Solaris™ Security Toolkit 4.1 Administration Guide

Sun Microsystems, Inc.
www.sun.com

Part No. 817-7424-10
October 2004, Revision A

Submit comments about this document at: http://www.sun.com/hwdocs/feedback

Contents

Figures

Tables

Code Samples

Preface

This manual contains reference information for understanding and using Solaris Security Toolkit software. This manual is primarily intended for persons who use the Solaris Security Toolkit software to secure Solaris™ Operating System (OS) versions 8 through 9, such as administrators, consultants, and others, who are deploying new Sun systems or securing deployed systems. The instructions apply to using the software in either its JumpStart™ mode or standalone mode.

Before You Read This Book

You should be a Sun Certified System Administrator for Solaris™ or Sun Certified Network Administrator for Solaris™ Operating System. You should also have an understanding of standard network protocols and topologies.

Because this book is designed to be useful to people with varying degrees of experience or knowledge of security, your experience and knowledge determine how you use this book.

How This Book Is Organized

This manual serves as a user guide. Its chapters contain information, instructions, and guidelines for using the software to secure systems. This book is structured as follows:

Chapter 1 describes the design and purpose of the Solaris Security Toolkit software. It covers the key components, features, benefits, and supported platforms.

Chapter 2 provides a methodology for securing systems. It provides a process that you can apply before securing your systems using the Solaris Security Toolkit software.

Chapter 3 provides instructions for downloading, installing, and running the Solaris Security Toolkit software and other security-related software.

Chapter 4 provides information and procedures for reversing (undoing) the changes made by the Solaris Security Toolkit software during hardening runs.

Chapter 5 provides information for configuring and managing JumpStart servers to use the Solaris Security Toolkit software.

Chapter 6 describes how to audit (validate) a system's security using the Solaris Security Toolkit software. Use the information and procedures in this chapter for maintaining an established security profile after hardening.

Chapter 7 describes how to apply the information and expertise provided in earlier chapters to a realistic scenario for installing and securing a new system.

Using UNIX® Commands

This document might not contain information on basic UNIX® commands and procedures such as shutting down the system, booting the system, and configuring devices. Refer to the following for this information:

- Software documentation that you received with your system
- Solaris Operating System documentation, which is at

  `http://docs.sun.com`

Shell Prompts

Shell	Prompt
C shell	*machine-name*%
C shell superuser	*machine-name*#
Bourne shell and Korn shell	$
Bourne shell and Korn shell superuser	#

Typographic Conventions

Typeface*	Meaning	Examples
AaBbCc123	The names of commands, files, and directories; on-screen computer output	Edit your .login file. Use ls -a to list all files. % You have mail.
AaBbCc123	What you type, when contrasted with on-screen computer output	% **su** Password:
AaBbCc123	Book titles, new words or terms, words to be emphasized. Replace command-line variables with real names or values.	Read Chapter 6 in the *User's Guide*. These are called *class* options. You *must* be superuser to do this. To delete a file, type rm *filename*.

* The settings on your browser might differ from these settings.

Accessing Sun Documentation

You can view, print, or purchase a broad selection of Sun documentation, including localized versions, at:

http://www.sun.com/documentation

Third-Party Web Sites

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

Related Resources

Related publications and web sites are listed in this section.

Publications

- Andert, Donna, Wakefield, Robin, and Weise, Joel. "Trust Modeling for Security Architecture Development," Sun BluePrints™ OnLine, December 2002, `http://www.sun.com/blueprints/1202/817-0775.pdf`.
- Dasan, Vasanthan, Noordergraaf, Alex, and Ordica, Lou. "The Solaris Fingerprint Database - A Security Tool for Solaris Software and Files," Sun BluePrints OnLine, May 2001, `http://www.sun.com/blueprints/0501/Fingerprint.pdf`.
- Englund, Martin, "Securing Systems with Host-Based Firewalls - Implemented With SunScreen Lite 3.1 Software," Sun BluePrints OnLine, September 2001, `http://sun.com/blueprints/0901/sunscreenlite.pdf`.
- Garfinkel, Simon, and Spafford, Gene. *Practical UNIX and Internet Security*, 2nd Edition, O'Reilly & Associates, April 1996.
- Howard, John S., and Noordergraaf, Alex. *JumpStart Technology: Effective Use in the Solaris Operating Environment*, The Official Sun Microsystems Resource Series, Prentice Hall, October 2001.
- Moffat, Darren J., FOCUS on SUN: *Solaris BSM Auditing*, `http://www.securityfocus.com/infocus/1362`
- Noordergraaf, Alex. "Solaris™ Operating Environment Minimization for Security: A Simple, Reproducible and Secure Application Installation Methodology Updated for Solaris 8 Operating Environment," Sun BluePrints OnLine, November 2000, `http://sun.com/blueprints/1100/minimize-updt1.pdf`.

- Noordergraaf, Alex. “Minimizing the Solaris Operating Environment for Security: Updated for Solaris 9 Operating Environment,” Sun BluePrints OnLine, November 2002,http://sun.com/blueprints/1102/816-5241.pdf.
- Noordergraaf, Alex. “Securing the Sun Cluster 3.x Software,” Sun BluePrints OnLine article, February 2003, http://www.sun.com/solutions/blueprints/0203/817-1079.pdf.
- Noordergraaf, Alex, “Securing the Sun Enterprise 10000 System Service Processors,” Sun BluePrints OnLine article, March 2002, http://www.sun.com/blueprints/0302/securingenter.pdf
- Noordergraaf, Alex, et. al. *Enterprise Security: Solaris Operating Environment Security Journal, Solaris Operating Environment Versions 2.5.1, 2.6, 7, and 8*, Sun Microsystems™, Prentice Hall Press, ISBN 0-13-100092-6, June 2002.
- Noordergraaf, Alex and Nimeh, Dina. “Securing the Sun Fire 12K and 15K Domains,” Sun BluePrints OnLine article, February 2003, http://www.sun.com/blueprints/0203/817-1357.pdf.
- Noordergraaf, Alex and Nimeh, Dina. “Securing the Sun Fire 12K and 15K System Controllers,” Sun BluePrints OnLine article, February 2003, http://www.sun.com/blueprints/0203/817-1358.pdf.
- Noordergraaf, Alex and Watson, Keith. “Solaris Operating Environment Security: Updated for the Solaris 9 Operating Environment,” Sun BluePrints OnLine, December 2002, http://www.sun.com/blueprints/1202/816-5242.pdf.
- O’Donnell, Nicholas and Noordergraaf, Alex. “Minimizing Domains for Sun Fire V1280, 6800, 12K, and 15K Systems," Sun BluePrints OnLine articles, September 2003, http://www.sun.com/blueprints/0903/817-3340.pdf [Part I] and http://www.sun.com/blueprints/0903/817-3628.pdf [Part II]
- Osser, William and Noordergraaf, Alex. “Auditing in the Solaris 8 Operating Environment,” Sun BluePrints OnLine, February 2001 http://www.sun.com/blueprints/0201/audit_config.pdf.
- Reid, Jason M. and Watson, Keith. “Building and Deploying OpenSSH in the Solaris Operating Environment,” Sun BluePrints OnLine, July 2001, http://sun.com/blueprints/0701/openSSH.pdf.
- Reid, Jason M. “Configuring OpenSSH for the Solaris Operating Environment,” Sun BluePrints OnLine article, January 2002, http://www.sun.com/blueprints/0102/configssh.pdf.
- Reid, Jason. *Secure Shell in the Enterprise*, The Official Sun Microsystems Resource Series, Prentice Hall, June 2003
- *Solaris Advanced Installation Guide*, Sun Microsystems, http://docs.sun.com.
- *SunSHIELD Basic Security Module Guide*, Sun Microsystems, Inc., http://docs.sun.com.

- Watson, Keith and Noordergraaf, Alex. "Solaris Operating Environment Network Settings for Security: Updated for Solaris 9 Operating Environment," Sun BluePrints OnLine, June 2003, `http://www.sun.com/solutions/blueprints/0603/816-5240.pdf`.
- Weise, Joel, and Martin, Charles R. "Developing a Security Policy," Sun BluePrints OnLine article, December 2001, `http://www.sun.com/solutions/blueprints/1201/secpolicy.pdf`.

Web Sites

- AUSCERT, *UNIX Security Checklist*, http://www.auscert.org.au/render.html?it=1935&cid=1920
- CERT/CC at `http://www.cert.org` is a federally funded research and development center working with computer security issues.
- Chkrootkit, `http://www.chkrootkit.org`
- Galvin, Peter Baer, *The Solaris Security FAQ*, `http://www.itworld.com/Comp/2377/security-faq/`
- HoneyNet Project, "Know Your Enemy: Motives" `http://project.honeynet.org/papers/motives/`
- List open files software, `ftp://vic.cc.purdue.edu/pub/tools/unix/lsof/`
- Nmap Port Scanner, `http://www.insecure.org`
- OpenSSH tool, `http://www.openssh.com/`
- Pomeranz, Hal, *Solaris Security Step by Step*, `http://www.sans.org/`
- Rhoads, Jason, *Solaris Security Guide*, `http://www.sabernet.net/papers/Solaris.html`
- Security Focus at `http://www.securityfocus.org` is a web site dedicated to discussing pertinent security topics.
- Sendmail Consortium, `sendmail` configuration information, `http://www.sendmail.org/`
- Spitzner, Lance, *Armoring Solaris*, `http://secinf.net/unix_security/Armoring_Solaris.html`
- SSH Communications Security, Secure Shell (SSH) tool, `http://www.ssh.com/`
- Sun BluePrints OnLine, `http://sun.com/blueprints`
- Sun BluePrints OnLine Tools for FixModes software and MD5 scripts, `http://jsecom15k.sun.com/ECom/EComActionServlet?StoreId=8&PartDetailId=817-0074-10&TransactionId=try&LMLoadBalanced=`
- Sun Enterprise Authentication Mechanism™ information, `http://www.sun.com/software/solaris/ds/ds-seam`

- SunSolve[SM] – `http://sunsolve.sun.com`

Contacting Sun Technical Support

If you have technical questions about this product that are not answered in this document, go to:

`http://www.sun.com/service/contacting`

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. You can submit your comments by going to:

`http://www.sun.com/hwdocs/feedback`

Please include the title and part number of your document with your feedback:

Solaris Security Toolkit 4.1 Administration Guide, part number 817-7424-10

CHAPTER 1

Introduction

This chapter describes the design and purpose of the Solaris Security Toolkit software. It covers the key components, features, benefits, and supported platforms. This chapter provides guidelines for maintaining version control of modifications and deployments, and it sets forth important guidelines for customizing the Solaris Security Toolkit software.

This chapter contains the following topics:

Securing Systems With the Solaris Security Toolkit Software

The Solaris Security Toolkit software, informally known as the JASS (JumpStart Architecture and Security Scripts) toolkit, provides an automated, extensible, and scalable mechanism to build and maintain secure Solaris OS systems. Using the Solaris Security Toolkit software, you can harden, minimize, and audit the security of systems.

- **Hardening** – The modification of Solaris OS configurations to improve the security of a system.

- **Minimizing** – The removal of Solaris OS packages that are not needed on a particular system. (Because each system's requirements vary, what is deemed unnecessary also varies and has to be evaluated.) This removal reduces the number of components to be patched and made secure, which in turn reduces entry points available to a possible intruder.
- **Auditing** – The process of determining if a system's configuration is in compliance with a predefined security profile.
- **Scoring** – A score is a value associated with the number of failures uncovered during an audit run. So, if no failures (of any kind) are found, then the resulting score is 0. The Solaris Security Toolkit increments the score (also known as a vulnerability value) by 1 whenever a failure is detected.

System installation and configuration should be as automated as possible (ideally, 100 percent). This guideline includes OS installation and configuration, network configuration, user accounts, applications, and security modifications. The security modification may include hardening and/or minimization, depending on the purpose of the system. One technology available to automate Solaris OS installations is JumpStart software. The JumpStart software provides a mechanism to install systems over a network, with little or no human intervention required. The Solaris Security Toolkit software provides a framework and scripts to implement and automate most of the tasks associated with hardening and minimizing Solaris OS systems in JumpStart software-based installations.

In addition, the Solaris Security Toolkit software has a standalone mode. This mode provides the ability to perform all the same hardening functionality as in JumpStart mode, but on deployed systems. In either mode, the security modifications made can, and should, be customized to match security requirements for your system.

Regardless of how a system is installed, you can use the Solaris Security Toolkit software initially to harden and minimize your systems. Then periodically use the Solaris Security Toolkit software to audit whether the security profile of secured systems is not accidently or maliciously modified.

Note – The term *audit* describes the Solaris Security Toolkit software's automated process of validating a security posture by comparing it with a predefined security profile. The use of this term in this publication does not represent a guarantee that a system is completely secure after using the audit option.

Understanding the Software Components

This section provides an overview of the structure of the Solaris Security Toolkit software components. The Solaris Security Toolkit software is a collection of files and directories. FIGURE 1-1 shows an illustration of the structure.

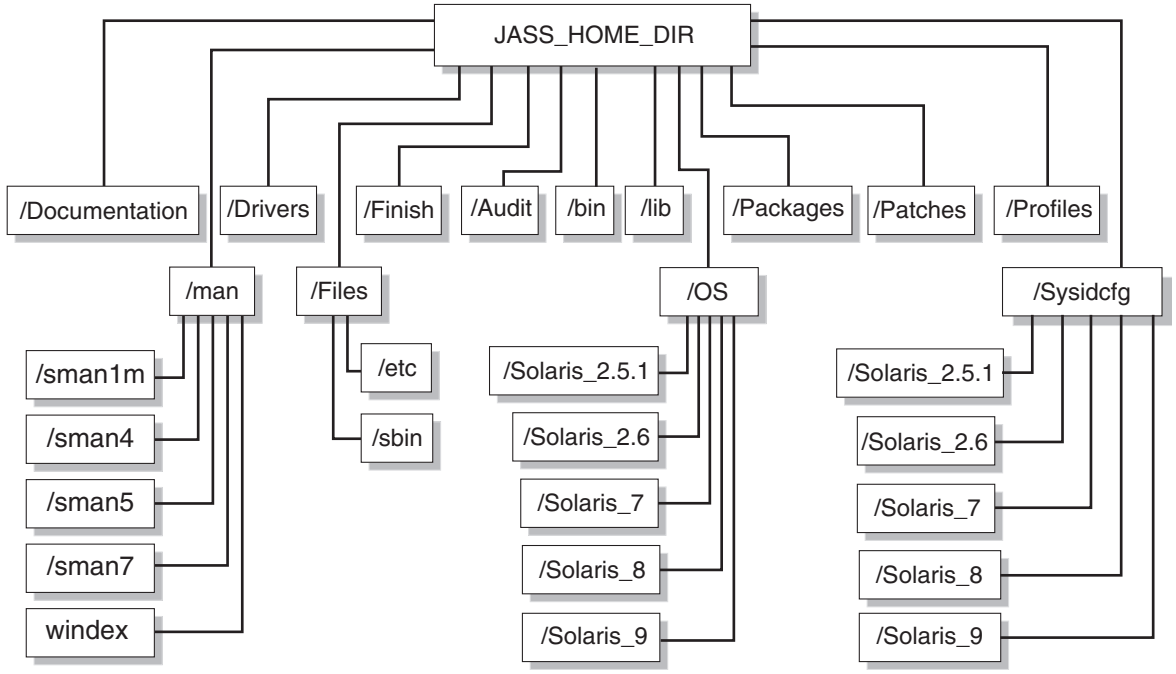

FIGURE 1-1 Software Component Structure

In addition to these directories and subdirectories, the following files are in the top level of the Solaris Security Toolkit software structure, in `/bin`:

- `add-client` – A JumpStart helper program for adding clients into a JumpStart environment.
- `rm-client` – A JumpStart helper program for removing clients from a JumpStart environment.
- `make-jass-pkg` – A command that provides the ability to create a Solaris OS package from the contents of the Solaris Security Toolkit directory, to simplify internal distribution of a customized Solaris Security Toolkit configuration.

- `jass-check-sum` – A command that provides the ability to determine if any files modified by the Solaris Security Toolkit software have been changed, based on a checksum created during each Solaris Security Toolkit run.
- `jass-execute` – A command that configure the Solaris Security Toolkit application.

Directories

The components of the Solaris Security Toolkit architecture are organized in the following directories:

- `/Audit`
- `/bin`
- `/Documentation`
- `/man`
- `/Drivers`
- `/Files`
- `/Finish`
- `/lib`
- `/OS`
- `/Packages`
- `/Patches`
- `/Profiles`
- `/Sysidcfg`

Each directory is described in this section. Where appropriate, each script, configuration file, or subdirectory is listed, and references to other chapters are provided for detailed information.

The Solaris Security Toolkit directory structure is based on the structure in the Sun BluePrints book *JumpStart Technology: Effective Use in the Solaris Operating Environment*.

`Audit` Directory

This directory contains the audit scripts that evaluate a system's compliance with a defined security profile or set of audit scripts. The scripts in this directory are organized into the following categories:

- Disable
- Enable
- Install
- Minimize
- Print
- Remove

- Set
- Update

For detailed listings of the scripts in each of these categories and descriptions of each script, refer to the *Solaris Security Toolkit 4.1 Reference Manual*.

`Documentation` Directory

This directory contains text files with information for the user, such as a `README` file.

`man` Directory

This directory contains subdirectories for the sections of man pages for commands, functions, and drivers. This directory also contains the `windex` file, which is an index of the commands and provided as a courtesy.

For more information about these man pages, refer to the man pages themselves or the *Solaris Security Toolkit 4.1 Man Page Guide*.

`Drivers` Directory

This directory contains files of configuration information specifying which files are executed and installed when you run the Solaris Security Toolkit software. This directory contains drivers, scripts, and configuration files.

The following is an example of the drivers and scripts in the `Drivers` directory:

- `common_{log|misc}.funcs`
- `config.driver`
- `desktop-{config|hardening|secure}.driver`
- `driver.{funcs|init|run}`
- `hardening.driver`
- `finish.init`
- `install-Sun_ONE-WS.driver`
- `jumpstart-{config|hardening|secure}.driver`
- `secure.driver`
- `starfire-{config|hardening|secure}.driver`
- `suncluster3x-{config|hardening|secure}.driver`
- `sunfire_15k_domain-{config|hardening|secure}.driver`
- `sunfire_15k_sc-{config|hardening|secure}.driver`
- `sunfire_mf_msp-{config|hardening|secure}.driver`
- `undo.{funcs|init|run}`
- `hardening.driver`
- `user.init.SAMPLE`
- `user.run.SAMPLE`

- `audit_{private|public}.funcs`

All product-specific drivers and some other drivers include three files for each driver:

- *name*`-secure.driver`
- *name*`-config.driver`
- *name*`-hardening.driver`

These three files are indicated in brackets in the previous list, for example, `sunfire_15k_sc-{config|hardening|secure}.driver`. These files are listed for completeness. Use only the *name*`-secure.driver` when you want to execute a driver. That driver automatically calls the related drivers.

The Solaris Security Toolkit architecture includes configuration information to enable driver, finish, and audit scripts to be used in different environments, while not modifying the actual scripts themselves. All variables used in the finish and audit scripts are maintained in a set of configuration files—these configuration files are imported by drivers, which make the variables available to the finish and audit scripts as they are called by the drivers.

The Solaris Security Toolkit software has three main configuration files, all of which are stored in the `Drivers` directory:

- `driver.init`
- `finish.init`
- `user.init`

Finish scripts called by the drivers are located in the Finish directory. Audit scripts called by the drivers are located in the Audit directory. Files installed by the drivers are read from the Files directory. For more information about finish and audit scripts, see the respective chapters in this book.

FIGURE 1-2 shows a flow chart of the driver control flow.

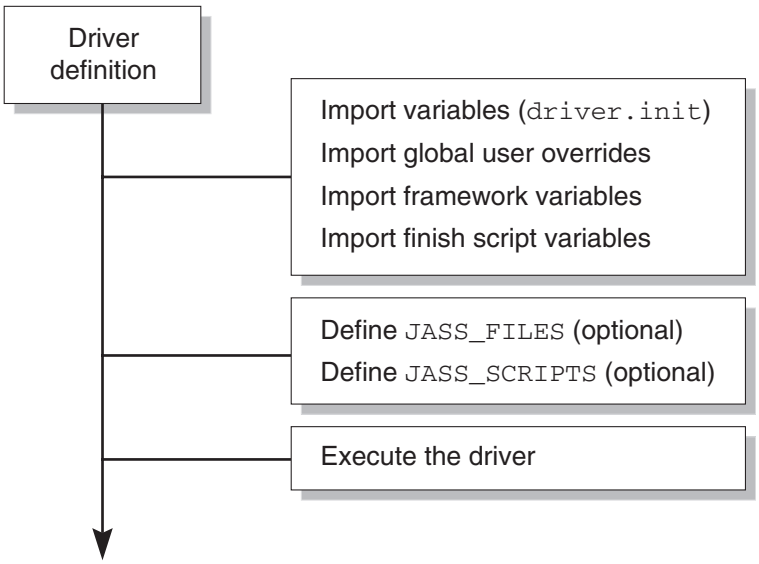

FIGURE 1-2 Driver Control Flow

All of the environment variables from the various `.init` files are imported first. Once this is complete, the driver moves on to part two, which defines `JASS_FILES` and `JASS_SCRIPTS`. The definition of these are optional; either a single environment can be defined, or both, or none. Part three of the driver calls `driver.run` to perform the tasks defined by the `JASS_FILE` and `JASS_SCRIPTS` environment variables.

CODE EXAMPLE 1-1 demonstrates driver control flow.

CODE EXAMPLE 1-1 Driver Control Flow

```
DIR="`/bin/dirname $0`"

export DIR
. ${DIR}/driver.init

JASS_FILES="
                            /etc/cron.d/cron.allow
                            /etc/default/ftpd
                            /etc/default/telnetd
"

JASS_SCRIPTS="
                            install-at-allow.fin
                            remove-unneeded-accounts.fin
"
. ${DIR}/driver.run
```

This code example sets and exports the DIR environment variable so that the drivers recognize the starting directory. Next, the JASS_FILES environment variable is defined as containing those files that are copied from the JASS_HOME_DIR/Files directory onto the client. The JASS_SCRIPTS environment variable is then defined with the finish scripts that are run by the Solaris Security Toolkit software. Finally, the execution of the hardening run is started by calling the driver.run driver. Once called, driver.run copies the files specified by JASS_FILES, and runs the scripts specified by JASS_SCRIPTS.

Files Directory

This directory is used by the JASS_FILES environment variable and the driver.run script. This directory stores files that are copied to the JumpStart client.

The following files are in this directory:

- /.cshrc
- /.profile
- /etc/default/sendmail
- /etc/dt/config/Xaccess
- /etc/hosts.{allow|deny}
- /etc/init.d/nddconfig
- /etc/init.d/set-tmp-permissions
- /etc/init.d/sms_arpconfig
- /etc/init.d/swapadd
- /etc/issue
- /etc/motd
- /etc/notrouter
- /etc/rc2.d/S00set-tmp-permissions
- /etc/rc2.d/S07set-tmp-permissions
- /etc/rc2.d/S70nddconfig
- /etc/rc2.d/S73sms_arpconfig
- /etc/rc2.d/S73swapadd
- /etc/security/audit_class
- /etc/security/audit_control
- /etc/security/audit_event
- /etc/sms_domain_arp
- /etc/sms_sc_arp
- /etc/syslog.conf

Finish Directory

This directory contains the finish scripts that perform system modifications and updates during installation. The scripts in this directory are organized into the following categories:

- Disable
- Enable
- Install
- Minimize
- Print
- Remove
- Set
- Update

For detailed listings of the scripts in each of these categories and descriptions of each script, refer to the *Solaris Security Toolkit 4.1 Reference Manual*.

`OS` Directory

This directory contains only Solaris OS images. These are used by the JumpStart software installation process as the source of the client installation and to provide the `add_install_client` and `rm_install_client` scripts. The `add_client` script accepts these additional directory names.

For more information about loading and modifying Solaris OS images, refer to the Sun BluePrints book *JumpStart Technology: Effective Use in the Solaris Operating Environment*.

The standard installation naming conventions follow.

Solaris OS

Use the following naming standard for Solaris OS:

`Solaris_`*`os version_4 digit year_2 digit month of CD release`*

For example, the Solaris 8 Operating Environment CD, dated April 2001, would have a directory name of `Solaris_8_2001-04`. By separating updates and releases of the Solaris OS, very fine control can be maintained for testing and deployment purposes.

Trusted Solaris OS

Use the following directory naming standard for Trusted Solaris:

`Trusted_Solaris_`*`os version_4 digit year_2 digit month of CD release`*

For example, if the Trusted Solaris software release were dated February 2000, the directory name would be: `Trusted_Solaris_8_2000-02`.

Solaris OS Intel Platform Edition

Use the following directory naming for Solaris OS Intel Platform Edition:

`Solaris_`*os version_4 digit year_2 digit month of CD release*`_ia`

For example, if the Solaris OS Intel Platform Edition release were dated April 2001, the directory name would be: `Solaris_8_2001-04_ia`.

`Packages` Directory

This directory contains software packages that can be installed with a finish script. For example, the Sun Java™ System Web Server, formerly the Sun™ ONE Web Server, and before that the iPlanet™ Web Server, software package could be stored in the `Packages` directory so that the appropriate finish script installs the software as required.

Several finish scripts included in the Solaris Security Toolkit software perform software installation and basic configuration functions. The scripts that install software from the `Packages` directory include:

- `install-fix-modes.fin`
- `install-Sun_ONE-WS.fin`
- `install-jass.fin`
- `install-md5.fin`
- `install-openssh.fin`

`Patches` Directory

This directory is for storing Recommended and Security Patch Clusters for the Solaris OS. Download and extract required patches into this directory.

By placing and extracting the patches in this directory, you streamline installation. When the patches are extracted into this directory, the Solaris Security Toolkit software's patch installation script automates installation so that you do not have to manually extract the patch clusters for each system installation.

Create subdirectories for each of the Solaris OS versions used. For example, you might have directories `2.5.1_Recommended` and `2.6_Recommended` within the `Patches` directory.

Solaris Security Toolkit software supports Solaris OS Intel Platform Edition patch clusters. The supported naming convention for these patch clusters is the same as made available through SunSolve OnLine℠ service.

The format is `Solaris_<release>_x86_Recommended`. The Solaris OS Intel Platform Edition patch cluster for Solaris 8 OS would be in a directory named `Solaris_8_x86_Recommended`.

`Profiles` Directory

This directory contains all JumpStart profiles. These profiles contain configuration information used by JumpStart software to determine Solaris OS clusters for installation (for example, Core, End User, Developer, or Entire Distribution), disk layout, and installation type (for example, standalone) to perform.

JumpStart profiles are listed and used in the `rules` file to define how specific systems or groups of systems are built.

`Sysidcfg` Directory

Similar to the `Profiles` directory, the `Sysidcfg` directory contains files that are only used during JumpStart mode installations. These files automate Solaris OS installations by providing the required installation information. A separate directory tree stores OS-specific information.

Each Solaris OS has a separate directory. For each release, there is a directory named `Solaris_`*OS Version*. The Solaris Security Toolkit software includes sample `sysidcfg` files for Solaris OS versions 2.5.1 through 9.

The sample `sysidcfg` files can be extended to other types such as per network, host, etc. The Solaris Security Toolkit software supports arbitrary `sysidcfg` files.

For additional information on `sysidcfg` files, refer to the Sun BluePrints book *JumpStart Technology: Effective Use in the Solaris Operating Environment*.

Data Repository

The data repository is an environment variable in the `JASS_REPOSITORY` directory that supports Solaris Security Toolkit undo runs, saves data on how each run is executed, maintains a manifest of files modified by the software, and saves data for the execution log. The undo feature relies on the information stored in the data repository.

Maintaining Version Control

Maintaining version control for all files and scripts used by the Solaris Security Toolkit software is critical for two reasons. First, one of the goals of this environment is to be able to recreate a system installation. This goal would be impossible without a snapshot of all file versions used during an installation. Second, because these scripts are performing security functions—which is a critical process for many organizations—extreme caution must be exercised to ensure that only appropriate and tested changes are implemented.

A Source Code Control System (SCCS) version control package is provided in the Solaris OS `SUNWsprot` package. You can use other version control software available from freeware and commercial vendors to manage version information. Whichever version control product you use, put a process in place to manage updates and capture version information for future system re-creation.

Use an integrity management solution in addition to version control to determine whether the contents of files were modified. Although privileged users of a system may be able to bypass the version control system, they would not be able easily to bypass an integrity management system, which maintains its integrity database on a remote system. Integrity management solutions work best when centralized, because locally-stored databases could be maliciously modified.

Running Supported Solaris OS Versions

Sun support for Solaris Security Toolkit software is available only for its use in the Solaris 8 and Solaris 9 Operating Systems. While the software can be used in the Solaris 2.5.1, Solaris 2.6, and Solaris 7 Operating Systems, Sun support is not available for its use in those operating systems.

The Solaris Security Toolkit software automatically detects which version of the Solaris Operating System software is installed, then runs tasks appropriate for that operating system version.

Running Supported SMS Versions

If you are using System Management Services (SMS) to manage your system controller (SC), Sun support is available for Solaris Security Toolkit 4.1 software if you are using SMS versions 1.3 through 1.4.1.

Configuring and Customizing the Solaris Security Toolkit Software

The Solaris Security Toolkit software contains default values for scripts, framework functions, and variables that implement all security guidelines in the Sun BluePrints book titled *Enterprise Security: Solaris Operating Environment Security Journal, Solaris Operating Environment Versions 2.5.1, 2.6, 7, and 8* and Sun BluePrints OnLine articles about security. These settings are not appropriate for all systems, so you must customize the Solaris Security Toolkit software to meet the security requirements for your systems.

One of the most significant characteristics of the Solaris Security Toolkit software is that you can easily customize it to fit your environment, systems, and requirements. To customize the Solaris Security Toolkit software, adjust its actions through drivers, finish scripts, audit scripts, framework functions, environment variables, and file templates.

Most users do not need to modify the Solaris Security Toolkit code. Any changes may negatively impact supportability and upgrades. If code modifications are absolutely necessary for using the Solaris Security Toolkit software in your environment, copy the code to a unique file or function name, so that you can easily track changes as in "Guidelines" on page 14.

Throughout this guide, guidelines and instructions for customizing the Solaris Security Toolkit software are provided where applicable in each chapter. Refer to the the *Solaris Security Toolkit 4.1 Reference Manual* to find information helpful about customizing the drivers. Customizing includes modifying and creating files or variables.

The following chapters provide examples for customizing the Solaris Security Toolkit software. The examples highlight some ways that you can customize the Solaris Security Toolkit software; however, there are many possibilities.

The following sections present information that must be clearly understood before attempting to customize the Solaris Security Toolkit software. The information is based on shared experiences collected from many deployments, so that you can avoid common pitfalls.

Policies and Requirements

When customizing and deploying the Solaris Security Toolkit software, proper planning ensures that the resulting platform configuration is correct and in line with your organization's expectations.

In your planning phase, be sure to obtain input from a variety of sources, including security policies and standards, industry regulations and guidelines, and vendor-supplied preferred practices.

In addition to this information, it is essential that you consider application and operational requirements to ensure that the resulting configuration does not impact a platform's ability to serve its intended business function.

Guidelines

When customizing the Solaris Security Toolkit software, consider the following guidelines. Understanding and observing these guidelines help make the process of sustaining a deployment much simpler and more effective.

- As a general rule, never alter any of the *original* files (drivers, scripts, files, and so on) provided with the Solaris Security Toolkit software. Changing the original files inhibits and restricts your organization's ability to upgrade to newer versions of the Solaris Security Toolkit software, because any changes to the original files may be overwritten by new versions of the files. (All of your custom changes would be lost, and your system's configuration might change in undesirable ways.) To customize any of the files, first make a copy, then modify the copy, leaving the original intact. Only three exceptions exist to this guideline, the `sysidcfg` files, templates in the Files directory, and when instructed by Sun BluePrints OnLine articles.

- Name your copy of a driver or script appropriately, so that it can be distinguished from the original. Use a prefix or keyword that is indicative of the purpose of the script. For example, a prefix that contains the name or stock symbol of the company, a department identifier, or even a platform or application type are all excellent naming standards. TABLE 1-1 lists a few examples of naming standards.

TABLE 1-1 Naming Standards for Custom Files

Custom File	Naming Standard
`abccorp-secure.driver`	Company prefix
`abcc-nj-secure.driver`	Company stock symbol, location
`abbcorp-nj-webserver.driver`	Company, location, application type
`abc-nj-trading-webserver.driver`	Company, location, organization, application type

- Review the following Solaris Security Toolkit files for suitability to your system. To customize these files, copy the original files, rename the copies to `user.init` and `user.run`, then modify or add content to the copies.

`Drivers/user.init.SAMPLE`	Used for customizing global parameters.
`Drivers/user.run.SAMPLE`	Used for customizing global functions.

- If necessary, modify the following original files. These files are the *only* original Solaris Security Toolkit files that you should ever modify directly.

`Sysidcfg/*/sysidcfg`	Used for JumpStart automatic configuration.
`Files/*`	Used as file templates and copied to systems.

Note – Be aware that if `SUNWjass` is removed using the `pkgrm` command, the `user.init` and `user.run` files, if created, are not removed. This behavior occurs for any customer files that are added to the Solaris Security Toolkit directory structure and are not included in the distribution. The files in the Files directory included in the Solaris Security Toolkit distribution and `sysidcfg` files exist, and would therefore be removed.

CHAPTER 2

Securing Systems: Applying a Methodology

This chapter provides a methodology for securing systems. It provides a process that you can apply before securing your systems using the Solaris Security Toolkit software.

This chapter contains the following topics:

Planning and Preparing

Proper planning is key to successfully using the Solaris Security Toolkit software to secure systems. The planning phase constructs a Solaris Security Toolkit profile for the system, based on an organization's security policies and standards, as well as the application and operation requirements of the system. This phase is divided into the following tasks:

Although not covered in this book, other considerations for this phase might include understanding risks and exposures, understanding infrastructure and its security requirements, and considering accountability, logging, and usage auditing.

Considering Risks and Benefits

This section presents considerations that must be clearly understood before you attempt to secure a system. Carefully weigh the risks with the benefits to determine which actions are appropriate for your organization.

When hardening systems, special precautions must be taken to help ensure that the system is functional after the Solaris Security Toolkit software is implemented. Further, it is important that the process be optimized to ensure any downtime is as minimal as possible.

Note – When securing a deployed system, it may be more effective in some cases for an organization to rebuild the system, hardening it at installation time, then reload all of the software necessary for operation.

---

1. Understand the requirements of the services and applications on the system.

   You must identify the services and applications running on a system prior to running the Solaris Security Toolkit software. Any dependencies associated with the services and applications must be enumerated so that the configuration of the Solaris Security Toolkit software can be sufficiently adjusted. Failure to do so could disable or prevent necessary services from starting. While the changes made by the Solaris Security Toolkit software can in most cases be undone, developing a correct profile before installation, limits the potential downtime associated with the Solaris Security Toolkit software implementation.

2. Take into account that the system has to be taken offline and rebooted.

   For the changes made by the Solaris Security Toolkit software to take effect, the system must be rebooted. Depending on how vital the system is, the services that it provides, and the availability of a maintenance window, an organization may face difficulties implementing the software. A decision must be made after carefully weighing the cost of downtime versus the risks of not enhancing security.

3. A system might require multiple reboots to verify functionality.

   Make all changes on nonproduction systems prior to implementing the systems in a mission-critical setting. This is not always possible; for example, due to lack of sufficient hardware or software that effectively mirrors the target environment. Testing must be conducted both before and after the installation of the Solaris Security Toolkit software. There could still be unidentified dependencies that require troubleshooting after a system is hardened. In most cases, these issues can be resolved fairly quickly using the techniques described in this chapter. If functionality problems are discovered after the Solaris Security Toolkit software installation, additional platform reboots might be necessary to either undo the

effects of the Solaris Security Toolkit software or to make further changes to the security configuration of the system to support and enable the missing functionality.

4. Platform security entails more than just hardening and minimizing.

   When considering retrofitting a system's configuration to enhance its security posture, it is critical to understand that platform hardening and minimizing represent only a fraction of what can and should be done to protect a system, services, and data. A treatment of the additional measures and controls is outside the scope of this document, but you are encouraged to consider issues related to account management, privilege management, file system and data integrity, host-based access control, intrusion detection, vulnerability scanning and analysis, and application security.

5. The system might already have been exploited or has exploitable vulnerabilities.

   The platform being hardened might have already been exploited by an attacker. The Solaris Security Toolkit software is probably being implemented too late to offer protection for an exploited vulnerability. In this case, reinstall the system, then install and use the Solaris Security Toolkit software to enhance security.

Reviewing Security Policy, Standards, and Related Documentation

The first task in securing a system is to understand your organization's relevant security policies, standards, and guidelines with respect to platform security. Use these documents to form your foundation of the Solaris Security Toolkit's profile, because these documents communicate requirements and practices to be followed for all systems in your organization. If your organization does not have documentation, developing it increases your ability to customize the Solaris Security Toolkit software.

Note – When looking for these documents, keep in mind that some material might be listed in best practices or other documentation.

For more information on security policies, refer to the Sun BluePrints OnLine article "Developing a Security Policy." This document can be used to gain a greater understanding of the role that security policies play in an organization's security plan.

The following two examples illustrate how policy statements can directly impact the way that the Solaris Security Toolkit's profile is configured.

Example 1

- **Policy** – An organization must use management protocols that support strong authentication of users and encryption of transmitted data.
- **Profile Impact** – Clear-text protocols such as Telnet, FTP, SNMPv1, and others should not be used. By default, the Solaris Security Toolkit disables such services, so no additional configuration is needed.

Note – Both Telnet and FTP services can be configured to support stronger authentication and encryption using extensions such as Kerberos. These services are listed as examples, however, because their default configurations do not support these added levels of security.

Example 2

Policy – All users are forced to change their passwords every 30 days.

Profile Impact – The Solaris Security Toolkit software can be configured to enable password aging. By default, the Solaris Security Toolkit software sets a password maximum age to 8 weeks (56 days). To comply with the policy, the Solaris Security Toolkit software's profile must be changed. Refer to the *Solaris Security Toolkit 4.1 Reference Manual*.

Although the Solaris Security Toolkit software enables password aging by default when run on a system, this change does not affect existing users. To enable password aging for existing users, invoke the `passwd`(1) command on each user account.

Determining Application and Service Requirements

This task ensures that services remain functional after a system is hardened. This task is comprised of the following steps:

- "Identifying Application and Operational Service Inventory" on page 20
- "Determining Service Requirements" on page 21

Identifying Application and Operational Service Inventory

Inventory the applications, services, and operational or management functions. This inventory is necessary to determine the software that is actually being used on a system. In many cases, systems are configured with more software than is used and with software that does not support business functions.

Systems should be constructed minimally whenever possible. That is, software that is not required to support a business function should not be installed. Unnecessary software applications on a system increase the number of opportunities that an attacker can use to exploit the system. Additionally, more software on a system usually equates to more patches that must be applied. For information on minimizing the Solaris OS, refer to the Sun BluePrints OnLine article "Minimizing the Solaris Operating Environment for Security."

When building the inventory of software, be sure to include infrastructure components such as management, monitoring, and backup software in addition to applications residing on the system.

Determining Service Requirements

After you complete an application and service inventory, determine if any components have dependencies that could be impacted by the hardening process. Many third-party applications do not directly use services provided by the Solaris OS. For those applications that do, the following sections provide helpful information.

- "Shared Libraries" on page 21
- "Configuration Files" on page 24
- "Service Frameworks" on page 25

Shared Libraries

It is important to understand which libraries are needed to support an application. This knowledge is most useful in debugging circumstances, but also is useful in preparing a system to be hardened. When the state of a system is unknown, gather as much information as possible so that issues such as software dependencies are clearly understood.

Three methods can be used to determine which libraries are used by an application, depending upon the Solaris OS version you install:

- The first is used against a file system object (for example, application binary).
- The second is used when analyzing a running application.
- The third is used to trace a program when it is started.

Example: Determine the libraries that are needed to support the DNS server software.

To collect information about a file system object, use the `/usr/bin/ldd` command.

CODE EXAMPLE 2-1 Obtaining Information About File System Objects

```
# ldd /usr/sbin/in.named
libresolv.so.2 => /usr/lib/libresolv.so.2
libsocket.so.1 => /usr/lib/libsocket.so.1
libnsl.so.1 =>    /usr/lib/libnsl.so.1
libc.so.1 =>      /usr/lib/libc.so.1
libdl.so.1 =>    /usr/lib/libdl.so.1
libmp.so.2 =>    /usr/lib/libmp.so.2
/usr/platform/SUNW,Ultra-5_10/lib/libc_psr.so.1
```

To collect the information from a running process, use the `/usr/proc/bin/pldd` command (available on Solaris OS versions 8 and 9).

CODE EXAMPLE 2-2 Collecting Information From a Running Process

```
# pldd 20307
20307:  /usr/sbin/in.named
/usr/lib/libresolv.so.2
/usr/lib/libsocket.so.1
/usr/lib/libnsl.so.1
/usr/lib/libc.so.1
/usr/lib/libdl.so.1
/usr/lib/libmp.so.2
/usr/platform/sun4u/lib/libc_psr.so.1
/usr/lib/dns/dnssafe.so.1
/usr/lib/dns/cylink.so.1
```

The `pldd` command shows the shared libraries that are loaded dynamically by the application, in addition to those against which the application is linked. This information can also be gathered using the following `truss` command.

Note – The following output is truncated for brevity.

CODE EXAMPLE 2-3 Identifying Dynamically Loaded Applications

```
# truss -f -topen,open64 /usr/sbin/in.named
20357:  open("/usr/lib/libresolv.so.2", O_RDONLY)       = 3
20357:  open("/usr/lib/libsocket.so.1", O_RDONLY)       = 3
20357:  open("/usr/lib/libnsl.so.1", O_RDONLY)          = 3
20357:  open("/usr/lib/libc.so.1", O_RDONLY)            = 3
20357:  open("/usr/lib/libdl.so.1", O_RDONLY)           = 3
20357:  open("/usr/lib/libmp.so.2", O_RDONLY)           = 3
20357:  open("/usr/lib/nss_files.so.1", O_RDONLY)       = 4
20357:  open("/usr/lib/nss_files.so.1", O_RDONLY)       = 4
20357:  open("/usr/lib/dns/dnssafe.so.1", O_RDONLY)     = 4
20357:  open("/usr/lib/dns/cylink.so.1", O_RDONLY)      = 4
20357:  open("/usr/lib/dns/sparcv9/cylink.so.1", O_RDONLY) = 4
```

This version of the output contains the process identifier, the system call (in this case `open`) and its arguments, as well as the system call's return value. Using the return value, it is clear when the system call is successful in finding and opening the shared library.

Once the list of shared libraries are well known, use the following command to determine the Solaris OS packages to which they belong. .

```
# grep '/usr/lib/dns/cylink.so.1' /var/sadm/install/contents
/usr/lib/dns/cylink.so.1 f none 0755 root bin 63532 24346 \
  1018126408 SUNWcsl
```

The resulting output, indicates that this shared library belongs to the `SUNWcsl` (Core, Shared Libs) package. This process is especially useful when performing platform minimization, because this process helps to identify the packages that are required to support an application or service.

Configuration Files

Another way to gather requirements is through configuration files. This process has a more direct impact on how a system is hardened, because configuration files can be renamed or removed to disable services. For more information, refer to the *Solaris Security Toolkit 4.1 Reference Manual*.

To determine if a configuration file is in use, use the `truss` command.

Note – The following output is truncated for brevity.

CODE EXAMPLE 2-4 Determining if a Configuration File Is In Use

```
# truss -f -topen,open64 /usr/sbin/in.named 2>&1 | \
grep -v "/usr/lib/.*.so.*"
20384:  open("/etc/resolv.conf", O_RDONLY)              = 3
20384:  open("/dev/conslog", O_WRONLY)                  = 3
20384:  open("/usr/share/lib/zoneinfo/US/Eastern", O_RDONLY) = 4
20384:  open("/var/run/syslog_door", O_RDONLY)          = 4
20384:  open("/etc/nsswitch.conf", O_RDONLY)            = 4
20384:  open("/etc/services", O_RDONLY)                 = 4
20384:  open("/etc/protocols", O_RDONLY)                = 4
20384:  open("/etc/named.conf", O_RDONLY)               = 4
20384:  open("named.ca", O_RDONLY)                      = 5
20384:  open("named.local", O_RDONLY)                   = 5
20384:  open("db.192.168.1", O_RDONLY)                  = 5
20384:  open("db.internal.net", O_RDONLY)               = 5
```

In this example, the DNS service uses configuration files such as `/etc/named.conf`. As with the previous example, if the return value of a service indicates an error, there might be a problem. Carefully documenting the results both before and after hardening can help to speed the entire validation process.

Service Frameworks

This category includes frameworks or meta-services on which larger, more complex applications are built. The types of frameworks typically found in this category are naming services (for example, NIS, NIS+, and LDAP), authentication services (for example, Kerberos and LDAP), and services such as port mapper used by the RPC facility.

It is not always clear when an application depends on these types of services. When special adjustments are needed to configure an application, such as adding it to a Kerberos realm, the dependency is known. In some cases, application dependencies do not require any added tasks and the actual dependency might not be documented by the vendor.

One such example is the RPC port mapper. By default, the Solaris Security Toolkit software disables the RPC port mapper. This action may cause unexpected behavior in other services relying on this service. Based on past experiences, services abort, hang, or fail depending on how well the application's code is written to handle exception cases. To determine if an application is using the RPC port mapper, use the `rpcinfo` command. For example:

CODE EXAMPLE 2-5 Determining Which Applications Use RPC

```
# rpcinfo -p
100000    3   tcp    111  rpcbind
100000    4   udp    111  rpcbind
100000    2   udp    111  rpcbind
100024    1   udp  32777  status
100024    1   tcp  32772  status
100133    1   udp  32777
100133    1   tcp  32772
100021    1   udp   4045  nlockmgr
100021    2   udp   4045  nlockmgr
100021    3   udp   4045  nlockmgr
100021    4   udp   4045  nlockmgr
100021    1   tcp   4045  nlockmgr
```

The service column is populated with information from the `/etc/rpc` file and/or a configured naming service such as LDAP.

If this file does not have an entry for a service, as is often the case for third-party products, the service field might be empty. This makes it more difficult to identify applications registered by other applications.

For example, consider the `rusers` command. This command relies on the RPC port mapping service. If the RPC port mapper is not running, the `rusers` command appears to hang. The program eventually times out with the following error message:

```
# rusers -a localhost
localhost: RPC: Rpcbind failure
```

This problem occurs because the program cannot communicate with the service. After starting the RPC port mapping service from `/etc/init.d/rpc`, however, the program immediately yields its result.

As another example, consider the case where the RPC port mapping service is running, and the `rusers` service is not configured to run. In this case, a completely different response is generated, and it is relatively straightforward to validate.

CODE EXAMPLE 2-6 Validating `rusers` Service

```
# rusers -a localhost
localhost: RPC: Program not registered
# grep rusers /etc/rpc
rusersd         100002  rusers
# rpcinfo -p | grep rusers
<No output generated>
```

Given that the `rpcinfo` command does not have a registry for the `rusers` service, it is safe to assume that the service is not configured to run. This assumption is validated by looking at the service entry in the `/etc/inet/inetd.conf`.

```
# grep rusers /etc/inet/inetd.conf
# rusersd/2-3   tli     rpc/datagram_v,circuit_v     wait root
/usr/lib/netsvc/rusers/rpc.rusersd     rpc.rusersd
```

The comment mark (#) at the beginning of the service line indicates that the `rusers` service is disabled. To enable the service, uncomment the line and send a `SIGHUP` signal to the `/usr/sbin/inetd` process as follows.

```
# pkill -HUP inetd
```

Note – The `pkill` command is only available in Solaris OS versions 7 through 9. For other versions, use the `ps` and `kill` commands respectively to find and signal the process.

Another way to determine if an application uses the RPC facility is to use the `ldd` command described earlier.

CODE EXAMPLE 2-7 Alternative Method for Determining Applications That Use RPC

```
# ldd /usr/lib/netsvc/rusers/rpc.rusersd
libnsl.so.1 =>   /usr/lib/libnsl.so.1
librpcsvc.so.1 =>        /usr/lib/librpcsvc.so.1
libc.so.1 =>     /usr/lib/libc.so.1
libdl.so.1 =>    /usr/lib/libdl.so.1
libmp.so.2 =>    /usr/lib/libmp.so.2
/usr/platform/SUNW,Ultra-250/lib/libc_psr.so.1
```

The entry for `librpcsvc.so.1` indicates, along with the file name, that this service relies on the RPC port mapping service.

In addition to the RPC port mapper, applications might rely on other common OS services such as FTP, SNMP, or NFS. You can use similar techniques to debug these services and to determine if they are actually needed to support a business function. One method involves using the `netstat` command as follows.

```
# netstat -a | egrep "ESTABLISHED|TIME_WAIT"
```

This command returns a list of services that are or were recently in use, for example:

TABLE 2-1 Listing Services Recently in Use

```
localhost.32827      localhost.32828      49152      0 49152      0
ESTABLISHED

localhost.35044      localhost.32784      49152      0 49152      0
ESTABLISHED

localhost.32784      localhost.35044      49152      0 49152      0
ESTABLISHED
```

TABLE 2-1 Listing Services Recently in Use *(Continued)*

```
localhost.35047      localhost.35046      49152      0 49152      0
ESTABLISHED

localhost.35046      localhost.35047      49152      0 49152      0
ESTABLISHED

filefly.ssh 192.168.0.3.2969    17615     1 50320     0 ESTABLISHED
```

In this example, many services are in use, but it is unclear which ports are owned by which services or applications. This information can be collected by inspecting the processes using the `pfiles` command (available on Solaris OS versions 8 and 9).

CODE EXAMPLE 2-8 Determining Which Ports Are Owned by Services or Applications

```
# for pid in `ps -aeo pid | grep -v PID`; do
> pfiles ${pid} | egrep "^${pid}:|sockname:"
> done
```

A more effective and efficient way to determine these dependencies is by using the `lsof` (list open files) command. This command determines which processes are using which files and ports. For example, to determine which processes are using port 35047 from the previous example, use the following command.

CODE EXAMPLE 2-9 Determining Which Processes Are Using Files and Ports

```
# ./lsof -i | grep 35047
ttsession   600 root 9u  IPv4 0x3000b4d47e8     0t1  TCP
localhost:35047->localhost:35046 (ESTABLISHED)

dtexec     5614 root 9u  IPv4 0x3000b4d59e8     0t0  TCP
localhost:35046->localhost:35047 (ESTABLISHED)
```

The output of `lsof` indicates that port 35047 is in use for communication between the `dtexec` and `ttsession` processes.

Using the `lsof` program, you might be able to more rapidly determine intersystem or interapplication dependencies that require file system or network usage. Nearly everything that is addressed in this section can be captured using various options of the `lsof` program.

To obtain the `lsof` program, download it from:

`ftp://vic.cc.purdue.edu/pub/tools/unix/lsof/`

Note – The methods described for determining dependencies might not find those items that are rarely used. In addition to using these methods, review documentation and vendor documentation.

Developing and Implementing a Solaris Security Toolkit Profile

After you complete the planning and preparation phase, develop and implement a security profile. A security profile consists of a hardening driver, for example, *name*-`hardening.driver`, and all related drivers, scripts, and files to implement your site-specific security policies.

Customize one of the security profiles provided with the Solaris Security Toolkit software, or develop your own. Each organization's policies, standards, and application requirements differ, even if only slightly.

To customize a security profile, adjust its actions through finish scripts, audit scripts, environment variables, framework functions, and file templates.

See the following chapters for more information:

- For important guidelines about customizing the software, see Chapter 1, "Configuring and Customizing the Solaris Security Toolkit Software" on page 13.
- For an example scenario where a security profile is created, see Chapter 7, "Creating a Security Profile" on page 101.
- For information about customizing drivers, refer to the *Solaris Security Toolkit 4.1 Reference Manual*.

  As needed, see the other applicable chapters of the *Solaris Security Toolkit 4.1 Reference Manual* for information about scripts, framework functions, environment variables, and files. Two key environment variables you might want to customize are `JASS_FILES` and `JASS_SCRIPTS`.

To enforce standards across a majority of platforms while still providing for platform-specific differences, use a technique known as nested or hierarchical security profiles. For more information, refer to the *Solaris Security Toolkit 4.1 Reference Manual*. Compare the resulting security profile with the policies, standards, and requirements of your organization to ensure that changes are not inadvertently or erroneously made.

Installing the Software

The installation of the Solaris Security Toolkit software is the same for both deployed and new systems that are being installed. For detailed instructions, see Chapter 3.

For deployed systems, a few special cases can make this process simpler and faster. These cases are not focused on the hardening process, but are focused on preinstallation and post-installation tasks.

Performing Preinstallation Tasks

Before hardening a deployed system, consider and plan two significant tasks—back up and verification. These tasks help to determine the state of the deployed system and to work out any potential configuration problems before the system is hardened.

Backing Up Data

This task focuses on contingency planning. In the event of a problem, it is necessary to ensure that the system's configuration and data are archived in some form. You must back up the system, ensure that the backup media can be read, and validate that its contents can be restored. Take this step before making any significant change to a system's configuration.

Verifying System Stability

The verification task is nearly as important as the backup task. Verification ensures that the system is in a stable and working state prior to the implementation of any configuration changes, such as those made by the hardening process. This verification process involves a reboot followed by the successful testing of any applications or services. While having a well-defined test and acceptance plan is preferred, documentation might not always be available. If that is the case, test the system in a reasonable way based on how it is used. The goal of this effort is to ensure that the running configuration, in fact, matches the saved configuration.

Investigate any error messages or warnings that are displayed when the system boots or an application starts. If you cannot correct the errors, log them so that during the hardening process they are not included as potential causes of problems. When looking at the log files, be sure to include system, service, and application logs such as:

- `/var/adm/messages`

- `/var/adm/sulog`
- `/var/log/syslog`
- `/var/cron/log`

This task is complete when you can restart the system without encountering errors or warning messages, or without encountering any unknown errors or warnings (all known ones are documented). The system should restart to a known and stable state. If, during the course of verification, you discover that the running and stored configurations of the system differ, reassess your organization's change control policies and processes to identify the gap that leads to that condition.

Performing Post-Installation Tasks

The post-installation task is an extension of the preinstallation tasks. The goal is to ensure that the hardening process did not cause any new faults to the system or applications. This task is primarily conducted by reviewing system and application log files. The log files created after hardening and the subsequent reboot should be similar to those collected before the system was hardened. In some cases, there might be fewer messages, because there are fewer services started. Most importantly, there should be no new error or warning messages.

In addition to reviewing log files, test the functionality, because some applications might fail without generating a log entry. See the following section for detailed verification information.

Verifying Application and Service Functionality

The final task in the process of securing a system involves verifying that the applications and services offered by the system are functioning correctly. This task also verifies that the security profile successfully implemented the requirements of the security policies. Perform this task thoroughly and soon after the reboot of the hardened platform, to ensure that any anomalies or problems are detected and corrected immediately. This task is divided into two subtasks: Verifying security profile installation and verifying application and service functionality.

Verifying Security Profile Installation

To verify that the Solaris Security Toolkit software installed the security profile correctly and without error, review the installation log file. This file is installed in `JASS_REPOSITORY/jass-install-log.txt`.

Note – Refer to this log file to understand what the Solaris Security Toolkit software did to a system. For each run on a system, there is a new log file stored in a directory based on the start time of the run.

In addition to verifying that the profile is installed, assess the security configuration of the system. Perform a manual inspection or use a tool to automate the process.

Verifying Application and Service Functionality

To verify process applications and services, execute a well-defined test and acceptance plan. This plan exercises the various components of a system or application to determine that they are in an available and working order. If such a plan is not available, test the system in a reasonable way based on how it is used. The goal of this effort is to ensure that the hardening process in no way affected the ability of applications or services to perform their functions.

If you discover that an application or service malfunctions after a system was hardened, determine the problem by reviewing the application log files. In many cases, you can use the `truss` command to determine at what point an application is having difficulty. Once this is known, you can target the problem and trace it back to a change made by the Solaris Security Toolkit software.

Maintaining System Security

A common mistake that many organizations make is addressing security only during installation, then rarely or never revisiting it. Maintaining security is an ongoing process. System security must be reviewed and revisited periodically.

Maintaining a secure system requires vigilance, because the default security configuration for any system becomes increasingly open over time. For example, system vulnerabilities become known. The following basic guidelines offer an overview:

- Solaris OS patches might install additional software packages as part of their installation and could overwrite your system configuration files. Ensure that you review the security posture of a system before and after any patch is installed. Also, it's important to keep your systems updated with the latest patches.

  The Solaris Security Toolkit software can assist you with applying patches, because it supports repetitive runs on a system, so that you can secure the system after installing patches. Run the software after any patch installation, with the applicable drivers, to ensure that your configuration remains consistent with your defined security policies. In addition, perform a manual review of the system, because the version of the Solaris Security Toolkit software being used might not support the new features added by the installed patches.
- Monitor the system on an ongoing basis to ensure that unauthorized behavior does not occur. Review system accounts, passwords, and access patterns; they can provide a great deal of information about what is happening to a system.
- Deploy and maintain a centralized `syslog` repository to collect and parse `syslog` messages. You can obtain valuable information by gathering and reviewing these logs.
- Institute a comprehensive vulnerability and audit strategy to monitor and maintain system configurations. This requirement is particularly important in the context of maintaining systems in secure configurations over time.
- Periodically update your systems with the latest version of the Solaris Security Toolkit software.

The Solaris Security Toolkit software includes default security profiles for use as a starting point.

CHAPTER 3

Installing and Running Security Software

This chapter provides instructions for downloading, installing, and running the Solaris Security Toolkit software and other security-related software. Included are instructions for configuring your environment for either standalone or JumpStart mode, and for obtaining support.

Follow the instructions and process provided in this section to install, configure, and execute the software. These instructions include downloading additional security software, helpful examples, and guidelines.

Although the Solaris Security Toolkit software is a standalone product, it is most effective when used with the additional security software provided for downloading. This software includes the latest Recommended and Security Patch Cluster from SunSolve OnLine, Secure Shell software for Solaris OS releases that do not include it, permission and ownership modification software to tighten Solaris OS and third-party software permissions, and integrity validation binaries to validate the integrity of Sun files and executables.

This section contains the following tasks:

Performing Planning and Preinstallation Tasks

Proper planning is key to successfully using the Solaris Security Toolkit software to secure systems. See Chapter 2 for detailed information about planning before you install the software.

If you are installing the software on a deployed system, see "Performing Preinstallation Tasks" on page 30, for information about performing preinstallation tasks prior to installing the software on deployed systems.

Dependencies

Solaris Security Toolkit 4.1 software has few dependencies.

Hardware Dependency

See "Running Supported Solaris OS Versions" on page 12 for information about supported versions of the Solaris Operating System.

Software Dependency

The Solaris Security Toolkit 4.1 software depends upon SUNWloc package. The absence of this package will cause the Solaris Security Toolkit to fail.

See "Running Supported SMS Versions" on page 13 for information about supported versions of the System Managements Services (SMS) software.

Determining Which Mode to Use

Harden systems during or immediately after installation, to limit the period a system might be exposed to attack while in an unsecured state. Before using the Solaris Security Toolkit software to secure a system, configure the Solaris Security Toolkit software to run properly in your environment.

The Solaris Security Toolkit software has a modular framework. If you are not using the JumpStart product, the flexibility of the Solaris Security Toolkit software's framework enables you to efficiently prepare for using JumpStart later. If you are using JumpStart, you benefit from the Solaris Security Toolkit software's ability to integrate into existing JumpStart architectures.

The following sections describe the standalone and JumpStart modes.

Standalone Mode

The Solaris Security Toolkit software runs directly from a Solaris OS shell prompt in standalone mode. This mode enables you to use the Solaris Security Toolkit software on those systems that require security modifications or updates, yet cannot be taken out of service to re-install the OS from scratch. However, systems should be reinstalled from scratch to secure them, if possible.

Standalone mode is particularly useful when hardening a system after installing patches. You can run the Solaris Security Toolkit software multiple times on a system with no ill effects. Patches might overwrite or modify files the Solaris Security Toolkit software has modified; by rerunning the Solaris Security Toolkit software, any security modifications negated by the patch installation can be reimplemented.

Note – In production environments, stage patches in test and development environments before installing the patches in live environments.

The standalone mode is one of the best options to harden a deployed system as quickly as possible. No special steps are required to integrate the Solaris Security Toolkit software into an architecture without JumpStart, other than those steps in the downloading and installing instructions provided in "Downloading Security Software" on page 38.

JumpStart Mode

JumpStart technology, which is Sun's network-based Solaris OS installation mechanism, can run Solaris Security Toolkit scripts during the installation process. This book assumes that the reader is familiar with JumpStart technology and has an existing JumpStart environment available. For more information about JumpStart technology, refer to the Sun BluePrints book *JumpStart Technology: Effective Use in the Solaris Operating Environment*.

For use in a JumpStart environment, copy the Solaris Security Toolkit source in either the `JASS_HOME_DIR` (for `tar` downloads) or `/opt/SUNWjass` (for `pkg` downloads) into the base directory of the JumpStart server. The default is `/jumpstart` on the JumpStart server. `JASS_HOME_DIR` becomes the base directory of the JumpStart server.

Only a few steps are required to integrate the Solaris Security Toolkit software into a JumpStart architecture. See Chapter 5 for instructions on how to configure a JumpStart server.

Downloading Security Software

The first stage in hardening a system requires downloading additional software security packages onto the system you want to secure. This section covers the following tasks:

Note – Of the software described in this section, the Solaris Security Toolkit software, Recommended and Security Patch Cluster, FixModes, and MD5 software are essential. Instead of OpenSSH, you can substitute a commercial version of Secure Shell, available from a variety of vendors. Install and use a Secure Shell product on all systems. If using Solaris 9 OS, use the Secure Shell version that is included.

Downloading Solaris Security Toolkit Software

First download the Solaris Security Toolkit software, then install it on the server on which you are using the Solaris Security Toolkit software in standalone mode or on a JumpStart server for JumpStart mode.

Note – The following instructions use filenames that do not reference the version number. Always download the latest version from the web site.

Throughout the rest of this guide, the `JASS_HOME_DIR` environment variable refers to the root directory of the Solaris Security Toolkit software. When the Solaris Security Toolkit software is installed from the `tar` archive, `JASS_HOME_DIR` is defined to be the path up to, and including, `jass-`*n.n*. If you install the `tar` version of the distribution in the `/opt` directory, the `JASS_HOME_DIR` environment variable is defined as `/opt/jass-`*n.n*.

The Solaris Security Toolkit software is distributed in Solaris OS package format, in addition to the traditional compressed `tar` archive. The same software is included in both archives.

Choose the format most appropriate for your situation. The `pkg` format is best for clients, and the `tar` is best for JumpStart systems and for developing custom packages.

Procedures for downloading and installing these two different archive types are provided in the following sections.

▼ To Download the `tar` Version

1. **Download the software distribution file** (`jass-`*n.n*`.tar.Z`)**.**

   The source file is located at the following web site:

   `http://www.sun.com/security/jass`

2. **Extract the software distribution file into a directory on the server using the** `zcat` **and** `tar` **commands as shown**:

```
# zcat jass-n.n.tar.Z | tar xvf -
```

   Where *n.n* is the most current version that you are downloading.

   Executing this command creates the `jass-`*n.n* subdirectory in the current working directory. This subdirectory contains all the Solaris Security Toolkit directories and associated files.

▼ To Download the pkg Version

1. **Download the software distribution file (`SUNWjass-n.n.pkg.Z`).**

   The source file is located at:

   `http://www.sun.com/security/jass`

   Note – If you encounter difficulty downloading the software, use your browser's integrated Save As option.

   ---

2. **Extract the software distribution file into a directory on the server by using the `uncompress` command**:

   ```
   # uncompress SUNWjass-n.n.pkg.Z
   ```

3. **Install the software distribution file into a directory on the server using the `pkgadd` command as shown:**

   ```
   # pkgadd -d SUNWjass-n.n.pkg SUNWjass
   ```

   Where *n.n* is the most current version that you are downloading.

   Executing this command creates the `SUNWjass` directory in `/opt`. This subdirectory contains all the Solaris Security Toolkit directories and associated files.

Downloading Recommended Patch Cluster Software

Patches are released by Sun to provide Solaris OS fixes for performance, stability, functionality, and security. It is critical to the security of a system that the most up-to-date patch cluster is installed. To ensure that the latest Solaris OS Recommended and Security Patch Cluster is installed on your system, this section describes how to download the latest patch cluster.

Note – Before installing any patches, evaluate and test them on nonproduction systems or during scheduled maintenance windows.

▼ To Download Recommended Patch Cluster Software

Before you install a patch cluster, review individual patch README files and other information provided. The information often contains suggestions and information helpful to know before installing a patch cluster.

1. **Download the latest patch cluster from the SunSolve OnLine Web site at:**

   `http://sunsolve.sun.com`

2. **Click the Patches link at the top of the left navigation bar.**

3. **Click the Recommended Patch Clusters link.**

   The license agreement is displayed.

4. **Select the appropriate Solaris OS version in the Recommended Solaris Patch Clusters box.**

   In our example, we select Solaris 8 OS.

5. **Select the best download option, either HTTP or FTP, with the associated radio button, then click Go.**

   A Save As dialog box is displayed in your browser window.

6. **Save the file locally.**

7. **Move the file securely to the system being hardened.**

   Use the `scp` (`scp(1)` – secure copy (remote copy program)) command, or another method that provides secure file transfer.

   Use the `scp` command as follows:

```
# scp 8_Recommended.zip target01:
```

8. **Move the file to the** `/opt/SUNWjass/Patches` **directory and uncompress it.**

   For example:

CODE EXAMPLE 3-1 Moving a Patch File to `/opt/SUNWjass/Patches` Directory

```
# cd /opt/SUNWjass/Patches
# mv /directory in which file was saved/8_Recommended.zip .
# unzip 8_Recommended.zip
Archive:     8_Recommended.zip
   creating: 8_Recommended/
  inflating: 8_Recommended/CLUSTER_README
  inflating: 8_Recommended/copyright
  inflating: 8_Recommended/install_cluster
[. . .]
```

The patch cluster software is installed automatically after you download the other security packages and execute the Solaris Security Toolkit software.

Note – If you do not place the Recommended and Security Patch Cluster software into the `/opt/SUNWjass/Patches` directory, a warning message displays when you execute the Solaris Security Toolkit software. You can safely ignore this message if no patch clusters apply, as is often the case with new releases of the OS.

Downloading FixModes Software

FixModes is a software package that tightens the default Solaris OS directory and file permissions. Tightening these permissions can significantly improve overall security. More restrictive permissions make it even more difficult for malicious users to gain privileges on a system.

Note – With the Solaris 9 OS release, changes were made to improve the default permissions of objects previously altered by the FixModes software. However, the FixModes software is still necessary, because third-party and unbundled software require tightening of file and directory permissions.

▼ To Download FixModes Software

1. **Download the FixModes precompiled binaries from:**

   `http://www.sun.com/security/jass`

   The FixModes software is distributed as a precompiled and compressed package version file formatted for Solaris OS systems. The file name is `SUNBEfixm.pkg.Z`.

2. **Move the file securely to the system being hardened by using the `scp` command, or another method that provides secure file transfer.**

   Use the `scp` command as follows:

```
# scp SUNBEfixm.pkg.Z target01:
```

3. **Uncompress and save the file, `SUNBEfixm.pkg.Z`, in the Solaris Security Toolkit `Packages` directory in `/opt/SUNWjass/Packages`, with the following commands:**

```
# uncompress SUNBEfixm.pkg.Z
# mv SUNBEfixm.pkg /opt/SUNWjass/Packages/
```

   Later, the FixModes software is installed automatically after downloading all the other security packages and executing the Solaris Security Toolkit software.

Downloading OpenSSH Software

In any secured environment, the use of encryption in combination with strong authentication is required to protect user-interactive sessions. At a minimum, network access must be encrypted.

The tool most commonly used to implement encryption is Secure Shell software, either a version bundled with the Solaris OS, a third-party commercial version, or a freeware version. To implement all the security modifications performed by the Solaris Security Toolkit software, you must include a Secure Shell software product.

Note – If using Solaris 9 OS, use the version of Secure Shell provided with the software. This version of Secure Shell integrates with other Solaris OS security features such as the Basic Security Module (BSM) as well as its support by Sun's support organization.

Information on where to obtain commercial versions of Secure Shell is provided in "Related Resources" on page xx.

The Solaris Security Toolkit software disables all unencrypted user-interactive services and daemons on the system, in particular daemons such as `in.telnetd`, `in.ftpd`, `in.rshd`, and `in.rlogind`.

Secure Shell enables you to gain access to the system as you would using Telnet and FTP.

▼ To Download OpenSSH Software

Note – If the server is running Solaris 9 OS, you can use the bundled Secure Shell software and skip the OpenSSH installation steps in this section.

- **Obtain the following Sun BluePrints OnLine article, and use the instructions in the article for downloading the software.**

  A Sun BluePrints OnLine article about how to compile and deploy OpenSSH titled "Building and Deploying OpenSSH on the Solaris Operating Environment" is available at:

  `http://www.sun.com/blueprints`

  Or obtain the Sun BluePrints publication *Secure Shell in the Enterprise*, which is available at book stores.

  After downloading all the other security packages and executing the Solaris Security Toolkit software, the OpenSSH software is installed automatically.

Caution – Do not compile OpenSSH on the system being hardened and do not install the compilers on the system being hardened. Use a separate Solaris OS system—running the same Solaris OS version, architecture, and mode (for example, Solaris 8 OS, Sun4U (`sun4u`), and 64-bit)—to compile OpenSSH. If you implement a commercial version of SSH, no compilation is required. The goal is to limit the availability of compilers to potential intruders. However, refraining from installing compilers locally on a system does not provide significant protection against determined attackers, because they can still install precompiled tools.

Downloading the MD5 Software

The MD5 software generates MD5 digital fingerprints on the system being hardened. Generate the digital fingerprints, then compare them with what Sun has published as correct, to detect system binaries that are altered or *trojaned* (hidden inside

something that appears safe) by unauthorized users. By modifying system binaries, attackers provide themselves with backdoor access onto a system; they hide their presence and could cause systems to operate in unstable manners.

▼ To Download the MD5 Software

1. **Download the MD5 binaries from the following web site:**

   `http://www.sun.com/security/jass`

   The MD5 programs are distributed as a compressed package version file.

2. **Move the file** `SUNBEmd5.pkg.Z` **securely to the system being hardened with the** `scp` **command, or another method that provides secure file transfer.**

   Use the `scp` command as follows:

```
# scp SUNBEmd5.pkg.Z target01:
```

3. **Uncompress and move the file to the Solaris Security Toolkit** `Packages` **directory in** `/opt/SUNWjass/Packages`**, using a command similar to the following:**

```
# uncompress SUNBEmd5.pkg.Z
# mv SUNBEmd5.pkg /opt/SUNWjass/Packages/
```

   After the MD5 software is saved to the `/opt/SUNWjass/Packages` directory, the execution of the Solaris Security Toolkit software installs the software.

   After the MD5 binaries are installed, you can use them to verify the integrity of executables on the system through the Solaris fingerprint database. More information on the Solaris fingerprint database is available in the Sun BluePrints OnLine article titled “The Solaris Fingerprint Database — A Security Tool for Solaris Software and Files.”

4. **(Optional) Download and install Solaris Fingerprint Database Companion and Solaris Fingerprint Database Sidekick software from the Sun BluePrint web site at:**

   `http://www.sun.com/blueprints/tools`

   Install and use these optional tools with the MD5 software. These tools simplify the process of validating system binaries against the database of MD5 checksums. Use these tools frequently to validate the integrity of the Solaris OS binaries and files on a secured system.

   These tools and instructions for downloading them are in the Sun BluePrints OnLine article titled "The Solaris Fingerprint Database — A Security Tool for Solaris Software and Files."

   The integrity of the security tools downloaded should be verified. Before installing and running the Solaris Security Toolkit software and additional security software, validate integrity by using MD5 checksums. On the download page of the Solaris Security Toolkit, MD5 checksums are available for this purpose.

Customizing Security Profiles

A variety of security profile templates are included with the Solaris Security Toolkit software distribution as drivers. As mentioned in the previous chapter, the default security profile and changes made by these drivers might not be appropriate for your systems. The security profiles implemented by these drivers disable services that are not required and enable optional security features disabled by default.

Before running the Solaris Security Toolkit software, review and customize the default security profiles for your environment, or develop new ones. Techniques and guidelines for customizing security profiles are provided in the *Solaris Security Toolkit 4.1 Reference Manual*.

Installing and Executing the Software

It is important that the following preliminary tasks be completed prior to executing the Solaris Security Toolkit software. Most of the hardening is done automatically when you execute the Solaris Security Toolkit software.

- Download the additional security software and the Solaris Security Toolkit software on the system you want to harden or on the JumpStart server. See "Downloading Security Software" on page 38.

- Configure your system for standalone or JumpStart mode. See "Determining Which Mode to Use" on page 37.
- If applicable, customize the Solaris Security Toolkit software for your environment.
- Before installing and running the Solaris Security Toolkit software and additional security software, validate integrity through the use of MD5 checksums.

You can execute the Solaris Security Toolkit software directly from the command line or a JumpStart server.

For command line options and other information about executing the software, see one of the following:

- "Executing the Software in Standalone Mode" on page 47
- "Executing the Software in JumpStart Mode" on page 56

Executing the Software in Standalone Mode

CODE EXAMPLE 3-2 shows a sample of command line usage in standalone mode.

CODE EXAMPLE 3-2 Sample Command Line Usage in Standalone Mode

```
# ./jass-execute -h

usage:

To apply this Toolkit to a system, using the syntax:
   jass-execute [-r root_directory -p os_version ]
      [ -q | -o output_file ] [ -m e-mail_address ]
      [ -V [3|4] ] [ -d ] driver

To undo a previous application of the Toolkit from a system:
   jass-execute -u [ -b | -f | -k ] [ -q | -o output_file ]
      [ -m e-mail_address ] [ -V [3|4] ]

To audit a system against a pre-defined profile:
   jass-execute -a driver [ -V [0-4] ] [ -q | -o output_file ]
      [ -m e-mail_address ]

To display the history of Toolkit applications on a system:
   jass-execute -H

To display the last application of the Toolkit on a system:
   jass-execute -l

To display this help message:
   jass-execute -h
```

CODE EXAMPLE 3-2 Sample Command Line Usage in Standalone Mode *(Continued)*

```
   jass-execute -?

To display version information for this program:
   jass-execute -v

#
```

TABLE 3-1 lists the command-line options available and describes each.

TABLE 3-1 Using Command-Line Options With `jass-execute`

Option	Description
`-a`	Determines if a system is in compliance with its security profile.
`-b`	Used with the –u option. Backs up any files that have manually changed since the last hardening run, then restores the system to its original state.
`-d`	Specifies the driver to be run in standalone mode.
`-f`	Used with the –u option. Reverses changes made during a hardening run without asking you about exceptions, even if files were manually changed after a hardening run.
`-h`	Displays the `jass-execute` help message, which provides an overview of the available options.
`-H`	Displays the history of the Solaris Security Toolkit software on the system.
`-k`	Used with the –u option. Keeps any manual changes you made since the last hardening run.
`-l`	Displays the last application of the Solaris Security Toolkit on the system.
`-m`	Mails output to a specified email address.
`-o`	Directs output to a specified file.
`-p`	Used with the `-r` *root_directory* option. Specifies the OS version of the Solaris operating system. The format is the same as that of `uname -r`.
`-q`	Prevents the display of output to the screen. Also known as the quiet option.

TABLE 3-1 Using Command-Line Options With `jass-execute` *(Continued)*

Option	Description
`-r`	Must be used with the `-p` *os_version*. Specifies the root directory used during `jass-execute` runs. By default, the root filesystem is `/`. This root directory is defined by the Solaris Security Toolkit (JASS) environment variable, `JASS_ROOT_DIR`. The Solaris OS being secured is available through `/`. For example, if you wanted to secure a separate OS directory, temporarily mounted under `/mnt` then use the `-r` option to specify `/mnt`.
`-u`	Runs undo option with interactive prompts that ask you what action you want to take when exceptions are encountered. Cannot be used with –d, –a, –h, –l, or –H options
`-v`	Displays version information for this program.
`-V`	Specify level of detail in message output.
`-?`	Displays the `jass-execute` help message, which provides an overview of the available options.

For detailed information about the options available with `jass-execute` command in standalone mode, see the following sections:

- "Audit Option" on page 51
- "Display Help Option" on page 51
- "Driver Option" on page 52
- "Email Notification Option" on page 53
- "Execute History Option" on page 53
- "Most Recent Execute Option" on page 54
- "Output File Option" on page 54
- "Quiet Output Option" on page 55
- "Root Directory Option" on page 55
- "Undo Option" on page 56

For a complete listing of available drivers, see the Drivers directory. Newer versions of the software may contain additional drivers.

▼ To Execute the Software in Standalone Mode

1. **Execute the** `secure.driver` **(or a product specific-script such as** `sunfire_15k_sc-secure.driver`**) as follows:**

CODE EXAMPLE 3-3 Executing the Software in Standalone Mode

```
# cd /opt/SUNWjass
# ./jass-execute -d secure.driver

[NOTE] The following prompt can be disabled by setting
JASS_NOVICE_USER to 0.
[WARN] Depending on how the Solaris Security Toolkit is configured,
it is both possible and likely that by default all remote shell
and file transfer access to this system will be disabled upon
reboot effectively locking out any user without console access to
the system.

Are you sure that you want to continue? (YES/NO) [NO]
y

[NOTE] Executing driver, secure.driver

============================================================
secure.driver: Driver started.
============================================================

============================================================
Solaris Security Toolkit Version: 4.1.0
Node name:                        ufudu
Host ID:                          8085816e
Host address:                     10.8.31.115
MAC address:                      8:0:20:85:81:6e
OS version:                       5.9
Date:                             Tue May 4 16:28:24 EST 2004
============================================================
[...]
```

For a complete listing of available drivers, see the Drivers directory. Newer versions of the software may contain additional drivers.

2. **After running the Solaris Security Toolkit software on a system, reboot the system to implement the changes.**

During hardening, a variety of modifications are made to the configuration of the client. These modifications might include disabling startup scripts for services, disabling options for services, and installing new binaries or libraries through patches. Until the client is restarted, these modifications might not be effective.

3. **After rebooting the system, verify the correctness and completeness of the modifications.**

   See "Validating the System Modifications" on page 57.

4. **If any errors are encountered, fix them and run the Solaris Security Toolkit software again in standalone mode.**

Audit Option

Through the `-a` option, the Solaris Security Toolkit software can perform an audit run to determine if a system is in compliance with its security profile. This run validates not only if system file modifications made are still active, but also if previously disabled processes are running or removed software packages are reinstalled. For more information on this function, see Chapter 6.

Example usage to audit a system against a security profile:

```
# jass-execute -a driver [ -V [0-4] ] [ -q | -o output-file ]
[ -m email-address ]
```

Display Help Option

The `-h` option displays the `jass-execute` help message, which provides an overview of the available options.

The `-h` option produces output similar to the following:

CODE EXAMPLE 3-4 Sample `-h` Option Output

```
# ./jass-execute -h
To apply this Toolkit to a system, using the syntax:
   jass-execute [-r root_directory -p os_version ]
      [ -q | -o output_file ] [ -m e-mail_address ]
      [ -V [3|4] ] [ -d ] driver

To undo a previous application of the Toolkit from a system:
   jass-execute -u [ -b | -f | -k ] [ -q | -o output_file ]
      [ -m e-mail_address ] [ -V [3|4] ]

To audit a system against a pre-defined profile:
   jass-execute -a driver [ -V [0-4] ] [ -q | -o output_file ]
      [ -m e-mail_address ]

To display the history of Toolkit applications on a system:
   jass-execute -H
```

CODE EXAMPLE 3-4 Sample -h Option Output *(Continued)*

```
To display the last application of the Toolkit on a system:
   jass-execute -l

To display this help message:
   jass-execute -h
   jass-execute -?

To display version information for this program:
   jass-execute -v

Note that just the driver name should be specified when using the
'-d' or '-a' options. A path need not be specified as the script
is assumed to exist in the Drivers directory.

The '-u' undo option is mutually exclusive with the '-d' and '-a'
options. The default undo behavior is to ask the user what to do if
a file to be restored has been modified as the checksum is
incorrect.

The -u option can be combined with the '-k', '-b', or '-f' to
override the default interactive behavior. The use of one of these
options is required when run in quiet mode ('-q').

The '-k' option can be used to always keep the current file and
backup if checksum is incorrect. The 'b' can be used to backup the
current file and restore original if the checksum is incorrect.
The 'f' option will always overwrite the original if the checksum
is incorrect, without saving the modified original.
```

Driver Option

The -d *driver* option specifies the driver to be run in standalone mode.

You must specify a driver with the -d option. The Solaris Security Toolkit software prepends Drivers/ to the name of the script added. You need to enter only the script name on the command line.

Note – You cannot use the -d option with the -u, -H, -h, or -a options.

A `jass-execute` hardening run using the `-d` *driver* option produces output similar to the following:

CODE EXAMPLE 3-5 Sample `-d` *driver* Option Output

```
# ./jass-execute -d secure.driver
[...]
[NOTE] Executing driver, secure.driver

============================================================
secure.driver: Driver started.
============================================================

============================================================
Solaris Security Toolkit Version: 4.1.0
Node name:                        ufudu
Host ID:                          8085816e
Host address:                     10.8.31.115
MAC address:                      8:0:20:85:81:6e
OS version:                       5.9
Date:                             Tue Oct 4 16:28:24 EST 2004
============================================================
[...]
```

Email Notification Option

The `-m` *email-address* option provides a mechanism by which standalone hardening and undo output can be emailed automatically by the Solaris Security Toolkit software when the run completes. The email report is in addition to any logs generated on the system using other options.

A Solaris Security Toolkit run calling `sunfire_15k_sc-config.driver` using the email option would be similar to the following:

```
# ./jass-execute -m root -d sunfire_15k_sc-config.driver
[...]
```

Execute History Option

The `-H` option provides a simple mechanism to determine how many times the Solaris Security Toolkit software has been run on a system. All runs are listed regardless of whether they have been undone.

The -H option produces output similar to the following:

CODE EXAMPLE 3-6 Sample -H Option Output

```
# ./jass-execute -H
Note: This information is only applicable for applications of
      the Solaris Security Toolkit starting with version 0.3.

The following is a listing of the applications of the Solaris
Security Toolkit on this system.  This list is provided in
reverse chronological order:

1.   June 31, 2004 at 12:20:19 (20040631122019) (UNDONE)
2.   June 31, 2004 at 12:10:29 (20040631121029)
3.   June 31, 2004 at 12:04:15 (20040631120415)
```

The output indicates that the Solaris Security Toolkit software was run on this system three times and that the last run was undone.

Most Recent Execute Option

The -l option provides a mechanism to determine the most recent run. This is always the last run listed by the -H option as well.

The -l option provide outputs similar to the following:

CODE EXAMPLE 3-7 Sample -l Option Output

```
# ./jass-execute -l

Note: This information is only applicable for applications of
      the Solaris Security Toolkit starting with version 4.1.0.

The last application of the Solaris Security Toolkit was:

1.   June 31, 2004 at 12:20:19 (20040631122019) (UNDONE)
```

Output File Option

The -o *output-file* option redirects the console output of jass-execute runs to a separate file, *output-file*.

This option has no effect on the logs kept in the JASS_REPOSITORY directory. This option is particularly helpful when performed over a slow terminal connection, because there is a significant amount of output generated by a Solaris Security Toolkit run.

This option can be used with either the -d, -u, or -a options.

The -o option produces output similar to the following:

CODE EXAMPLE 3-8 Sample -o Option Output

```
# ./jass-execute -o jass-output.txt -d secure.driver
[NOTE] Executing driver, secure.driver
[NOTE] Recording output to jass-output.txt
```

Quiet Output Option

The -q option disables Solaris Security Toolkit output to standard input output (stdio) stream during a hardening run.

This option has no effect on the logs kept in the JASS_REPOSITORY directory. Similar to the -o option, this option is particularly helpful when running the Solaris Security Toolkit software through a cron job or over slow network connections.

This option can be used with either the -d, -u, or -a options.

The -q option produces output similar to the following:

CODE EXAMPLE 3-9 Sample -q Option Output

```
# ./jass-execute -q -d secure.driver
[NOTE] Executing driver, secure.driver
```

Root Directory Option

The -r *root-directory* option is for specifying the root directory used during jass-execute runs. Using the -r option also requires using the -p option to specify the platform (OS) version. The format of the -p option is equivalent to that produced by uname -r.

By default, the root filesystem directory is /. This root directory is defined by the Solaris Security Toolkit environment variable JASS_ROOT_DIR. The Solaris OS being secured is available through /. For example, if you want to secure a separate OS directory, temporarily mounted under /mnt, then use the -r option to specify /mnt. All the scripts are applied to that OS image.

Undo Option

Through the `-u` option, the Solaris Security Toolkit software can undo system modifications performed during hardening. Each finish script can be undone with the `-u` option. In addition, the Solaris Security Toolkit's undo ability is tightly integrated with the checksums generated during each run. For more information on this capability, see Chapter 4.

Example command line usage of an undo command:

```
# jass-execute -u [ -b | -f | -k ] [ -q | -o output_file ]
        [ -m e-mail_address ] [ -V [3|4] ]
```

Executing the Software in JumpStart Mode

The JumpStart mode is controlled by the Solaris Security Toolkit driver inserted in the `rules` file on the JumpStart server.

If you have not configured your environment to use JumpStart mode, see Chapter 5.

For more information on the JumpStart technology, refer to the Sun BluePrints book *JumpStart Technology: Effective Use in the Solaris Operating Environment*.

▼ To Execute the Software in JumpStart Mode

To execute the Solaris Security Toolkit software in JumpStart mode, it must be integrated into your JumpStart environment and called as part of the finish scripts associated with a JumpStart installation. For information about how to integrate the Solaris Security Toolkit software into your environment, see Chapter 5.

1. **After making all of the required modifications to the drivers, install the client using the JumpStart infrastructure.**

   This task is done using the following command from the client's `ok` prompt.

```
ok> boot net - install
```

   Once the installation is completed, the system is rebooted by the JumpStart software.

   The system should be in its correct configuration. During hardening, a variety of modifications are made to the configuration of the client. These modifications could include disabling startup scripts for services, disabling options for services, and installing new binaries or libraries through patches. Until the client is restarted, these modifications might not be effective.

2. **After the system is rebooted, verify the correctness and completeness of the modifications.**

   See "Validating the System Modifications" on page 57.

3. **If any errors are encountered, fix them and reinstall the client's OS.**

Validating the System Modifications

After rebooting the system, validate the correctness and completeness of the modifications as described in the following sections.

Performing QA Checks of Services

One of the significant challenges involved in securing systems is determining what OS services must be left enabled for the system to function properly. Solaris OS services might be needed because they are used directly, such as Secure Shell to log into a system. Or they could be used indirectly, such as using the Remote Procedure Call (RPC) daemon for the graphical user interface of third-party software management tools.

Most of these requirements should be determined before running the Solaris Security Toolkit software. (See "Determining Application and Service Requirements" on page 20.) However, the only definitive mechanism is to install and secure the system, then perform thorough testing of its required functionality through quality assurance (QA) testing. A QA plan should be executed in place for any new system

being deployed after the system is hardened. Similarly, for deployed systems being hardened, thorough testing must be performed to ensure that all required and expected functionality is present.

If the QA process uncovers any discrepancies, perform the following:

1. Determine the problem area, based on the guidelines in Chapter 2.
2. Validate that the application runs in the modified configuration.
3. Undo the Solaris Security Toolkit run.
4. Modify the security profile (driver) based on the problem resolution.
5. Run the Solaris Security Toolkit software again.

The end result should be a security profile that can be run on the system without adversely affecting any required functionality.

Performing Security Assessments of Configuration

While validating that the system performs all required functions, also evaluate the security configuration to determine if the system is secured to the desired level. Depending on what hardening or minimization was performed on the system, this may involve different aspects.

At a minimum, the configuration of the system should be reviewed in the following ways:

- Ensure that all appropriate Security and Recommended Patches are installed.
- Verify that only required and appropriate processes are running, and that they are running with the appropriate arguments.
- Ensure that only required daemons are running, and that they are running with the appropriate arguments.
- Verify that only required ports are open on the system by checking locally (for example, `netstat -a`) and remotely by using a port scanner such as Nmap, which can determine which ports are available on a network interface.
- Make sure that only required Solaris OS packages were installed if the system was minimized.

This review should be considered a minimum for newly built and secured systems. When hardening legacy systems, the underlying OS should be verified to determine if unauthorized modifications were made. Integrity checking of this nature is best done by mounting the system's file system in read-only mode and running integrity

checking software from a known OS instance. The tools described in the Sun BluePrints OnLine article titled "The Solaris Fingerprint Database—A Security Tool for Solaris Software and Files" are useful in these scenarios.

Validating Security Profile

After a system is secured and you validate its required services and capabilities, use the audit function to make sure that the security profile was applied properly and completely. This task is critical for two reasons. The first is to ensure that the system is hardened as required. The second is to ensure that the security profile defined for the system is properly reflected in the Solaris Security Toolkit configuration. This check is critical because the configuration information is used to maintain the security profile of the system over its entire deployed life cycle.

For more information about the audit function, see Chapter 6.

Performing Post-Installation Tasks

If you installed the software on a deployed system, see "Performing Post-Installation Tasks" on page 31, for information about performing post-installation tasks on deployed systems.

CHAPTER 4

Reversing System Changes

This chapter provides information and procedures for reversing (undoing) the changes made by the Solaris Security Toolkit software during hardening runs. This option provides an automated mechanism by which you can return a system to its state prior to a Solaris Security Toolkit hardening run or sequence of runs.

This chapter contains the following topics:

Understanding How Changes Are Logged and Reversed

Each Solaris Security Toolkit hardening run creates a run directory in `JASS_REPOSITORY`. The names of these directories are based on the date and time the run is initiated. In addition to displaying the output to a screen, the Solaris Security Toolkit software creates a set of files in the directory to track the changes and log the operations.

The files stored in the directory track modifications performed on the system and enable the undo feature to work.

Caution – The contents of the files in the `JASS_REPOSITORY` should never be modified by an administrator.

When you use the Solaris Security Toolkit software to harden a system, either in JumpStart or standalone mode, the software logs the changes in the `JASS_REPOSITORY/jass-manifest.txt` file. This file lists operations that the undo feature uses to reverse changes. The file contains information about the hardening operations implemented by the Solaris Security Toolkit software, including files created, copied, moved, or removed. In addition, this file may contain both standard and custom entries that are required when reversing more complex changes, such as software package installations. A separate `jass-manifest.txt` file is created for each hardening run.

Note – The Solaris Security Toolkit software undo feature only reverses changes for which there are entries in manifest files.

The undo run goes through the manifest files generated during a Solaris Security Toolkit run and stored in the `JASS_REPOSITORY`. The run restores the backed-up files to their original locations. If files were not backed up, then the undo function is not available.

When a Solaris Security Toolkit run is undone, the associated directory is not removed. Instead, two files are created in the `JASS_REPOSITORY` directory: `jass-undo-log.txt` and `reverse-jass-manifest.txt`. Afterward, the run that was undone is not listed the next time `jass-execute -u` is executed. A hardening run can be undone only once.

Requirements for Undoing System Changes

Be aware of the following limitations and requirements for using the undo feature of the Solaris Security Toolkit software.

- In Solaris Security Toolkit versions 0.3 through 4.1, you can use the undo feature for runs that were initiated in either standalone or JumpStart mode. However, you can undo changes only in standalone mode. The undo feature cannot be used during a JumpStart installation.
- If you select the Solaris Security Toolkit option not to create backup files, either through JumpStart or standalone modes, the undo feature is not available. The creation of back-up file copies is disabled by setting the `JASS_SAVE_BACKUP` parameter to `0`.
- A run can only be undone once.

- If you develop a new finish script that does not use the Solaris Security Toolkit framework functions, you must create a matching audit script and add entries to the manifest file by using the `add_to_manifest` function. Otherwise, the Solaris Security Toolkit has no way of knowing about your custom development.
- Do not modify the contents of the `JASS_REPOSITORY` directories under any circumstances. Modifying the files can corrupt the contents and cause unexpected errors or system corruption when you use the undo feature.

Customizing Scripts to Undo Changes

The Solaris Security Toolkit framework provides flexibility for designing and building finish scripts. The framework allows you to extend the capabilities of the Solaris Security Toolkit software to better meet the needs of your organization while also helping you to better manage the configuration of systems over their life cycles.

When customizing scripts, it is important to understand how the actions you take can affect the undo feature. To simplify customizing scripts, helper functions make the appropriate changes to the manifest files. (The undo feature relies on the contents of manifest files to reverse hardening runs.) In most cases, these helper functions provide what you need to customize scripts for your organization.

For a list of helper functions and information about using them, refer to the *Solaris Security Toolkit 4.1 Reference Manual*. Use these helper functions in place of their system command counterparts, so that undo runs can reference the related entries in manifest files.

In some cases, you might need to perform a function for which there is not a helper function. In these cases, use the special function called `add_to_manifest`. Using this function, you can manually insert entries into manifest files without needing to call one of the helper functions. Use this special function with care, so that you protect the integrity of the system and the Solaris Security Toolkit repository. An example of when you might use this special function is when you want to add software packages that are not in Sun's `pkg` format. In this example, you would need to tell the undo feature how to remove the packages that were added in another format during the hardening run.

With the helper functions and the special `add_to_manifest` function, the Solaris Security Toolkit software provides a simple and flexible way to customize scripts and have the changes extended to undo runs.

If you make changes to a finish script's behavior without using these functions, there is no way for the Solaris Security Toolkit software to know that a change was made. Therefore, you would have to manually undo any changes that are not referenced in manifest files.

In another example, before modifying a file on the system, the original version of the file should first be saved. Outside the context of the Solaris Security Toolkit software, typically users accomplish this task by executing the `/usr/bin/cp` command. However, within the context of the Solaris Security Toolkit software, if you use this command directly, the Solaris Security Toolkit software has no way of knowing that a manifest entry needs to be created. Instead of using the `cp` command, use the `backup_file` helper function. This function saves a copy of the original file, with a suffix of `JASS_SUFFIX`, and adds a manifest entry instructing the Solaris Security Toolkit software that a copy of the file was made. Also, this function causes the file checksums to be calculated. File checksums are used both by the undo feature as well as the `jass-check-sum` command.

Checking for Files That Were Manually Changed

Although using the `jass-execute -u` command automatically checks for files that were changed manually after a hardening run, sometimes you might find it helpful to use the `jass-check-sum` command to list and review the files that have been changed.

This command enables you to review the contents of the `JASS_REPOSITORY` and perform checksums on all of the files listed in manifest files to determine which files listed have changed since their checksums were recorded during a hardening run. Performing this check before proceeding with a forced undo run provides valuable information that may save many hours of needless troubleshooting.

The following is an example output.

CODE EXAMPLE 4-1 Sample Output of Files That Were Manually Changed

```
# ./jass-check-sum
File Name              Saved CkSum             Current CkSum
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
/etc/inet/inetd.con    1643619259:6883         2801102257:6879
/etc/logadm.conf       2362963540:1042         640364414:1071
/etc/default/inetd     3677377803:719          2078997873:720
```

The output indicates that three files were changed after the hardening run was completed.

Using Options With Undo Feature

This section describes the `jass-execute -u` command and options that you can use when executing an undo run.

Note – You cannot use the `-d`, `-a`, `-h`, `-l` or `-H` options with the undo feature. You must provide either `-b`, `-k`, or `-f` options when running `undo` in quiet mode.

The `jass-execute -u` command is the standard for executing an undo run. This command automatically discovers any files that were manually modified since the last hardening run. If the Solaris Security Toolkit software discovers files that were manually changed after a hardening run, it asks you to choose a response:

1. Back up the most current file before restoring the original (the one that existed before the hardening run).
2. Keep the most current file, and do not restore the original file.
3. Force an overwrite to any manually changed file (might lose data) and restore original file.

If you want to change the default undo behavior, use the `-b`, `-k`, and `-f` options when executing the undo command.

TABLE 4-1 lists command-line options you can use with undo. For detailed information about each option, see the sections that follow.

TABLE 4-1 Using Command-Line Options With Undo Command

Option	Description
`-b`	Backs up any files that have been manually changed since the last hardening run, then restores system to original state.
`-f`	Reverses changes made during a hardening run without asking you about exceptions, even if files were manually changed after a hardening run.
`-k`	Keeps any manual changes you made to files after a hardening run.
`-m`	Mails output to an email address.
`-o`	Directs output to a file.
`-q`	Prevents the display of output to the screen. Also known as the quiet option. Output is stored in `JASS_REPOSITORY/jass-undo-log.txt`.

Backup Option

The `-b` option automatically backs up any files that have been manually changed since the last hardening run, then restores the files to their original state prior to the hardening run. To implement the manual changes, you need to compare the restored files with the backed up files, and reconcile the differences manually. If a file is backed up using this option, it appears similar to the following example.

```
/etc/motd.BACKUP.JASS_SUFFIX
```

Force Option

The `-f` option reverses the changes made during a hardening run with no exceptions, even if files were manually changed after a hardening run. The undo run does not compare the saved file checksums to the current versions of the files. As a result, if you manually changed files after a hardening run, the changes would be overwritten and lost after the undo run.

It may be necessary to manually re-implement changes after the undo run completes. Furthermore, it may be necessary to reconcile differences between groups of files depending on the types of changes made. To help prevent these problems, use the `jass-check-sum` command or the `-b` command-line option mentioned previously.

Keep Option

The `-k` option automatically keeps any manual changes you made to files after a hardening run instead of restoring the original files. The `-k` option discovers any mismatches in files, causes a notice to be generated and logged, and does not overwrite the file with the original. The only changes reversed are those for which the saved checksums in `jass-checksums.txt` file are valid.

This option is not without its drawbacks. For example, a system can be rendered into an inconsistent state if a subset of files modified by a finish script are later modified.

Consider the `remove-unneeded-accounts.fin` finish script. This script modifies both the `/etc/passwd` and `/etc/shadow` files on the system. If a user manually changes a password after a hardening run is finished, then the checksum associated with the `/etc/shadow` file does not match the value saved by the Solaris Security Toolkit software. As a result, if the keep option is used, then only the `/etc/passwd` file is copied back to its original state. The `/etc/shadow` file remains in its current form. The two files are no longer consistent.

Output File Option

The `-o` *output-file* option redirects the console output of `jass-execute` runs to a separate file, *output-file*.

This option has no affect on the logs kept in the `JASS_REPOSITORY` directory. This option is particularly helpful when performed over a slow terminal connection, because there is a significant amount of output generated by a Solaris Security Toolkit undo run.

Quiet Output Option

The `-q` option prevents the Solaris Security Toolkit software from displaying output to the screen. This option has no affect on the logs kept in the `JASS_REPOSITORY` directory. Similar to the `-o` option, this option is particularly helpful when running the Solaris Security Toolkit through a `cron` job or over slow network connections.

Email Notification Option

The `-m` *email-address* option instructs the Solaris Security Toolkit software to e-mail a copy of the completed run to an email address. The email report is in addition to any logs generated on the system using other options.

Undoing System Changes

Sometimes it's necessary to reverse the changes made during one or multiple Solaris Security Toolkit hardening runs. If you find that the changes made during a hardening run have negatively impacted your system, undo the changes.

For example, if after a hardening run you discover that a required service such as NFS was disabled, undo the hardening run. Then, enable NFS and repeat the hardening run with the revised security profile.

This section provides instructions for reversing changes made during one or multiple hardening runs. Note that there are limitations and requirements for effectively reversing a hardening run. See "Requirements for Undoing System Changes" on page 62.

▼ To Undo a Solaris Security Toolkit Run

1. **Back up and reboot your system.**

   Reboot and back up the system before each undo run to ensure that it returns to or can be brought back to a known and working state.

2. **Determine which options you want to use with the `jass-execute -u` command.**

   See "Using Options With Undo Feature" on page 65.

   The following instructions assume that you are using the `jass-execute -u` command.

3. **To undo one or more hardening runs using the standard `-u` option, enter the following command from `JASS_HOME_DIR`:**

```
# ./jass-execute -u
```

   The Solaris Security Toolkit software collects information about each hardening run by finding all of the manifest files located in `JASS_REPOSITORY`. If a manifest file is empty or nonexistent, it is assumed that there are no changes to be undone and that run is omitted. In addition, if a file called `jass-undo-log.txt` exists in the same directory as the manifest file, it is assumed that the run has already been reversed, so that run is omitted. After the collection process is completed, the results are displayed. The following is an example output.

CODE EXAMPLE 4-2 Sample Output of Runs Available to Undo

```
# ./jass-execute -u
[NOTE] Executing driver, undo.driver
Please select a JASS run to restore through:
1. January 24, 2003 at 13:57:27
(/var/opt/SUNWjass/run/20030124135727)
2. January 24, 2003 at 13:44:18
(/var/opt/SUNWjass/run/20030124134418)
3. January 24, 2003 at 13:42:45
(/var/opt/SUNWjass/run/20030124134245)
4. January 24, 2003 at 12:57:30
(/var/opt/SUNWjass/run/20030124125730)

Choice? ('q' to exit)?
```

   In this example, four separate hardening runs are found. These runs made changes to the system and have not been undone. The list of hardening runs is always presented in reverse chronological order. The first entry in the list is the most recent hardening run.

4. **Review the output to determine which run(s) you want to undo, then enter the corresponding number.**

   For any entry selected, the Solaris Security Toolkit software reverses each run with an index number equal to or less than the value selected. That is, the undo run undoes the changes in the reverse order that they were originally made, starting with the most recent hardening run and continuing to the one you select. Using the

previous example as a guide, if you select run 3, then the `undo run` first reverses changes for run 1, then moves on to reverse changes for run 2, then finishes by reversing changes to run 3.

The following example shows output generated when the undo run processes two manifest file entries.

CODE EXAMPLE 4-3 Sample Output of an Undo Run Processing Multiple Manifest File Entries

```
[...]

================================================================
undo.driver: Performing UNDO of
//var/opt/SUNWjass/run/20030124135727.
================================================================

[...]

================================================================
undo.driver: Undoing Finish Script: update-cron-allow.fin
================================================================

[NOTE] Undoing operation COPY.
cp -p /etc/cron.d/cron.allow.JASS.20030125223417
/etc/cron.d/cron.allow
rm -f /etc/cron.d/cron.allow.JASS.20030125223417

[NOTE] Removing a JASS-created file.
rm -f /etc/cron.d/cron.allow

[...]
```

In this example, the Solaris Security Toolkit software undoes a copy operation and removes a file that was added during the hardening run. The output of an undo run documents the actual commands that are taken to restore the system, so that the process can be clearly understood and referenced in case you need to troubleshoot a system's configuration.

The undo run continues until all runs and corresponding manifest files are processed and the changes reversed.

In addition to the Solaris Security Toolkit software collecting information about each hardening run by finding all of the manifest files located in `JASS_REPOSITORY`, the Solaris Security Toolkit software compares the checksum of each modified file. Any mismatches in the checksum files cause a notice to be generated and logged. For these files, the undo run asks you what action you want to take.

5. **If the undo run discovers an exception (a file that was manually changed after the hardening run), enter one of the options.**

   The following is an example output showing an exception and the choices for handling the exception.

CODE EXAMPLE 4-4 Sample Output of Undo Exception

```
[...]

================================================================
undo.driver: Undoing Finish Script: install-templates.fin
================================================================

[NOTE] Undoing operation COPY.
cp -p /etc/skel/local.login.JASS.20030125223413
/etc/skel/local.login
rm -f /etc/skel/local.login.JASS.20030125223413

[NOTE] Undoing operation COPY.
[WARN] Checksum of current file does not match the saved value.
[WARN]    filename = /etc/.login
[WARN]    current = 3198795829:585, saved = 1288382808:584

Please select the course of action:

1. Backup.  Save current file before restoring original.
2. Keep.    Keep the current file, making no changes.
3. Force.   Ignore manual changes and overwrite current file.

Enter 1, 2, or 3:
```

In our example, if we choose item 1, the following output is displayed.

CODE EXAMPLE 4-5 Sample Output of Choosing Backup Option During Undo

```
Enter 1, 2, or 3: 1

[NOTE] BACKUP specified, creating backup copy of /etc/.login.
[NOTE] File to be backed up is from an undo operation.
[NOTE] Copying /etc/.login to /etc.login.BACKUP.JASS.20030125224926
cp -p /etc/.login.JASS.20030125223413 /etc/.login
rm -f /etc/.login.JASS.20030125223413

[...]
```

Take appropriate action regarding files that were manually modified after any hardening runs.

When an undo run encounters modified files and you choose to not overwrite them, reconcile these before rebooting the system.

Note – In our example, the modified file is saved with the new name: `/etc/.login.BACKUP.JASS.20030125224926`. After the undo run is complete, compare that file to `/etc/.login` to determine if any further reconciliation is needed.

---

6. **Reconcile any exceptions before continuing.**

7. **After reconciling any exceptions, reboot the system.**

   Rebooting the system is necessary for the system to stop and start services available before it was hardened.

CHAPTER 5

Configuring and Managing JumpStart Servers

This chapter provides information for configuring and managing JumpStart servers to use the Solaris Security Toolkit software. JumpStart technology, which is Sun's network-based Solaris OS installation mechanism, can run Solaris Security Toolkit software during the installation process.

The Solaris Security Toolkit's JumpStart mode is based on JumpStart technology, available for the Solaris OS product since version 2.1. JumpStart technology helps you manage complexity by fully automating the Solaris OS and system software installation, facilitating the correctness and standardization of systems. It provides a way to meet the requirements of rapidly installing and deploying systems.

The advantages of using JumpStart technology are apparent in the area of system security. By using JumpStart technology with the Solaris Security Toolkit software, you can secure systems during automated Solaris OS installations. This practice helps ensure that system security is standardized and addressed at the time of system installation. For more information about JumpStart technology, refer to the Sun BluePrints book *JumpStart Technology: Effective Use in the Solaris Operating Environment*.

This chapter contains the following topics:

Configuring JumpStart Servers and Environments

For use in a JumpStart environment, you must copy the Solaris Security Toolkit source in either the JASS_HOME_DIR (for tar downloads) or /opt/SUNWjass (for pkg downloads) into the base directory of the JumpStart server. The default directory is /jumpstart on a JumpStart server. After this task is done, JASS_HOME_DIR becomes the base directory of the JumpStart server.

This section assumes that the reader is familiar with JumpStart technology and has an existing JumpStart environment available.

Only a few steps are required to integrate the Solaris Security Toolkit software into a JumpStart architecture.

▼ To Configure for JumpStart Mode

1. **Copy the Solaris Security Toolkit source into the root directory of the JumpStart server.**

   For example, if the Solaris Security Toolkit archive was extracted to JASS_REPOSITORY, and the JumpStart server root directory is /jumpstart, the following command copies the Solaris Security Toolkit source:

```
# pwd
/opt/SUNWjass
# tar cf - . | (cd /jumpstart; tar xf -)
```

   Typically, the Solaris Security Toolkit software is installed in the SI_CONFIG_DIR of the JumpStart server, which would normally also be JASS_HOME_DIR.

2. **If you make any modifications to the Solaris 2.5.1 OS** sysidcfg **file, make them to the one in the** JASS_HOME_DIR/Sysidcfg/Solaris_2.5.1 **directory.**

   If you are using Solaris 2.5.1 OS, the sysidcfg file in JASS_HOME_DIR/Sysidcfg/Solaris_2.5.1 cannot be used directly because this version of Solaris only supports sysidcfg files in SI_CONFIG_DIR and not in separate subdirectories. To address this limitation on Solaris 2.5.1 OS, the Solaris Security Toolkit software has SI_CONFIG_DIR/sysidcfg, which is linked to the JASS_HOME_DIR/Sysidcfg/Solaris_2.5.1/sysidcfg file.

3. **Copy the** `JASS_HOME_DIR/Drivers/user.init.SAMPLE` **to** `JASS_HOME_DIR/Drivers/user.init` **with the following command:**

```
# pwd
/jumpstart/Drivers
# cp user.init.SAMPLE user.init
```

4. **If you experience problems with a multihomed JumpStart server, modify the two entries for** `JASS_PACKAGE_MOUNT` **and** `JASS_PATCH_MOUNT` **to the correct path to the** `JASS_HOME_DIR/Patches` **and** `JASS_HOME_DIR/Packages` **directories.**

5. **If you want to install the Solaris Security Toolkit software under a subdirectory of** `SI_CONFIG_DIR`**, such as** `SI_CONFIG_DIR/path/to/JASS`**, then add the following to the** `user.init` **file:**

```
if [ -z "${JASS_HOME_DIR}" ]; then
       if [ "${JASS_STANDALONE}" = 0 ]; then
              JASS_HOME_DIR="${SI_CONFIG_DIR}/path/to/JASS"
       fi
fi
export JASS_HOME_DIR
```

6. **Select or create a Solaris Security Toolkit driver (for example, the default** `secure.driver`**).**
   - If all the scripts listed in the `hardening.driver` and `config.driver` are to be used, then add the `Drivers/secure.driver` path to the `rules` file.
   - If only selected scripts are to be used, make copies of those files, then modify the copies.

7. **After completing the driver, make the appropriate entry in the** `rules` **file.**

   The entry should be similar to the following:

```
hostname imbulu - Profiles/core.profile Drivers/secure.driver
```

Caution – Never modify the original scripts included with the Solaris Security Toolkit software. To allow for efficient migration to new releases of the Solaris Security Toolkit software, maintain the original files and your custom files separately.

One other modification may be required to successfully integrate the Solaris Security Toolkit software into the existing JumpStart environment.

8. **If you use the `sysidcfg` files provided with the Solaris Security Toolkit software to automate the JumpStart client installation, review them for applicability.**

   If the JumpStart server encounters any errors while parsing the `sysidcfg` file, the entire content of the file is ignored.

   After completing all the configuration steps in this section, you are able to use JumpStart technology on the client, and successfully harden or minimize the OS during the installation process.

Using JumpStart Profile Templates

JumpStart profile templates are files used only with JumpStart mode. The required and optional contents of profiles are described in the Sun BluePrints book *JumpStart Technology: Effective Use in the Solaris Operating Environment*.

Use the JumpStart profile templates as samples from which to make your individual site modifications. Review the profiles to determine what changes are necessary, if any, to use in your environment.

Make copies of the profiles, then modify them for your site. Do not modify the originals, because updates to the Solaris Security Toolkit software might overwrite your customization.

The following JumpStart profiles are included with the Solaris Security Toolkit software:

- `32-bit-minimal.profile`
- `core.profile`
- `end-user.profile`
- `developer.profile`
- `entire-distribution.profile`
- `oem.profile`
- `minimal-Sun_ONE-WS-Solaris*.profile`
- `minimal-SunFire_Domain*.profile`

The following subsections describe these profiles.

`32-bit-minimal.profile`

This JumpStart profile is a relatively generic JumpStart profile for a 32-bit minimized system. It is a reasonable starting point for the development of minimized systems and was used as the starting point for the `minimal-Sun_ONE-WS-Solaris*.profile` minimization scripts.

core.profile

This JumpStart profile installs the smallest Solaris OS cluster, `SUNWCreq`. Other than specifying that the partitioning of the disk include a root and swap partitions, no other configuration modifications are made.

end-user.profile

This JumpStart profile installs the End User Solaris OS cluster, `SUNWCuser`, and the two Solaris OS packages required for process accounting to work properly. In addition, disk partitioning is defined to include only root and swap partitions.

developer.profile

This JumpStart profile installs the Developer Solaris OS cluster, `SUNWCprog`, and the two Solaris OS packages required for process accounting to work properly. As in the `core.profile` definition, the only other configuration definitions made, in addition to the Solaris OS cluster, are for the disk partitioning to include root and swap.

entire-distribution.profile

This JumpStart profile installs the Entire Distribution Solaris OS cluster, `SUNWCall`. As with the other profiles, disk partitioning is defined to include root and swap partitions.

oem.profile

This JumpStart profile installs the OEM Solaris OS cluster, `SUNWCXall`. This cluster is a superset of the Entire Distribution cluster, and it installs OEM-provided software.

minimal-Sun_ONE-WS-Solaris*.profile

All of the following profiles are based on the Sun BluePrints OnLine article *Minimizing the Solaris Operating Environment for Security*. All the Solaris OS versions addressed in this article have specific profiles. The following JumpStart profiles are the same as those referenced in the article.

- minimal-Sun_ONE-WS-Solaris.26.profile
- minimal-Sun_ONE-WS-Solaris7-32bit.profile
- minimal-Sun_ONE-WS-Solaris7-64bit.profile
- minimal-Sun_ONE-WS-Solaris8-32bit.profile
- minimal-Sun_ONE-WS-Solaris8-64bit.profile
- minimal-Sun_ONE-WS-Solaris9-64bit.profile

minimal-SunFire_Domain*.profile

All the following profiles are based on the Sun BluePrints OnLine article *Minimizing Domains for Sun Fire V1280, 12K, and 15K Systems*. The following JumpStart profiles are the same as those referenced in the article.

- minimal-SunFire_Domain-Apps-Solaris8.profile
- minimal-SunFire_Domain-Apps-Solaris9.profile
- minimal-SunFire_Domain-NoX-Solaris8.profile
- minimal-SunFire_Domain-NoX-Solaris9.profile
- minimal-SunFire_Domain-X-Solaris8.profile
- minimal-SunFire_Domain-X-Solaris9.profile

Adding and Removing Clients

The following information describes the scripts available for use in JumpStart mode. The mode is controlled by the Solaris Security Toolkit driver inserted in the `rules` file on the JumpStart server.

If you have not configured your environment to use JumpStart mode, see "Configuring JumpStart Servers and Environments" on page 74.

add-client Script

To simplify adding clients from JumpStart servers, use this script included with the Solaris Security Toolkit software. The command and options are described in the following paragraphs, however, the underlying JumpStart technology is not. Refer to the Sun BluePrints book *JumpStart Technology: Effective Use in the Solaris Operating Environment* for information about JumpStart technology.

The `add-client` script is a wrapper around the `add_install_client` command, which accepts the following arguments.

Example Usage:

```
# add-client -c client -i server -m client-class -o client-OS -s sysidcfg
```

TABLE 5-1 describes the valid input for the `add-client` command.

TABLE 5-1 JumpStart `add-client` Command

Value	Description
-c *client*	Resolvable host name of the JumpStart client.
-h	Displays usage information. Use without any other options. Any additional option will be ignored.
-i *server*	IP address or resolvable host name of the JumpStart server interface for this JumpStart client. If no value is specified, a list of interfaces available on the local host are displayed.
-m *client-class*	Machine class of the JumpStart client. This value is in the same format as the output of the `uname -m` command.
-o *client-OS*	Revision of the Solaris OS, available in the `JASS_HOME_DIR/OS` directory, that is to be installed on the client. If no value is specified, a list of available Solaris OS versions in the `JASS_HOME_DIR/OS` directory are displayed.
-s *sysidcfg*	Optional path name to an alternate directory containing a `sysidcfg` file that you want to use for system identification and configuration. By default, this value is set to the `JASS_HOME_DIR/Sysidcfg/Solaris_version/` directory, where the version is extracted from the OS specified for this client. If specified, a path name relative to the `JASS_HOME_DIR` directory must be used. Only specify the path to the `sysidcfg` file.
-v	The version information for this program.
-?	Displays usage information. Use without any other options. Any additional option will be ignored.

To add a JumpStart client called `jordan` to a JumpStart server called `nomex` using Solaris 8 OS (4/01) on an interface called `nomex-jumpstart`, you would use the following `add-client` command

```
#./add-client -c jordan -o Solaris_8_2001-04 -m sun4u -i nomex-jumpstart
updating /etc/bootparams
```

To add the same JumpStart client (`jordan`) using the `sysidcfg` option, you would use the following command:

```
#./add-client -c jordan -o Solaris_8_2001-04 -m sun4u -i nomex-jumpstart -s
Hosts/jordan
updating /etc/bootparams
```

`rm-client` Script

To simplify removing clients from JumpStart servers, use this script included with the Solaris Security Toolkit software. The command and options are described in the following paragraphs; however, the underlying JumpStart technology is not. Refer to *JumpStart Technology: Effective Use in the Solaris Operating Environment* for information about JumpStart technology.

The `rm-client` script is a wrapper around the `rm_install_client` command in much the same way as `add-client`:

Example Usage: **`rm-client`** `[-c]` *`client`*

Where *`client`* is the resolvable host name of the JumpStart client.

TABLE 5-2 describes the valid input for the `rm-client` command.

TABLE 5-2 JumpStart `rm-client` Command

Value	Description
`-c` *client*	Resolvable host name of the JumpStart client.
`-h`	Displays usage information. Use without any other options. Any additional option will be ignored.
`-v`	The version information for this program.
`-?`	Displays usage information. Use without any other options. Any additional option will be ignored.

To remove a JumpStart client called `jordan`, you would use the following `rm-client` command:

```
# ./rm-client -c jordan
removing jordan from bootparams
```

CHAPTER 6

Auditing System Security

This chapter describes how to audit (validate) a system's security using the Solaris Security Toolkit software. Use the information and procedures in this chapter for maintaining an established security profile after hardening. For systems that are already deployed, you may want to use the information in this chapter to assess security before hardening.

Note – The term *audit* is used in this chapter and book to define the Solaris Security Toolkit software's automated process of validating a security posture by comparing it with a predefined security profile. The use of this term in this publication does not guarantee that a system is completely secure after using the audit option.

This chapter contains the following topics:

Maintaining Security

Maintaining security is an ongoing process that must be reviewed and revisited periodically. Maintaining a secure system requires vigilance, because the default security configuration for any system tends to become increasingly open over time. (For more information about maintaining security, see "Maintaining System Security" on page 32.)

Based upon user experience and requests, we developed an automated method for the Solaris Security Toolkit software to audit the security posture of a system, by determining its level of compliance with a specified security profile.

Note – This method is only available in standalone mode using the `jass-execute -a` command and cannot be used during a JumpStart installation.

Audit the security posture of your systems periodically, either manually or automatically (for example, via `cron` job or an `rc` script). For example, after hardening a new installation, execute the Solaris Security Toolkit software audit command (`jass-execute -a` *driver-name*) five days later to determine if the system security has changed from the state defined by the security profile.

How often you audit security depends on the criticality of the environment and your security policy. Some users run an audit every hour, every day, or only once a month. Some users run a mini-scan (limited number of checks) every hour, and a full scan (with all the possible checks) once a day.

Audit an essential component to maintain the security posture of deployed systems. If security posture is not periodically audited, then configurations often drift over time due to entropy or modifications that unknowingly or maliciously change the desired security posture. Without periodic review, these changes go undetected and corrective measures are not taken. The result is a system that becomes less secure and more vulnerable.

In addition to periodic audits, perform audits after upgrades, patches, and other significant system configuration changes.

Reviewing Security Prior to Hardening

In some cases, you may find it useful to review the security posture on deployed systems *before* hardening them. For example, if you assume responsibility for deployed systems that another person administrated, inspect the state of the systems so that you know their posture and, if necessary, can bring them into compliance with the same security profiles used on your other systems.

Customizing Security Audits

The audit option provides a highly flexible and extensible mechanism for evaluating the state of a system. As with hardening scripts, you can customize the actions of audit scripts. For example, you can customize environment variables, customize framework and helper functions, add new checks, and add functionality to the audit framework.

Most users find the standard and product-specific audit scripts are suitable as templates from which to customize auditing for their environments. For this scenario, customize audit script actions through drivers, finish scripts, environment variables, and file templates. These custom changes can be made with little effort and without modifying the code. Whatever changes you make for hardening are automatically known by the Solaris Security Toolkit software when you perform auditing.

Occasionally, some users find it necessary to add checks or functionality that the Solaris Security Toolkit software does not provide. For this scenario, add the checks or new functionality to the audit script. (You might want to make related changes in the corresponding finish script.) In some cases, you may need to modify the code. Use extreme care when performing code additions and modifications, to avoid introducing bugs and failures.

Some users need to create entirely new proprietary, or site-specific, drivers and scripts. Use the templates and samples as guidelines when coding the new drivers and scripts. Be advised that site-specific drivers, finish scripts, variables, and functions are *not* automatically known to the Solaris Security Toolkit software when you use the audit option. For example, if you add a site-specific driver named `abcc-nj-secure.driver` that contains a site-specific finish script, `abcc-nj-install-foo.fin`, then you need to create a site-specific audit script, `abcc-nj-install-foo.aud`. Similarly, if you start with only the audit script, you should create the matching finish script.

To customize or create new drivers, scripts, variables, and functions, refer to the *Solaris Security Toolkit 4.1 Reference Manual*.

For example, you might need to add a patch that the Solaris Security Toolkit software does not install. You can extend one of the standard or product-specific templates, or you can create your own. If you create your own templates, create a finish script to add the patch, then create the corresponding audit script to check for the patch installation.

Preparing to Audit Security

To use the instructions and guidelines in this chapter, you need a security profile. For information about developing and implementing a security profile, see Chapter 2.

A variety of security profile templates are included with the Solaris Security Toolkit distribution as drivers. As mentioned earlier in this book, the default security profile and changes made by these drivers might not be appropriate for your systems. Typically, the security profiles implemented by these drivers are "high-water" marks for security. By this, we mean that they disable services that are not required, and they enable optional security features disabled by default.

Many Solaris Security Toolkit software users find that the standard and product-specific security profile templates are acceptable for their environments. If this applies to your situation, then determine which security profile is closest to the security posture you want, and use it for both assessing and hardening your systems.

Review and customize the security profile templates for your environment, or develop new ones. Techniques and guidelines for customizing security profiles are provided in the *Solaris Security Toolkit 4.1 Reference Manual*. This approach provides a security posture tailored for your organization, and it minimizes the amount of false errors returned during a security assessment. For example, if you know that Telnet needs to be enabled, you can customize the security profile so that when performing a security assessment, the software does not consider Telnet a vulnerability. For example, a site using Telnet with Kerberos, for authentication and encryption, would not consider the use of Telnet a vulnerability.

Using Options and Controlling Audit Output

This section describes the options available for executing an audit run and the options for controlling output. This section contains the following topics:

Command Line Options

Example usage to audit a system against a security profile:

```
# jass-execute -a driver [ -V [0-4]] [ -q | -o output-file ]
[ -m email-address ]
```

When executing the Solaris Security Toolkit software audit command, you can use the following options listed in TABLE 6-1.

TABLE 6-1 Using Command-Line Options With the Audit Command

Option	Description
`-a`	Determines if a system is in compliance with its security profile.
`-h`	Displays the `jass-execute` help message, which provides an overview of the available options.
`-m`	Mails output to an email address.
`-o`	Directs output into a file.
`-q`	Prevents the display of output to the console. Also known as the quiet option.
`-V`	Specifies the verbosity level for an audit run.

For detailed information about the options available with `jass-execute -a` command, see the following sections:

- "Display Help Option" on page 87
- "Email Notification Option" on page 88
- "Output File Option" on page 89
- "Quiet Option" on page 89
- "Verbosity Option" on page 89

Display Help Option

The `-h` option displays the `jass-execute` help message, which provides an overview of the available options.

The -h option produces output similar to the following:

CODE EXAMPLE 6-1 Sample -h Option Output

```
# ./jass-execute -h

To apply this Toolkit to a system, using the syntax:
  jass-execute [-r root_directory -p os_version ]
  [ -q | -o output_file ] [ -m e-mail_address ]
  [ -V [3|4] ] [ -d ] driver

To undo a previous application of the Toolkit from a system:
  jass-execute -u [ -b | -f | -k ] [ -q | -o output_file ]
     [ -m e-mail_address ] [ -V [3|4] ]

To audit a system against a pre-defined profile:
  jass-execute -a driver [ -V [0-4] ] [ -q | -o output_file ]
     [ -m e-mail_address ]

To display the history of Toolkit applications on a system:
  jass-execute -H

To display the last application of the Toolkit on a system:
  jass-execute -l

To display this help message:
  jass-execute -h
  jass-execute -?

To display version information for this program:
  jass-execute -v
```

Email Notification Option

The -m *email-address* option provides a mechanism by which output can be emailed automatically by the Solaris Security Toolkit software when the run completes. The email report is in addition to any logs generated on the system using other options.

A Solaris Security Toolkit run calling sunfire_15k_sc-config.driver using the email option would be similar to the following:

```
# ./jass-execute -m root -a sunfire_15k_sc-config.driver
[...]
```

Output File Option

The `-o` *output-file* option redirects the console output of `jass-execute` runs to a separate file, *output-file*.

This option has no effect on the logs kept in the `JASS_REPOSITORY` directory. This option is particularly helpful when performed over a slow terminal connection, because there is a significant amount of output generated by a Solaris Security Toolkit run.

This option can be used with either the `-d`, `-u`, or `-a` options.

The `-o` option produces output similar to the following:

CODE EXAMPLE 6-2 Sample `-o` Option Output

```
# ./jass-execute -o jass-output.txt -a secure.driver
[NOTE] Executing driver, secure.driver
[NOTE] Recording output to jass-output.txt
#
```

Quiet Option

The `-q` option disables Solaris Security Toolkit output to standard input/output (`stdio`) stream during a hardening run.

This option has no effect on the logs kept in the `JASS_REPOSITORY` directory. Similar to the `-o` option, this option is particularly helpful when running the Solaris Security Toolkit software through a cron job or over slow network connections.

This option can be used with either the `-d`, `-u`, or `-a` options.

The `-q` option produces output similar to the following:

CODE EXAMPLE 6-3 Sample `-q` Option Output

```
# ./jass-execute -q -a secure.driver
[NOTE] Executing driver, secure.driver
```

Verbosity Option

The `-V` option specifies the verbosity level for an audit run. This option is only available for auditing. Verbosity levels provide a highly flexible way of displaying the results of an audit run. For example, if you have 100 machines to audit, you

might want to limit the output to a single line for each machine to simply determine which machines pass or fail. Then, for the machines that fail, you might want to run an audit that produces expanded output, to focus on the problem areas.

The five verbosity levels (0 through 4) are controlled by the `-V` option. Each incremental level provides additional detail that you can use to more fully understand which checks are passing and which are failing. TABLE 6-2 describes the verbosity levels.

TABLE 6-2 Audit Verbosity Levels

Level	Output
0	Single line indicating pass or fail.
1	For each script, a single line indicating pass or fail. One grand total score line below all the script lines.
2	For each script, provides results of all checks.
3	Multiple lines providing full output, including banner and header messages.
4	Multiple lines (all data provided from level 3) plus all entries that are generated by the `logDebug` logging function. This level is for debugging.

Note – The default verbosity level for the `jass-execute -V` command is 3.

For complete descriptions of the verbosity levels, refer to the *Solaris Security Toolkit 4.1 Reference Manual*.

Banners and Messages Output

You can configure the Solaris Security Toolkit audit option to report or omit banners and messages. The `JASS_LOG_BANNER` variable cannot be used with verbosity levels 0-2. These output options apply to verbosity levels 3 and 4. For example, you might want to eliminate pass messages (`JASS_LOG_SUCCESS` variable) from the output so you can report and focus only on fail messages (`JASS_LOG_FAILURE` variable).

TABLE 6-3 lists the banners and messages that you can control through logging variables. (For detailed information about logging variables, refer to the *Solaris Security Toolkit 4.1 Reference Manual*.) If the logging variable is set to `0`, then no

output is generated for messages of that type. Conversely, if the logging variable is set to `1`, then messages are displayed. The default action for each of these variables is to display the output. TABLE 6-3 describes the logging variables.

TABLE 6-3 Displaying Banners and Messages in Audit Output

Logging Variable	Log Prefix	Description
`JASS_LOG_BANNER`	All Banner Output	This parameter controls the display of banner messages. These messages are usually surrounded by separators comprised of either equal sign (“=”) or dash (“-”) characters.
`JASS_LOG_ERROR`	`[ERR]`	This parameter controls the display of error messages. If set to 0, no error messages will be generated.
`JASS_LOG_FAILURE`	`[FAIL]`	This parameter controls the display of failure messages. If set to 0, no failure messages will be generated.
`JASS_LOG_NOTICE`	`[NOTE]`	This parameter controls the display of notice messages. If set to 0, no notice messages will be generated.
`JASS_LOG_SUCCESS`	`[PASS]`	This parameter controls the display of success or passing status messages. If set to 0, no success messages will be generated.
`JASS_LOG_WARNING`	`[WARN]`	This parameter controls the display of warning messages. If set to 0, no warning messages will be generated.

Using these options is very useful when you only need to view specific messages. By setting these options, you can minimize output, yet still focus on areas you deem critical. For example, by setting all logging variables to 0 except for `JASS_LOG_FAILURE` (leave it at the default of 1), the audit reports only on failures generated by the `logFailure` function.

CODE EXAMPLE 6-4 Sample Output of Reporting Only Audit Failures

```
# JASS_LOG_FAILURE=1
# export JASS_LOG_FAILURE
[setting of other parameters to 0 omitted]
# ./jass-execute -a secure.driver -V 2
update-at-deny        [FAIL] User test is not listed in
   /etc/cron.d/at.deny.
update-at-deny        [FAIL] Audit Check Total : 1 Error(s)
update-inetd-conf     [FAIL] Service ftp is enabled in
   /etc/inet/inetd.conf.
update-inetd-conf     [FAIL] Service telnet is enabled in
   /etc/inet/inetd.conf.
update-inetd-conf     [FAIL] Service rstatd is enabled in
   /etc/inet/inetd.conf.
update-inetd-conf     [FAIL] Audit Check Total : 3 Error(s)
```

Host Name, Script Name, and Timestamp Output

You can configure the Solaris Security Toolkit audit option to include host name, script name, and timestamp information for verbosity levels 0 – 2. For example, if you have many machines to audit, you may want to be able to sort the output by host name, script name, or timestamp. TABLE 6-4 lists the variables.

TABLE 6-4 Displaying Host Name, Script Name, and Timestamp Audit Output

Variable Name	Variable Description
JASS_DISPLAY_HOSTNAME	Setting this parameter to 1 causes the Solaris Security Toolkit software to prepend each log entry with the host name of the system. This information is based on the JASS_HOSTNAME parameter. By default, this parameter is empty, so the Toolkit does not display this information.
JASS_DISPLAY_SCRIPTNAME	By default, this parameter is set to 1, so the Solaris Security Toolkit software prepends each log entry with the name of the audit script currently being run. Setting this parameter to any other value causes the Toolkit to not display this information.
JASS_DISPLAY_TIMESTAMP	Setting this parameter to 1 causes the Solaris Security Toolkit software to prepend each log entry with the timestamp associated with the audit run. This information is based on the JASS_TIMESTAMP parameter. By default, this parameter is empty, so the software does not display this information.

By configuring the Solaris Security Toolkit software to prepend host, script, and timestamp information, you can combine many runs from either a single system or group of systems and sort them based on the key data. You can use the information to look for problems that span several systems or that are symptomatic of deployment processes. For example, using the information in this way, an administrator can tell if every system build using a given process always has the same failed checks.

For example, by setting the JASS_DISPLAY_TIMESTAMP parameter to 1 and setting the JASS_DISPLAY_SCRIPTNAME value at 0, output similar to the following would be generated.

CODE EXAMPLE 6-5 Sample Output of Auditing Log Entries

```
# JASS_DISPLAY_SCRIPTNAME=0
# JASS_DISPLAY_TIMESTAMP=1
# export JASS_DISPLAY_SCRIPTNAME JASS_DISPLAY_TIMESTAMP
# ./jass-execute -a secure.driver -V 2
20030101233525 [FAIL] User test is not listed in
   /etc/cron.d/at.deny.
20030101233525 [FAIL] Audit Check Total : 1 Error(s)
20030101233525 [FAIL] Service ftp is enabled in
   /etc/inet/inetd.conf.
20030101233525 [FAIL] Service telnet is enabled in
   /etc/inet/inetd.conf.
20030101233525 [FAIL] Service rstatd is enabled in
   /etc/inet/inetd.conf.
20030101233525 [FAIL] Audit Check Total : 3 Error(s)
```

Performing a Security Audit

Performing a security assessment periodically on your systems provides a benchmark of how closely the security matches the security profile you implemented. The most common scenario for performing security assessments is as a security maintenance task sometime after hardening new installations. We designed the security assessment option so that you simply execute the same hardening driver(s) that you used to harden the system, but that now you use the `-a` option to check the current state compared to the security profile implemented during hardening. This design eliminates complexity and provides flexibility. For example, when you update your security profile, subsequent security assessments use the updated security profile.

In another possible scenario, you might be responsible for securing systems that are already deployed. Before you harden them, you want to perform a security assessment. In this scenario, you would define your own security profile, customize a Solaris Security Toolkit security profile template, or use one of the security profile templates as is.

▼ To Perform a Security Audit

Before performing an audit, you need to define or choose a security profile. For more information, see "Preparing to Audit Security" on page 86.

Caution – If you are performing a security assessment on a deployed system that you did not harden previously, first back up the machine and reboot it to verify that it is in a known, working, and consistent configuration. Any errors or warnings detected during this preliminary reboot should be corrected or noted before proceeding with security assessment.

1. **Choose the security profile (hardening driver) that you want to use:**
   - If you hardened the system previously, use the same security profile.

     For example, `secure.driver`.
   - If you have not hardened the system, use one of the standard security profiles or your own.

     For example, `secure.driver` or `abccorp-secure.driver`.

   For a complete and up-to-date listing of available drivers, download the most recent version of the Solaris Security Toolkit software from the following Web site:

   `http://www.sun.com/security/jass`

   Refer to the *Solaris Security Toolkit 4.1 Reference Manual* for information about standard and product-specific drivers. For the most current listing of drivers, see the Drivers directory.

2. **Determine the command line options you want and how you want to control the output.**

   See "Using Options and Controlling Audit Output" on page 86.

3. **Enter the `jass-execute -a` command, the name of the security profile, and the options you want.**

   The following is a sample audit run using the `sunfire_15k_sc-secure.driver`.

CODE EXAMPLE 6-6 Sample Output of Audit Run

```
# ./jass-execute -a sunfire_15k_sc-secure.driver
[NOTE] Executing driver, sunfire_15k_sc-secure.driver

[...]

======================================================================
sunfire_15k_sc-secure.driver: Audit script: enable-rfc1948.aud
======================================================================

#----------------------------------------------------------------
# RFC 1948 Sequence Number Generation
#
# Rationale for Audit:
#
# The purpose of this script is to audit that the system is
# configured and is in fact using RFC 1948 for its TCP sequence
# number generation algorithm (unique-per-connection ID). This is
# configured by setting the 'TCP_STRONG_ISS' parameter to '2' in
# the /etc/default/inetinit file.
#
# Determination of Compliance:
#
[...]
#----------------------------------------------------------------

[PASS] TCP_STRONG_ISS is set to '2' in /etc/default/inetinit.
[PASS] System is running with tcp_strong_iss=2.

# The following is the vulnerability total for this audit script.

[PASS] Audit Check Total : 0 Error(s)

======================================================================

# The following is the vulnerability total for this driver profile.

[PASS] Driver Total : 0 Error(s)

======================================================================
sunfire_15k_sc-secure.driver: Driver finished.
```

CODE EXAMPLE 6-6 Sample Output of Audit Run *(Continued)*

```
================================================================

[PASS] Grand Total : 0 Error(s)
```

When an audit run is initiated, the Solaris Security Toolkit software accesses files from the `JASS_HOME_DIR/Audit` directory. Although the files in both the `JASS_HOME_DIR/Audit` and `JASS_HOME_DIR/Finish` directories share the same base file names, they have different file name suffixes. The `driver.run` script automatically translates the finish scripts defined by the `JASS_SCRIPTS` variable into audit scripts, by changing their suffixes from `.fin` to `.aud`.

The audit run starts and initializes the state of the Solaris Security Toolkit software. Each driver that is accessed during the run evaluates the state of all of its file templates and audit scripts. Each check results in a state of success or failure, represented by a vulnerability value of either zero or nonzero, respectively. In most cases, failure is represented by a number 1. Each script that is run produces a total security score, based on the total vulnerability value of each check contained within a script. Furthermore, the total vulnerability value result for each driver is displayed at the completion of a driver's assessment. A grand total of all scores is presented at the end of the run.

The security assessment option provides a comprehensive view of the state of a system at the time the assessment run is initiated. The Solaris Security Toolkit software checks the stored state of the system by inspecting configuration files and checks the running state of the system by inspecting process table information, device driver information, and so on. The Solaris Security Toolkit software checks for the existence of each file or service, and checks if the software associated with a service is installed, configured, enabled, and running. This approach yields an accurate snapshot of the current state of a system.

CHAPTER 7

Securing a System

This chapter describes how to apply the information and expertise provided in earlier chapters to a realistic scenario for installing and securing a new system. This chapter illustrates how to deploy the Solaris Security Toolkit software with a Check PointFirewall-1 NG for the Solaris 8 OS.

Use the information in this chapter as a guide and case scenario for securing a new system and applications.

Sun BluePrint books and online articles are available to guide you through the process of minimizing and hardening many of Sun's systems. Refer to the following Web site for the latest product-specific books and articles:

`http://www.sun.com/blueprints`

This chapter contains the following topics:

Planning and Preparing

To effectively and efficiently deploy minimized and secured systems as described in this case study, planning and preparation is critical. The underlying network infrastructure, policies, and procedures must be in place. In addition, support and maintenance of the systems must be defined and communicated. For more information about planning and preparing, see Chapter 2. The scenario described in

this chapter documents the process and tasks that a system administrator (SA) would perform to achieve a minimized and hardened Solaris OS image for a firewall system.

In this scenario, the SA is tasked with creating an automated and scalable solution for building and deploying Check PointFirewall-1 NG systems for a service provider (xSP) that wants to provide firewall service to its customers. For this scenario, xSP's requirements and considerations are as follows:

- Because xSP plans to deploy many of these systems, the time to build and deploy each system is critical and must be streamlined for efficiency.
- Systems are installed using a dedicated management network connected to the internal Ethernet interface of each system. This network is only used by xSP staff and not by subscribers.
- All other interfaces are on separate physical network interfaces and are filtered.
- The security of the management network is critical to the overall security of the deployed firewall systems.

Based on these requirements, the SA decides to automate the installation, minimization, and hardening of the OS images by using JumpStart technology and the Solaris Security Toolkit software.

Assumptions and Limitations

This chapter assumes we are using an already working Solaris Security Toolkit software and JumpStart technology installation. Other chapters in this book provide instructions and guidelines for installing the software; see applicable chapters for that information.

This chapter assumes we are developing a custom configuration for minimizing and hardening a specific application. The Solaris Security Toolkit software does not have any drivers or JumpStart profiles specifically for the application. Therefore, we need to create custom drivers and profiles for this application. This task is done by copying existing drivers and profiles, then modifying them to fit the application.

For this case scenario, the SA's skill level is the following:

- Has enough knowledge and experience to configure the OS and applications.
- Knows how to test the configuration and make adjustments to fine tune it.
- Knows how to build a JumpStart environment from which the client system is installed. Refer to the Sun BluePrint book *JumpStart Technology: Effective Use in the Solaris Operating Environment*.
- Is familiar with OS minimization techniques. Refer to *Enterprise Security: Solaris Operating Environment Security Journal, Solaris Operating Environment Versions 2.5.1, 2.6, 7, and 8*.

- Is familiar with the Solaris Security Toolkit software basics and is ready to focus on building a customized configuration using both minimization and hardening techniques and guidelines. See Chapter 1.

System Environment

The example scenario is based on the following hardware and software environment:

- Check PointFirewall-1 NG
- Solaris 8 OS
- JumpStart technology
- Solaris OS cluster (`SUNWCreq`)
- Solaris Security Toolkit software
- Platform based on SPARC technology
- At least two Ethernet interfaces

Security Requirements

For this scenario, the high-level requirements and software packages have been identified, but the specific components and services of all the packages need to be identified. Also, the Solaris OS capabilities needed to administer and manage the systems must be identified.

The following list provides a detailed view on how software components are used:

- Secure Shell for remote administration
- FTP to copy files
- Solstice DiskSuite™ software to mirror disks
- SYSLOG messages forwarded to a central repository

From this list, you can develop a security profile. For detailed information about developing security profiles and using the profile templates, see "Developing and Implementing a Solaris Security Toolkit Profile" on page 29.

Creating a Security Profile

A security profile defines what security modifications the Solaris Security Toolkit software makes when hardening and minimizing the security configuration of a system. None of the standard security profiles or drivers included in the Solaris

Security Toolkit software meet the requirements for the minimized Check PointFirewall-1 NG systems. Therefore, you must create a custom security profile to implement the appropriate system modifications.

For this scenario, the process for creating a security profile is described in several sections in this chapter, where appropriate for the scenario. First, we create new driver files based on existing drivers. Then we modify the new drivers to comply with the security requirements outlined previously. Minimization is described in "Installing the Software" on page 102, and the hardening modifications in "Customizing the Hardening Configuration" on page 109.

Installing the Software

This section demonstrates the process of installing the software. For the sample scenario, we provide any exceptions or scenario-specific instructions. For general instructions about installing software, use the references to other parts of this guide.

Note – You may use the instructions in the following as a template for handling related situations.

This section contains the following tasks:

- "Downloading and Installing Security Software" on page 102
- "Installing Patches" on page 103
- "Specifying and Installing the OS Cluster" on page 104

Downloading and Installing Security Software

Download and install the Solaris Security Toolkit and additional security software, including patches, on the JumpStart server as follows.

▼ To Download and Install the Security Software

1. **Download the Solaris Security Toolkit software and additional security software.**

   See "Downloading Security Software" on page 38.

2. **Install the downloaded Solaris Security Toolkit software and additional security software.**

   See "Installing and Executing the Software" on page 46.

Caution – Do not execute the Solaris Security Toolkit software yet. First perform the additional configuration and customizing described in the following sections.

Installing Patches

OS patches may address security vulnerabilities, availability issues, performance concerns, or other aspects of a system. When installing a new OS, and on an ongoing basis after the OS is installed, check to ensure that appropriate patches are installed.

The Solaris Security Toolkit software provides a mechanism to install the Recommended and Security Patch Cluster available from SunSolve Online. This OS-specific cluster of patches includes the most commonly needed patches.

▼ To Install Patches

1. **At a minimum, download the Recommended and Security Patch Cluster into the** `Patches` **directory and uncompress it.**

   If the `install-recommended-patches.fin` script is included in the hardening driver, then that patch cluster is installed automatically.

   There is an added issue for Check PointFirewall-1 NG. This application requires specific patches not included in the Recommended and Security Patch Cluster. The Check PointFirewall-1 NG requires the following patches:

   - 108434
   - 108435

2. **To automate the installation of patches 108434 and 108435, download the latest versions from SunSolve OnLine, and place them in the** `Patches` **directory.**

3. **Create a new finish script (for example,** `fw1-patch-install.fin`**) that calls the** `add_patch` **helper function, with the name of each patch.**

   This finish script calls the appropriate helper functions with the two Check PointFirewall-1 NG required patch IDs. For example:

```
# !/bin/sh

# add_patch 108434-10

# add_patch 108435-10
```

Specifying and Installing the OS Cluster

After defining a disk layout for the OS installation, the next task is to specify which Solaris OS cluster to install. Choose one of the five installation clusters available with Solaris OS: `SUNWCreq`, `SUNWCuser`, `SUNWCprog`, `SUNWCall`, and `SUNWCXall`.

▼ To Specify and Install the OS Cluster

1. **Specify the OS cluster to install.**

   Because the goal of this case scenario is to build a minimized and dedicated firewall device, the smallest of the available Solaris OS clusters, `SUNWCreq`, this package is also known as Core.

   Because this cluster includes a relatively small number of packages, other packages are probably required. These other required packages need to be included in the profile with the Solaris OS cluster definition.

   The baseline profile definition adds the following to the previously defined profile.

   ```
   cluster           SUNWCreq
   ```

   The `SUNWCreq` installation cluster includes packages that are not required for a firewall Sun server to function properly. Remove these extra packages after you have a working baseline. Refer to Sun BluePrints OnLine article "Minimizing the Solaris Operating Environment for Security: Updated for the Solaris 9 Operating Environment."

2. **Run through an installation with the appropriately defined security profile to determine if there are any package dependency issues.**

   Some package dependencies are encountered during installation, and we determine that the following Solaris OS packages are required for Check PointFirewall-1 NG:

   - `SUNWter` – Terminal information
   - `SUNWadmc` – System administration core libraries
   - `SUNWadmfw` – System and network administration framework

- `SUNWlibC` and `SUNWlibCx` – Required for the Check PointNG application

The complete listing of packages in the profile is as follows.

```
cluster           SUNWCreq
package           SUNWter          add
package           SUNWlibC         add
package           SUNWlibCx        add
package           SUNWadmc         add
package           SUNWadmfw        add
```

Although this list is complete for this case study, additional packages may be added or removed based on the actual environment into which this configuration is deployed.

Until the system is verified from both a function and security perspective, as described in "Testing for Quality Assurance" on page 115, the final list of packages might require modification. If so, modify the profile, reinstall the system, and repeat the testing.

3. **Create a `minimize-firewall.fin` `script`, based on the package dependencies in the previous two steps.**

Configuring the JumpStart Server and Client

This section demonstrates how to configure the JumpStart server and client to use a custom security profile for minimization. For detailed information about using the the Solaris Security Toolkit software in JumpStart environments, see Chapter 5.

This section contains the following tasks:

Preparing the Infrastructure

Perform the following tasks to prepare the infrastructure. The following tasks demonstrate the process of creating a baseline configuration for the client using existing drivers, profiles, and finish scripts. After this baseline is in place, verify that it works properly, then customize it for the chosen application.

▼ To Prepare the Infrastructure

1. **Configure your JumpStart server and environment.**

   See Chapter 5 for detailed instructions.

2. **Add the client to the JumpStart server by using the `add-client` command.**

   CODE EXAMPLE 7-1 Adding a Client to the JumpStart Server

```
# pwd
/jumpstart
# bin/add-client -c jordan -o Solaris_8_2002-02 -m sun4u
-s nomex-jumpstart
cleaning up preexisting install client "jordan"
removing jordan from bootparams
updating /etc/bootparams
```

3. **Create a `rules` file entry for the client, specifying the appropriate JumpStart profile and finish script. For example:**

```
hostname jordan - Profiles/xsp-minimal-firewall.profile \
  Drivers/xsp-firewall-secure.driver
```

4. **Create a profile file named `xsp-minimal-firewall.profile` and a driver file named `xsp-firewall-secure.driver` by copying the files provided with the Solaris Security Toolkit software.**

   You must create these files before you can successfully complete the next step. Initially, these files can be copies of files distributed with the Solaris Security Toolkit software. Never modify the original files distributed with the Solaris Security Toolkit software. The following example shows how to create the files.

   CODE EXAMPLE 7-2 Creating a Profile

```
# pwd
/jumpstart/Drivers
# cp install-Sun_ONE-WS.driver xsp-firewall-secure.driver
# cp hardening.driver xsp-firewall-hardening.driver
[...]
# pwd
/jumpstart/Profiles
# cp minimal-Sun_ONE-WS-Solaris8-64bit.profile \
     xsp-minimal-firewall.profile
```

   This example is based on a dedicated web server configuration, because it is a good baseline from which to develop a dedicated firewall.

5. **After creating the profile and driver files, modify the files as follows:**

   a. **Replace the** `xsp-firewall-secure.driver` **reference to** `hardening.driver` **with** `xsp-firewall-hardening.driver`**.**

   b. **Replace the two finish scripts defined in** `JASS_SCRIPTS` **with references to** `minimize-firewall.fin` **and your finish script (for example,** `fw1-patch-install.fin`**).**

   The modified script should appear similar to the following.

CODE EXAMPLE 7-3 Sample Output of Modified Script

```
DIR="`/bin/dirname $0`"
export DIR
. ${DIR}/driver.init
. ${DIR}/config.driver
JASS_SCRIPTS="
                minimize-firewall.fin
                fw1-patch-install.fin"
. ${DIR}/driver.run
. ${DIR}/xsp-firewall-hardening.driver
```

6. **Check the** `rules` **file entry for correctness using the following command.**

CODE EXAMPLE 7-4 Checking the `rules` File for Correctness

```
# pwd
/jumpstart
# ./check
Validating rules...
Validating profile Profiles/end-user.profile...
Validating profile Profiles/xsp-minimal-firewall.profile...
Validating profile Profiles/test.profile...
Validating profile Profiles/entire-distribution.profile...
Validating profile Profiles/oem.profile...
The custom JumpStart configuration is ok.
```

At this point, it should be possible to begin the JumpStart installation on the client, `jordan` in this example. Use the JumpStart configuration and Solaris Security Toolkit drivers, finish scripts, and profiles that you created.

7. **If you encounter problems when checking the** `rules` **file, see** **"Validating and Checking the Rules File" on page 108.**

8. **From the client's `ok` prompt, enter the following command to install the client using the JumpStart infrastructure.**

```
ok> boot net - install
```

If the client does not build, review the configuration and modify it until it works properly. Note that all aspects of the JumpStart configuration are not addressed in this section. Refer to the Sun BluePrint book *JumpStart Technology: Effective Use in the Solaris Operating Environment* for more details.

After achieving a correct run of the `rules` file and verifying that patches were installed correctly, you can start the base-level installation of the client system and its minimization and hardening.

Validating and Checking the Rules File

When validating the `rules` file for correctness, you might encounter a variety of problems. Some of the most common are addressed in this section.

The first run on the `rules` file results in the following output.

CODE EXAMPLE 7-5 Sample Output for `rules` File

```
# pwd
/jumpstart
# ./check
Validating rules...
Validating profile Profiles/xsp-minimal-firewall.profile...
Error in file "rules", line 20
hostname jordan - Profiles/xsp-minimal-firewall.profile
Drivers/xsp-firewall-secure.driver
ERROR: Profile missing:
   Profiles/xsp-minimal-firewall.profile
```

In this example, the profile specified in the `rules` entry for `jordan` does not exist. The profile, `xsp-minimal-firewall.profile`, was not present in the profiles directory. Typically, this error is generated due to a spelling mistake in the file name, forgetting to specify the correct directory for the profiles, or simply not having created the profile yet. Fix the problem and rerun the check.

The second run uncovers two other problems. The first problem is the driver being called in the `xsp-firewall-secure.driver`. Instead of calling `xsp-firewall-hardening.driver`, the `xsp-firewall-secure.driver` is still calling the `hardening.driver`.

The second problem is that the JASS_SCRIPTS variable is incorrectly set to minimize-Sun_ONE-WS.fin instead of minimize-firewall.fin.

The following is the incorrect script.

CODE EXAMPLE 7-6 Sample of Incorrect Script

```
#!/bin/sh
DIR="`/bin/dirname $0`"
export DIR
. ${DIR}/driver.init
. ${DIR}/config.driver
JASS_SCRIPTS="minimize-Sun_ONE-WS.fin"
. ${DIR}/driver.run
. ${DIR}/hardening.driver
```

The following is an example of a correct script.

CODE EXAMPLE 7-7 Sample of Correct Script

```
#!/bin/sh
DIR="`/bin/dirname $0`"
export DIR
. ${DIR}/driver.init
. ${DIR}/config.driver
JASS_SCRIPTS="
minimize-firewall.fin"
. ${DIR}/driver.run
. ${DIR}/xsp-firewall-hardening.driver
```

Customizing the Hardening Configuration

The hardening configuration of the proposed firewall is ready to be customized and fine-tuned. The initial scripts are based on the hardening.driver. This means that the system is turned into a “warm-brick,” that is, all of its services are disabled.

Because Solaris 8 OS does not include a Secure Shell client, you need to make modifications to allow for remote, network-based administration of the firewalls. For the firewall in this case scenario, the requirements specify that FTP services must remain enabled and a Secure Shell client must be installed for remote administration.

Restrict both of these services to the private management network only, thereby not enabling listening on any other network interfaces. For information about restricting these services, refer to the Sun BluePrints OnLine article titled "Solaris Operating Environment Security: Updated for Solaris 9 Operating Environment."

In addition to leaving these two services enabled, leave RPC services enabled so that we can use the Solstice DiskSuite graphical user interface (GUI) to configure Solstice DiskSuite for disk mirroring. If the Solstice DiskSuite GUI is not going to be used, then the RPC services are not needed. In this example, the GUI is required and therefore RPC services are left enabled. Note that installation and configuration of Solstice DiskSuite is beyond the scope of this book.

The final modification required for this client is that a customized `syslog.conf` is crafted that uses xSP's centralized SYSLOG server. This customized `syslog.conf` file must be installed on each of the firewall systems.

These modifications require changes to a variety of Solaris Security Toolkit configuration options. Each of the required modifications is detailed in the following sections.

Enabling FTP Service

For the firewall in this case scenario, leave the FTP services enabled.

▼ To Enable FTP Service

1. **To leave FTP enabled, modify the default behavior of the `update-inetd-conf.fin` file by setting the `JASS_SVCS_DISABLE` and `JASS_SVCS_ENABLE` variables.**

   To disable all standard Solaris OS services except for FTP, the best method for our case scenario is to define `JASS_SVCS_ENABLE` to be `ftp` while ensuring that `JASS_SVCS_DISABLE` is left with its default value obtained from the `finish.init` script. Refer to the *Solaris Security Toolkit 4.1 Reference Manual*.

2. **To implement the change through the environment variables, add an entry similar to the following to** `xsp-firewall-secure.driver` **before the call to** `xsp-firewall-hardening.driver`**.**

```
JASS_SVCS_ENABLE="ftp"
```

3. **Ensure that FTP is available only on xSP's management network by implementing it through the firewall software.**

   One of the other requirements was that FTP should be available only on xSP's management network. On Solaris 8 OS, you can implement this requirement either through incorporating TCP wrappers onto the system or through the firewall software itself. In this case scenario, implement it through the firewall software.

Installing Secure Shell Software

Because Solaris 8 OS does not include a Secure Shell client, install a Secure Shell client for remote administration.

You can configure the Solaris Security Toolkit software to install the OpenSSH tool. Use the `install-openssh.fin` script, which is listed in the `config.driver` file used by `xsp-firewall-secure.driver`.

▼ To Install Secure Shell

1. **Copy the default** `config.driver` **to** `xsp-firewall-config.driver`**.**
2. **In the copy of the file, uncomment the entry for** `install-openssh.fin`**.**
3. **Modify the entry in** `xsp-firewall-secure.driver` **that calls** `config.driver` **to call** `xsp-firewall-config.driver` **instead.**
4. **Obtain the latest version of OpenSSH.**

   As with patches and OS releases, use the most recent version of OpenSSH. Refer to the OpenSSH web pages for the latest release information:

   `http://www.openssh.org`

5. **Compile the latest OpenSSH package, name it appropriately, and install it in the** `Packages` **directory.**

   For more information about this package, refer to the Sun BluePrints OnLine article titled "Configuring OpenSSH for the Solaris Operating Environment."

6. **Update the `install-openssh.fin` script to reflect the correct OpenSSH package name.**

   Modifications to the `install-openssh.fin` script might be required. This script defines the package name of the OpenSSH package to be formatted similar to the following.

```
OBSDssh-3.5p1-sparc-sun4u-5.8.pkg
```

   Where the package name follows the version number (`3.5p1`), the architecture (`sparc`), the version of the architecture (`sun4u`), the OS for which the package was compiled (`5.8`), and a `pkg` suffix.

7. **Ensure that SSH is only available on xSP's management network by implementing it through the firewall software.**

   One of the other requirements stated was that Secure Shell should be available only on xSP's management network. With Solaris 8 OS, you can implement this requirement either by incorporating TCP wrappers onto the system or through the firewall software itself. In this case scenario, we implement it through the firewall software. Note that this requirement could also be implemented by modifying the Secure Shell server's configuration.

Enabling RPC Service

Leave RPC services enabled so that you can use SDS for disk mirroring, which requires RPC.

This modification is relatively straightforward because a specific finish script, `disable-rpc.fin`, is available to disable RPC services during a Solaris Security Toolkit run.

Note – Remotely accessing RPC services on a system should be explicitly denied by the system's firewall configuration.

▼ To Enable RPC

- **Comment out the entry for** `disable-rpc.fin` **in the** `xsp-firewall-hardening.driver`**.**

  Disable scripts from drivers by commenting them out instead of removing them. Be careful when commenting out entries in the `JASS_SCRIPTS` definition, because only certain combinations of comment values are accepted.

  The following is the comment, contained in the `driver.funcs script`, on what the Solaris Security Toolkit software accepts as comment indicators in the `JASS_SCRIPTS` definition.

```
#Very rudimentary comment handler. This code will only recognize
#comments where a single '#' is placed before the file name
#(separated by white space or not). It then will only skip the
#very next argument.
```

Customizing the `syslog.conf` File

The final modification required for this client is that a customized `syslog.conf` is crafted that uses xSP's centralized SYSLOG server. This customized `syslog.conf` file must be installed on each of the firewall systems.

▼ To Customize the `syslog.conf` File

1. **Copy the xSP standard** `syslog.conf` **file, rename it** `syslog.conf.jordan`**, then place it in the** `Files/etc` **directory.**

   The Solaris Security Toolkit software supports several different modes of copying files. The most appropriate option for this configuration is to append the system's host name as a suffix to the file so that the `syslog.conf` file is only copied to `jordan`, because it has unique firewall-specific modifications. In this case, the client is called `jordan`, so the actual filename used in `Files/etc` is `syslog.conf.jordan`. It is important to note that the `JASS_FILES` definition must not have this suffix appended. For more information about suffixes, refer to the *Solaris Security Toolkit 4.1 Reference Manual*.

2. **If the xSP standard** `syslog.conf` **file is not available, create a custom** `syslog.conf` **file as follows:**

   a. **Copy the** `syslog.conf` **file included with the Solaris Security Toolkit software, then rename it** `syslog.conf.jordan`**, and place it in the** `Files/etc` **directory.**

   b. **Modify the** `syslog.conf.jordan` **to conform to the xSP standard for** `SYSLOG`**.**

3. **Verify that the** `/etc/syslog.conf` **file is listed in the** `JASS_FILES` **definition of the** `xsp-firewall-hardening.driver`**.**

   By default, the modified `JASS_FILE` definition in `xsp-firewall-hardening.driver` appears as follows.

CODE EXAMPLE 7-8 Sample Output of Modified `xsp-firewall-hardening.driver`

```
JASS_FILES="
                    /etc/dt/config/Xaccess
                    /etc/init.d/inetsvc
                    /etc/init.d/nddconfig
                    /etc/init.d/set-tmp-permissions
                    /etc/issue
                    /etc/motd
                    /etc/notrouter
                    /etc/rc2.d/S00set-tmp-permissions
                    /etc/rc2.d/S07set-tmp-permissions
                    /etc/rc2.d/S70nddconfig
                    /etc/syslog.conf
"
```

At this point, all of the required modifications have been made. The installation of the OS, minimization, and hardening are customized for a specific application and fully automated. The only processes not fully automated are the configuration and installation of the firewall software and Solstice DiskSuite. Although it is possible to perform these configurations by using JumpStart technology, it is beyond the scope of this book. Refer to the Sun BluePrints book *JumpStart Technology: Effective Use in the Solaris Operating Environment*.

Installing the Client

After making all of the modifications to the drivers, install the client as described in this section.

▼ To Install the Client

1. **After all of the required modifications are made to the drivers, install the client using the JumpStart infrastructure.**

   Use the following command from the client's `ok` prompt.

```
ok> boot net - install
```

2. **If any errors are encountered, fix them and reinstall the client's OS.**

Testing for Quality Assurance

The final task in the process involves verifying that the applications and services offered by the system are functioning correctly. Also, this task verifies that the security profile successfully implemented the required modifications.

It is important that this task be done thoroughly and soon after the reboot of the now hardened and minimized platform, to ensure that any anomalies or problems are detected and corrected rapidly. This process is divided into two tasks: Verifying profile installation and verifying application and service functionality.

▼ To Verify Profile Installation

To verify that the Solaris Security Toolkit software installed the security profile correctly and without error, review, and evaluate the following.

1. **Review the installation log file.**

   This file is installed in `JASS_REPOSITORY/jass-install-log.txt`.

Note – This log file can be used as a reference to understand exactly what the Solaris Security Toolkit software did to the system. For each run on a system, there is a new log file stored in a directory based on the start time of the run. These files, and any other files in the `JASS_REPOSITORY` directory, must never be modified directly.

2. **Use the audit option to assess the security configuration of the system.**

   For detailed information about the audit option, see Chapter 6. For this scenario, we use the following command from the directory into which the Solaris Security Toolkit software was installed on the client.

CODE EXAMPLE 7-9 Assessing a Security Configuration

```
# ./jass-execute -a xsp-firewall-secure.driver
[NOTE] Executing driver, xsp-firewall-secure.driver
================================================================
===
xsp-firewall-secure.driver: Driver started.
================================================================
===

================================================================
===
Solaris Security Toolkit Version:   4.1.0
[...]
```

If the Solaris Security Toolkit verification run encounters any inconsistencies, they are noted. A summary at the end of the run reports on the total number of inconsistencies found. The entire output of the run is in the `JASS_REPOSITORY` directory.

▼ To Verify Application and Service Functionality

The verification process for applications and services involves the execution of a well-defined test and acceptance plan. This plan is used to exercise the various components of a system or application to determine that they are in an available and in working order. If such a plan is not available, test the system in a reasonable way based on how it is used. The goal is to ensure that the hardening process in no way affected the ability of applications or services to perform their functions.

1. **If you discover that an application or service malfunctions after the system is hardened, use the techniques described in Chapter 2 to determine the problem.**

   For example, use the `truss` command. This command can often be used to determine at what point an application is having difficulty. Once this is known, the problem can be targeted and traced back to the change made by the Solaris Security Toolkit software.

   Note – Based on the collective experience of many who have deployed the Solaris Security Toolkit software, the majority of problems can be avoided using the approach in this book.

   ---

2. **In a similar fashion, test the Check PointFirewall-1 NG software, trace any issues back to Solaris Security Toolkit software modifications, and correct the issues.**
3. **If the final list of packages requires modification, modify the profile, reinstall the system, and repeat the testing.**

Glossary

This list defines abbreviations and acronyms in the Solaris Security Toolkit.

A

ab2 AnswerBook2

ABI Application Binary Interface

ARP Address Resolution Protocol

ASPPP Asynchronous Point-to-Point Protocol

B

BIND Berkeley Internet Name Domain

BSD Berkeley Software Distribution

BSM Basic Security Model *(Solaris)*

C

CD compact disc

CD-ROM compact disc–read-only memory

CDE Common Desktop Environment

`cp`**(1)** copy files

`cron`**(1M)** clock daemon

D

DHCP Dynamic Host Configuration Protocol

DMI Desktop Management Interface

DMTF Distributed Management Task Force

DNS Domain Name System

E

EEPROM electronically erasable programmable read-only memory

F

FTP File Transfer Protocol

G

GID group identifier

H

HTTP HyperText Transfer Protocol

I

ID identifier

IETF Internet Engineering Task Force

INETD Internet service daemon

IP Internet Protocol

ISA instruction set architecture

J

JASS JumpStart Architecture and Security Scripts, *now* Solaris Security Toolkit

K

KDC Kerberos Key Distribution

L

LDAP Lightweight Directory Access Protocol

`lp`**(1)** line printer *(submit print request)*

M

MAN management network *(Sun Fire High-End Systems internal I1 network)*

MD5 message-digest 5 algorithm

MIP Mobile Internet Protocol

MSP midframe service processor

`mv`**(1)** move files

N

NFS Network File System

NG Next Generation

NIS, NIS+ Network Information Services

NSCD name service cache daemon

O

OE operating environment, *formerly used for Solaris*

OEM Original Equipment Manufacturer

OS Operating System, *now used for Solaris*

P

PAM Pluggable Authentication Module

PDF Portable Document Format

PICL Platform Information and Control Library

PPP Point-to-Point Protocol

PROM programmable read-only memory

Q

QA quality assurance

R

RBAC role-based access control

rc run-control *(file or script)*

`rlogin`**(1)** remote login

RFC Remote Function Call

RPC Remote Procedure Call

`rsh`**(1)** remote shell

S

SA system administrator

SC system controller *(Sun Fire High-End and Midrange Systems)*

`scp`**(1)** secure copy (remote file copy program)

SCCS Source Code Control System

SLP Service Location Protocol

SMA System Management Agent

SMC Solaris Management Console

SNMP Simple Network Management Protocol

SP service provider

SPARC Scalable Processor Architecture

SPC SunSoft Print Client

SSH Secure Shell *(Solaris)*

SSP system service processor *(Sun Enterprise 1000 Servers)*

`stdio` standard input/output

Sun ONE Sun Open Network Environment, *currently* Sun Java System, *formerly* iPlanet

T

TCP Transmission Control Protocol

`tftp`**(1)** trivial file transfer program

`ttl` time-to-live

U

U.S. United States

UDP User Diagram Protocol

UID user identifier

UUCP UNIX-to-UNIX Copy

V

VOLD Volume Management daemon

W

WBEM Web-based Enterprise Management

Index

B

C

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z