms and, 333–334
service
definition in Kerberos, 501
disabling on a host, 477–479
obtaining access for specific service, 508
service keys
definition in Kerberos, 501
keytab files and, 473–479
service management facility, refreshing Cryptographic Framework, 240
Service Management Facility (SMF), *See* SMF

service principal
  adding to keytab file, 473, 474–475
  description, 332
  planning for names, 341–342
  removing from keytab file, 476
session ID, audit, 605
session keys
  definition in Kerberos, 501
  Kerberos authentication and, 505–506
-setflags option, auditconfig command, 545–546
setgid permissions
  absolute mode, 119, 127
  description, 116
  security risks, 116
  symbolic mode, 118
-setnaflags option, auditconfig command, 545–546
setpin subcommand, pktool command, 256–257
-setplugin option
  auditconfig command, 562–563, 563–564
-setpolicy option, auditconfig command, 549–551
setting
  arge policy, 588
  argv policy, 588
  audit policy, 549–551
  audit queue controls, 551–552
  principal defaults (Kerberos), 458–459
setuid permissions
  absolute mode, 119, 127
  description, 116
  finding files with permissions set, 128
  security risks, 47, 116
  symbolic mode, 118
sftp command
  auditing file transfers, 595–596
  copying files with, 308
sftpcommand, description, 323
sh command, privileged version, 144
SHA1 kernel provider, 236
SHA2 kernel provider, 236
sharing files
  and network security, 49
  with DH authentication, 274–275
shell, privileged versions, 144
shell commands, passing parent shell process number, 191
shell process, listing its privileges, 191–192
shell scripts, writing privileged, 195
shosts.equiv file, description, 321
.shosts file, description, 321
signing
  PKCS #10 CSR, 261–262
    using the pktool command, 261–262
signing providers, Cryptographic Framework, 218
single-sign-on system, 490–495
  Kerberos and, 327
size of audit files
  reducing, 574–576
  reducing storage-space requirements, 538
slave_datatrans file
  description, 498
  KDC propagation and, 403–405
slave_datatrans_*slave* file, description, 498
slave KDCs
  automatically configuring, 364–365
  configuring, 366–369
  definition, 500
  interactively configuring, 365–366
  master KDC and, 333–334
  or master, 350
  planning for, 342–343
  swapping with master KDC, 399–403
slot, definition in Cryptographic Framework, 216
SMF
  auditd service, 597–598
  Cryptographic Framework service, 217
  device allocation service, 89
  enabling keyserver, 272
  kcfd service, 217
  managing Secure by Default configuration, 47
  restarting Cryptographic Framework, 248
  restarting Secure Shell, 300
  ssh service, 300
socket audit token, 615
solaris.device.revoke authorization, 90
sp audit event modifier, 611
special permissions
  setgid permissions, 116

special permissions *(Continued)*
- setuid permissions, 116
- sticky bit, 116–117

square brackets ([]), auditrecord output, 607
sr_clean script, description, 94
ssh-add command
- description, 322
- example, 305–306, 306
- storing private keys, 305–306

ssh-agent command
- description, 322
- from command line, 305–306

ssh command
- description, 322
- overriding keyword settings, 323
- port forwarding options, 307–308
- using, 304–305
- using a proxy command, 310

.ssh/config file
- description, 321
- override, 321

ssh_config file
- configuring Secure Shell, 313–314
- host-specific parameters, 318
- keywords, 314–319
  - *See* specific keyword
- override, 321

.ssh/environment file, description, 321
ssh_host_dsa_key file, description, 320
ssh_host_dsa_key.pub file, description, 320
ssh_host_key file, override, 321
ssh_host_key.pub file, description, 320
ssh_host_rsa_key file, description, 320
ssh_host_rsa_key.pub file, description, 320
.ssh/id_dsa file, 322
.ssh/id_rsa file, 322
.ssh/identity file, 322
ssh-keygen command
- description, 322
- passphrase protection, 295
- using, 302–304

ssh-keyscan command, description, 322
ssh-keysign command, description, 322
.ssh/known_hosts file
- description, 320
- override, 322

ssh_known_hosts file, 320
.ssh/rc file, description, 321
sshd command, description, 322
sshd_config file
- description, 320
- keywords, 314–319
  - *See* specific keyword
- overrides of /etc/default/login entries, 318–319

sshd.pid file, description, 320
sshrc file, description, 321
st_clean script
- description, 93
- for tape drives, 93

standard cleanup, st_clean script, 94
starting
- auditing, 571–572
- device allocation, 80
- KDC daemon, 369, 413
- Secure RPC keyserver, 272

stash file
- creating, 369, 413
- definition, 500

sticky bit permissions
- absolute mode, 119, 127
- description, 116–117
- symbolic mode, 118

Stop (RBAC), rights profile, 198
storage costs, and auditing, 537–538
storage overflow prevention, audit trail, 581–582
storing
- audit files, 531–532, 556–559
- passphrase, 233

StrictHostKeyChecking keyword, ssh_config file, 318
StrictModes keyword, sshd_config file, 318
su command
- displaying access attempts on console, 65–66
- in role assumption, 159–160
- monitoring use, 64–65

su file, monitoring su command, 64–65
subject audit token, format, 615

Subsystem keyword, sshd_config file, 318
success, audit class prefix, 601–602
sufficient control flag, PAM, 285
sulog file, 64–65
  monitoring contents of, 64
Sun Crypto Accelerator 1000 board, listing mechanisms, 246–248
Sun Crypto Accelerator 6000 board
  hardware plugin to Cryptographic Framework, 215
  listing mechanisms, 245–246
Sun Crypto Accelerator 6000 card, Secure Shell and FIPS-140, 296
SUPATH in Secure Shell, 319
superuser
  compared to privilege model, 145–153
  compared to RBAC model, 133–137
  differences from privilege model, 147
  eliminating in RBAC, 143
  monitoring access attempts, 65–66
  monitoring and restricting, 64–66
  troubleshooting becoming root as a role, 186
  troubleshooting remote access, 66
svc:/system/device/allocate, device allocation service, 89
svcadm command
  administering Cryptographic Framework, 217
  enabling Cryptographic Framework, 248
  enabling keyserver daemon, 272
  refreshing Cryptographic Framework, 240–241
  restarting
    Secure Shell, 300
    syslog daemon, 60, 563
svcs command
  listing cryptographic services, 248
  listing keyserver service, 272
swapping master and slave KDCs, 399–403
symbolic links, file permissions, 115
symbolic mode
  changing file permissions, 118, 124–125, 125
  description, 118
synchronizing clocks
  master KDC, 357, 364
  overview, 398–399
  slave KDC, 369, 413
SYS privileges, 147
syslog.conf file
  and auditing, 600
  audit.notice level, 563
  executable stack messages, 121
  kern.notice level, 121
  priv.debug entry, 207
  privilege debugging, 207
  saving failed login attempts, 60–61
SYSLOG_FAILED_LOGINS
  in Secure Shell, 319
  system variable, 60
SyslogFacility keyword, sshd_config file, 318
System Administrator (RBAC)
  protecting hardware, 66
  recommended role, 135
  rights profile, 198
system calls
  argument audit token, 609
  exec_args audit token, 610
  exec_env audit token, 610
  ioctl to clean audio device, 94
  return audit token, 614
system hardware, controlling access to, 66–68
system properties, privileges relating to, 147
system security
  access, 37
  changing
    root password, 56–57
  displaying
    user's login status, 57–58
    users with no passwords, 58
  firewall systems, 52–53
  hardware protection, 38, 66–68
  login access restrictions, 38
  machine access, 38
  overview, 37
  password encryption, 40
  passwords, 39
  privileges, 145–153
  protecting from risky programs, 127–128
  restricted shell, 46
  restricting remote root access, 65–66
  role-based access control (RBAC), 45, 133–137

system security *(Continued)*
- root access restrictions, 50, 65–66
- saving failed login attempts, 59–60
- special accounts, 41
- su command monitoring, 45, 64–65
- task map, 127
- UFS ACLs, 119–120

System V IPC
- ipc audit token, 612–613
- IPC_perm audit token, 613
- privileges, 147

system variables
- *See also* variables
- CRYPT_DEFAULT, 61
- KEYBOARD_ABORT, 67–68
- noexec_user_stack, 129–130
- noexec_user_stack_log, 129
- rstchown, 124
- SYSLOG_FAILED_LOGINS, 60

/system/volatile/sshd.pid file, description, 320

systems, protecting from risky programs, 127–128

T

-T option
- encrypt command, 233
- mac command, 231

-t option, audit command, 571

tables, gsscred, 511

tail command, example of use, 539

tape drives
- allocating, 85
- cleaning of data, 93
- device-clean scripts, 93

task maps
- administering Cryptographic Framework, 235–236
- administering policies (Kerberos), 461
- administering principals (Kerberos), 448–449
- administering Secure RPC, 272
- auditing, 541
- configuring audit logs, 556
- configuring auditing, 542–543
- configuring device policy, 76
- configuring devices, 75

task maps *(Continued)*
- configuring Kerberos NFS servers, 375
- configuring RBAC, 163
- configuring Secure Shell, 297
- Cryptographic Framework, 221
- device allocation, 79–80
- device policy, 76
- devices, 75
- Kerberos configuration, 349–350
- Kerberos maintenance, 350
- managing and using privileges, 186
- managing audit records, 572–573
- managing device allocation, 79–80
- managing device policy, 76
- managing RBAC, 176–177
- PAM, 279
- planning auditing, 529–534
- protecting against programs with security risk, 127
- protecting files with cryptographic mechanisms, 222
- protecting files with UNIX permissions, 122
- Secure Shell, 297
- securing logins and passwords, 56
- securing systems, 55–56
- system access, 55–56
- troubleshooting auditing, 582–583
- Using BART task map, 99
- using RBAC, 155–156
- using Secure Shell, 301–302
- using the Cryptographic Framework, 221
- using the default RBAC configuration, 156
- Using the Key Management Framework (Task Map), 252–253

TCP
- addresses, 612
- Secure Shell and, 300, 313

telnet command
- Kerberos and, 490–493, 499

telnetd daemon, Kerberos and, 500

temporary audit policy
- active audit policy, 549–551
- setting, 551

terminal ID, audit, 605

terminology
- authentication-specific, 501–502
- Kerberos, 500–505
- Kerberos-specific, 500

test manifests, 97
text audit token, format, 615
TGS, getting credential for, 506–507
TGT, in Kerberos, 329–330
ticket file, *See* credential cache
ticket-granting service, *See* TGS
ticket-granting ticket, *See* TGT
tickets
- creating, 481–482
- creating with kinit, 482
- definition, 328
- definition in Kerberos, 501
- destroying, 484
- -F option or -f option, 491
- file
  - *See* credential cache
- forwardable, 328, 482, 493, 502
- initial, 502
- invalid, 502
- -k option, 492
- klist command, 483–484
- lifetime, 503–504
- maximum renewable lifetime, 504
- obtaining, 481–482
- or credentials, 329
- postdatable, 502
- postdated, 328
- proxiable, 503
- proxy, 503
- renewable, 503
- requesting for specific realm, 492
- types of, 502–505
- viewing, 483–484
- warning about expiration, 392

time stamps, audit files, 605
TIMEOUT in Secure Shell, 319
/tmp/krb5cc_*uid* file, description, 498
/tmp/ovsec_adm.*xxxxx* file, description, 498
TMPFS file system, security, 117
token, definition in Cryptographic Framework, 216
trail audit policy
- and trailer token, 536
- description, 536

trailer audit token
- format, 616
- order in audit record, 616
- praudit display, 616

transparency, definition in Kerberos, 328
Trojan horse, 46
troubleshooting
- active plugin, 584
- allocating a device, 85
- audit classes
  - customized, 554, 584
- auditing, 582–583
- computer break-in attempts, 59–60
- encrypt command, 234
- finding files with setuid permissions, 128
- Kerberos, 439
- lack of privilege, 193–194
- list_devices command, 81
- mounting a device, 86
- praudit command, 580
- preventing programs from using executable stacks, 129–130
- privilege requirements, 193–194
- remote superuser access, 66
- root as a role, 186
- security properties, 173–176
- terminal where su command originated, 65
- user running privileged commands, 189–190

truss command, for privilege debugging, 193
trusted hosts, 53
types of tickets, 502–505
TZ in Secure Shell, 319

U

-U option, allocate command, 90
UDP
- addresses, 612
- port forwarding and, 300
- Secure Shell and, 300
- using for remote audit logs, 521

umask value
  and file creation, 117
  typical values, 117
umount command, with security attributes, 80
uninstalling, cryptographic providers, 242
UNIX file permissions, *See* files, permissions
unmounting, allocated devices, 87
update_drv command
  description, 88
  using, 77–78
URL for online help, Graphical Kerberos Tool, 347
use_authid option, SASL and, 291
use of authorization audit token, 616
use of privilege audit token, 616
UseOpenSSLEngine keyword, Secure Shell, 318
UsePrivilegedPort keyword, Secure Shell, 318
user accounts
  See also users
  changing root password, 56–57
  displaying login status, 57–58
user_attr database
  description, 201, 202
  listing user exceptions to audit
    preselection, 546–549
user_attr file, exceptions to system-wide audit
  classes, 519
user audit token, 616
user classes of files, 114
user database (RBAC), *See* user_attr database
user ID
  audit ID and, 515–516, 605
  in NFS services, 378
user ID numbers (UIDs), special accounts and, 42
User keyword, ssh_config file, 318
user principal, description, 332
user procedures
  adding plugins to KMF, 262–263
  allocating devices, 79–84
  assuming a role, 159–160
  chkey command, 274
  computing digest of a file, 229–230
  computing MAC of a file, 230–232
  creating self-signed certificate, 253–254
  decrypting files, 232–235

user procedures *(Continued)*
  encrypting files, 222
  encrypting NIS user's private key, 274
  exporting certificates, 255–256
  generating a symmetric key
    using the dd command, 222–224
    using the pktool command, 224–228
  generating passphrase for keystore, 256–257
  importing certificates, 254–255
  protecting files, 122
  using an assigned role, 159–160
  using pktool command, 252–253
  using Secure Shell, 301–302
user rights management, *See* privileges
User Security rights profile, modifying audit
  preselection for users, 546–549
useradd command, description, 205
userattr command
  description, 204
  displaying exceptions to system-wide
    auditing, 543–544
userdel command, description, 205
UserKnownHostsFile keyword, ssh_config file, 318
UserKnownHostsFile2 keyword, *See*
  UserKnownHostsFile keyword
usermod command
  audit_flags keyword, 546–549
usermod command
  changing user's RBAC properties, 179
  description, 205
  exceptions to system-wide auditing, 519
  limiting user to desktop icons only, 182
usermod command
  specifying user exceptions to audit
    preselection, 546–549
  using caret (^) prefix for audit_flags
    exception, 547
usermod command
  using to assign role, 167–169
users
  allocating devices, 84–85
  assigning allocate authorization to, 80–81
  assigning more than one role, 180
  assigning privileges to, 180–181

users *(Continued)*
- assigning RBAC defaults, 203–204
- assigning rights profiles, 180
- auditing all of their commands, 587–589
- auditing individual users, 548
- authenticating to rights profile, 184
- authenticating to role, 183–184
- basic privilege set, 150
- computing digest of files, 229–230
- computing MAC of files, 230–232
- creating
  - root user, 184–186
- creating rights profile for a group, 549
- deallocating devices, 87
- determining directly assigned privileges, 188–189
- determining own privileged commands, 189–190
- disabling login, 58–59
- displaying login status, 57–58
- encrypting files, 232–235
- exceptions to Secure Shell defaults, 300–301
- generating a symmetric key, 224–228
- having no passwords, 58
- initial inheritable privileges, 150
- modifying audit preselection mask of, 546–549
- modifying properties (RBAC), 179–181
- mounting allocated devices, 85–86
- removing audit flags, 548
- restricting basic privileges, 171
- troubleshooting running privileged commands, 189–190
- unmounting allocated devices, 87
- using rights profile, 184

UseRsh keyword, ssh_config file, 318

using
- allocate command, 84–85
- BART, 98
- cryptoadm command, 235
- Cryptographic Framework task map, 221
- dd command, 222–224
- deallocate command, 87
- default RBAC configuration task map, 156
- device allocation, 84–85
- digest command, 229–230
- encrypt command, 232–235

using *(Continued)*
- file permissions, 121–130
- mac command, 230–232
- new password algorithm, 62
- pktool command, 224–228, 257–261
- ppriv command, 191
- privileges task map, 187
- RBAC defaults, 156–162
- RBAC task map, 155–156
- rolemod command, 179
- Secure Shell task map, 301–302
- ssh-add command, 305–306
- ssh-agent daemon, 305–306
- truss command, 193
- umount command, 87
- usermod command, 180–181

Using the Key Management Framework (Task Map), 252–253

/usr/bin/ftp command, Kerberos and, 499

/usr/bin/kdestroy command, Kerberos and, 499

/usr/bin/kinit command, Kerberos and, 499

/usr/bin/klist command, Kerberos and, 499

/usr/bin/kpasswd command, Kerberos and, 499

/usr/bin/ktutil command, Kerberos and, 499

/usr/bin/kvno command, Kerberos and, 499

/usr/bin/rcp command, Kerberos and, 499

/usr/bin/rlogin command, Kerberos and, 499

/usr/bin/rsh command, Kerberos and, 499

/usr/bin/telnet command, Kerberos and, 499

/usr/lib/kprop command, description, 499

/usr/lib/krb5/kadmind daemon, Kerberos and, 500

/usr/lib/krb5/kpropd daemon, Kerberos and, 500

/usr/lib/krb5/krb5kdc daemon, Kerberos and, 500

/usr/lib/krb5/ktkt_warnd daemon, Kerberos and, 500

/usr/lib/libsasl.so library, overview, 289

/usr/sbin/gkadmin command, description, 499

/usr/sbin/gsscred command, description, 499

/usr/sbin/in.ftpd daemon, Kerberos and, 500

/usr/sbin/in.rlogind daemon, Kerberos and, 500

/usr/sbin/in.rshd daemon, Kerberos and, 500

/usr/sbin/in.telnetd daemon, Kerberos and, 500

/usr/sbin/kadmin command, description, 499

/usr/sbin/kadmin.local command, description, 499

/usr/sbin/kclient command, description, 499
/usr/sbin/kdb5_ldap_util command, description, 499
/usr/sbin/kdb5_util command, description, 499
/usr/sbin/kgcmgr command, description, 499
/usr/sbin/kproplog command, description, 499

V

v1 protocol, Secure Shell, 293
-v option
  digest command, 229
  mac command, 230
  ppriv command, 191
v2 protocol, Secure Shell, 293
/var/adm/auditlog file, text audit records, 563
/var/adm/loginlog file, saving failed login attempts, 59–60
/var/adm/messages file
  executable stack messages, 121
  troubleshooting auditing, 585
/var/adm/sulog file, monitoring contents of, 64
/var/krb5/.k5.*REALM* file, description, 498
/var/krb5/kadmin.log file, description, 498
/var/krb5/kdc.log file, description, 498
/var/krb5/principal file, description, 498
/var/krb5/principal.kadm5 file, description, 498
/var/krb5/principal.kadm5.lock file, description, 498
/var/krb5/principal.ok file, description, 498
/var/krb5/principal.ulog file, description, 498
/var/krb5/slave_datatrans file, description, 498
/var/krb5/slave_datatrans_*slave* file, description, 498
/var/log/authlog file, failed logins, 60–61
/var/log/syslog file, troubleshooting auditing, 585
variables
  adding to audit record, 535, 610
  auditing those associated with a command, 609–610
  for proxy servers and ports, 309
  KEYBOARD_ABORT, 67–68
  login and Secure Shell, 318–319
  noexec_user_stack, 121
variables *(Continued)*
  noexec_user_stack_log, 121
  rstchown, 124
  setting in Secure Shell, 319
verifiers
  description, 270
  returned to NFS client, 271
  window, 270
VerifyReverseMapping keyword, ssh_config file, 318
viewing
  audit record definitions, 573–574
  available cryptographic mechanisms, 238, 243
  binary audit files, 578–580
  contents of rights profiles, 199
  cryptographic mechanisms
    available, 238, 243
    existing, 237, 238, 243
    purpose, 238
  device allocation information, 81
  device policy, 76–77
  digest of a file, 229
  directly assigned privileges, 188
  existing cryptographic mechanisms, 238, 243
  file permissions, 122–123
  keylist buffer with list command, 477, 478
  list of policies, 461–463
  list of principals, 449–451
  MAC of a file, 231
  policy attributes, 463–465
  principal's attributes, 451–453
  privilege definitions, 187–188
  privileges, 186–187
  privileges in a shell, 189, 192
  privileges on a process, 191
  tickets, 483–484
  user's login status, 57–58
  users with no passwords, 58
  verbose listing of cryptographic mechanisms, 238
  XML audit records, 578
  your RBAC rights, 157–159
virus scanning
  configuring, 70–74
  described, 70
  engines, 69–70

virus scanning *(Continued)*
    files, 69–70
viruses
    denial of service attack, 47
    Trojan horse, 46
vnode audit token, format, 609

W

warn.conf file, description, 498
warning about ticket expiration, 392
wildcard characters
    for hosts in Secure Shell, 309
    in RBAC authorizations, 200
window verifier, 270
wr audit event modifier, 611
write permissions, symbolic mode, 118

X

X.509 v3 certificate, generating, 261–262
-X option, Kerberized commands, 492
X Window system, and SEAM Tool, 444–445
X11 forwarding
    configuring in ssh_config file, 315
    in Secure Shell, 313
X11DisplayOffset keyword, sshd_config file, 318
X11Forwarding keyword, sshd_config file, 318
X11UseLocalHost keyword, sshd_config file, 318
-x option, Kerberized commands, 492
xauth command, X11 forwarding, 318
XAuthLocation keyword, Secure Shell port forwarding, 318
xclient audit token, 617
XML format, audit records, 579
Xylogics tape drive device-clean script, 93

Z

ZFS File System Management rights profile, creating audit file systems, 556–559
ZFS file systems, creating for binary audit files, 556–559
ZFS Storage Management rights profile, creating pools for audit files, 556–559
zone.max-locked-memory resource control, 148–149
zonename audit policy
    description, 536
    using, 531, 600–601
zonename audit token, 617
zones
    auditing and, 526–527, 600–601
    configuring auditing in global zone, 550
    Cryptographic Framework and, 219
    cryptographic services and, 248
    devices and, 43
    perzone audit policy, 526–527, 531, 600–601
    planning auditing in, 530–531
    zonename audit policy, 531, 600–601