System Administration Guide: Security Services

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 817–0365–10
August 2003

030312@5533

Contents

Preface

System Administration Guide: Security Services is part of a multivolume set that covers a significant part of the Solaris™ system administration information. This book assumes that you have already installed the SunOS™ 5.9 operating system. This book also assumes that you have set up the networking software that you plan to use. The SunOS 5.9 operating system is part of the Solaris 9 product family. The Solaris 9 product family includes many features, such as the Solaris Common Desktop Environment (CDE).

Note – The Solaris operating environment runs on two types of hardware, or platforms – the SPARC® platform and the x86 platform. The Solaris operating environment runs on a 64-bit address space and a 32-bit address space. The information in this document pertains to both platforms and to both address spaces. Exceptions are called out in a special chapter, section, note, bullet, figure, table, example, or code example.

Who Should Use This Book

This book is intended for anyone who is responsible for administering one or more systems that run the Solaris 9 release. To use this book, you should have one year to two years of UNIX® system administration experience. Your attending training courses in UNIX system administration might be helpful.

How the System Administration Volumes Are Organized

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

Book Title	Topics
System Administration Guide: Basic Administration	User accounts and groups, server and client support, shutting down and booting a system, removable media, managing software (packages and patches), disks and devices, file systems, and backing up and restoring data
System Administration Guide: Advanced Administration	Printing services, terminals and modems, system resources (disk quotas, accounting, and crontabs), system processes, and troubleshooting Solaris software problems
System Administration Guide: IP Services	TCP/IP networks, IPv4 and IPv6, DHCP, IP Security, Mobile IP, and IP Network Multipathing
System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)	DNS, NIS, and LDAP naming and directory services
System Administration Guide: Naming and Directory Services (FNS and NIS+)	FNS and NIS+ naming and directory services
System Administration Guide: Resource Management and Network Services	Resource management, remote file systems, mail, SLP, and PPP
System Administration Guide: Security Services	Auditing, Device allocation, PAM, RBAC, Solaris Secure Shell, and SEAM

Related Books

The following is a list of related documentation that is referred to in this book.

- Carasik, Anne. *UNIX Secure Shell*. McGraw Hill, 1999.
- Cheswick, William R. and Steven M. Bellovin. *Firewalls and Internet Security*. Addison Wesley, 1994.

Accessing Sun Documentation Online

The docs.sun.com[SM] Web site enables you to access Sun technical documentation online. You can browse the docs.sun.com archive or search for a specific book title or subject. The URL is `http://docs.sun.com`.

Typographic Conventions

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

TABLE P–1 Typographic Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories On-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`
`AaBbCc123`	What you type, contrasted with on-screen computer output	`machine_name%` **`su`** `Password:`
AaBbCc123	Command-line placeholder that you replace with a real name or value	To delete a file, type **`rm`** *filename*.
AaBbCc123	Book titles, new words, or terms, or words to be emphasized.	Read Chapter 6 in *User's Guide*. These are called *class* options. You must be *root* to do this.

Shell Prompts in Command Examples

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P–2 Shell Prompts

Shell	Prompt
C shell prompt	`machine_name%`
C shell superuser prompt	`machine_name#`
Bourne shell and Korn shell prompt	`$`
Role shell prompt	`$`
Bourne shell and Korn shell superuser prompt	`#`

PART I Security Overview

This book focuses on the features that enhance security in the Solaris™ operating environment. This book is intended for system administrators and users of these security features. The following is a list of the information in the overview chapter.

CHAPTER 1

Security Services (Overview)

Introduction to Security Services

To maintain the security of its computing environment, the Solaris operating environment software provides the following features:

- **Machine Security –** The ability to protect machine resources and files from malicious modification or unintentional modification by users or intruders
- **Authentication –** The ability to securely identify a user, which requires the user's name and some form of proof, typically a password
- **Secure Communication –** The ability to ensure that authenticated parties can communicate without interception, modification, or spoofing
- **Auditing –** The ability to identify the source of security changes to the system, including file access, security-related system calls, and authentication failures

For a general discussion of system security, see Chapter 2.

Machine Security

Machine security ensures that the machine's resources are used properly. Access control enables users or administrators to restrict the users who are permitted access to resources on the system. The Solaris operating environment features for machine security and access control include the following:

- **Login control –** Access to the hardware, files, and processes on a computer. See Chapter 3.

- **UNIX® permissions –** Attributes of a file or directory. Permissions restrict the users and groups that are permitted to read, write, or execute a file, or search a directory. See Chapter 4.
- **Role-Based Access Control (RBAC) –** An architecture for creating special, restricted user accounts that are permitted to perform specific security-related tasks. See Chapter 5.
- **Security Enhancement Scripts –** Through the use of scripts, many system files and parameters can be adjusted to reduce security risks. See Chapter 8.
- **Device Allocation –** A facility that enables restriction on who can use a device, such as a floppy or CD-ROM drive. The facility ensures that a device is used by only one qualified user at a time. See "Managing Device Allocation (Tasks)" on page 369.
- **SunScreen™ 3.2 Secure Net –**A firewall for selectively controlling the flow of information into and out of an organization's network. The firewall also can control the flow of information between segments of a network. See the SunScreen 3.2 documentation set.

Authentication Services

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

- **Secure RPC –** An authentication technique that is based on the Diffie-Hellman method. This topic is covered in "Overview of Secure RPC" on page 161.
- **Pluggable Authentication Module (PAM) –** A framework that enables various authentication technologies to be plugged in without disturbing system entry services, such as `login` or `ftp`. See Chapter 10.
- **Sun Enterprise Authentication Module (SEAM) –** A client/server architecture that provides authentication with encryption. See Chapter 13.
- **Smart Card –** A plastic card with a microprocessor and memory that can be used with a card reader to access systems. See *Solaris Smartcard Administration Guide*.
- **Login Administration Tools –** Various commands for administering a user's ability to log in or to abort a session. See Chapter 3.

Secure Communication

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

- **Sun™ Enterprise Authentication Module (SEAM) –** A client/server architecture that provides encryption with authentication. See Chapter 13.
- **Internet Protocol Security Architecture (IPsec) –** An architecture that provides IP datagram protection. Protections include confidentiality, strong integrity of the data, data authentication, and partial sequence integrity. Partial sequence integrity is replay protection. See "IPsec (Overview)" in *System Administration Guide: IP Services*.
- **Solaris Secure Shell –** A protocol for protecting data transfers and interactive user network sessions from eavesdropping, session hijacking, and man-in-the-middle attacks. Strong authentication is provided through public key cryptography. X windows services and other network services can be tunneled safely over Secure Shell connections for additional protection. See Chapter 11.

Auditing and Device Management

Auditing is a fundamental concept of system security and maintainability. Auditing is the process of examining the history of actions and events on a system to find out what happened. Auditing entails keeping a log of what was done, by whom, when the action was done, and what was affected. Device management controls the allocation of peripheral devices, such as diskettes and CD-ROMs. For more information on Solaris auditing and device management, see Chapter 20.

PART II Managing System Security

Chapter 2	Provides overview information about file security, machine security, and network security.
Chapter 3	Provides step-by-step instructions on how to protect machines, monitor machine use, and guard against the misuse of root access.
Chapter 4	Provides step-by-step instructions on how to display file information, change file ownership and file permissions, and set special file permissions.
Chapter 5	Provides overview information for using role-based access control (RBAC).
Chapter 6	Provides step-by-step instructions for using RBAC.
Chapter 7	Provides reference information about RBAC.
Chapter 8	Provides overview information about the Automated Security Enhancement Tool (ASET). Provides step-by-step instructions on how to run ASET interactively or periodically. Periodic execution is accomplished through a `cron` job. This chapter also includes information about collecting client ASET reports on a server.

CHAPTER 2

Managing Machine Security (Overview)

To keep a machine's information secure is an important system administration responsibility. This chapter provides overview information about managing machine security.

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

Controlling Access to a Computer System

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

At the file level, the Solaris operating environment provides some standard security features that you can use to protect files, directories, and devices. At the system and network levels, the security issues are mostly the same. The first line of security defense is to control access to your system. You can control and monitor system access by doing the following:

- Maintaining physical security
- Maintaining login control
- Monitoring and limiting resource use

- Protecting files
- Controlling network access
- Reporting security problems

Maintaining Physical Security

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

You can protect a SPARC machine from unwanted access to the hardware settings. Use the `eeprom`(1M) command to require a password to access the PROM. See "How to Require a Password for Hardware Access" on page 59 for more information.

Maintaining Login Control

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

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

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

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

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

Sophisticated authentication and authorization mechanisms are available on Solaris systems. For a discussion of authentication and authorization mechanisms at the network level, see “Authentication and Authorization for Remote Access” on page 42.

Managing Password Information

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

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

Local Passwords

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

NIS and NIS+ Passwords

If your network uses NIS+, the password information is kept in the NIS+ database. Information in the NIS+ database can be protected by restricting access to authorized users. You can use the `passwd` command to change a user’s password that is stored in a NIS+ database.

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

LDAP Passwords

The Solaris LDAP Naming Service stores the password information and the shadow information in the `ou=people` container of the LDAP directory tree. On the Solaris LDAP naming service client, you can use the `passwd -r ldap` command to change a user’s password. The LDAP naming service stores the password in the LDAP repository.

In the Solaris 9 12/02 release, password policy is enforced on the Sun™ Open Net Environment (Sun ONE) Directory Server. Specifically, the client's `pam_ldap` module obeys the password policy controls that are enforced on the Sun ONE Directory Server. For more information, see "LDAP Naming Services Security Model" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

Password Encryption

Strong password encryption provides an early barrier against attack. The Solaris 9 12/02 release provides four password encryption modules. The MD5 modules and the Blowfish module provide more robust password encryption than the UNIX algorithm.

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

TABLE 2–1 Password Encryption Algorithms

Identifier	Description	Algorithm Man Page
`1`	The MD5 algorithm that is compatible with MD5 algorithms on BSD and Linux systems.	`crypt_bsdmd5`(5)
`2a`	The Blowfish algorithm that is compatible with the Blowfish algorithm on BSD systems.	`crypt_bsdbf`(5)
`md5`	The Sun MD5 algorithm, which is considered stronger than the BSD and Linux version of MD5.	`crypt_sunmd5`(5)
`__unix__`	The traditional UNIX encryption algorithm. This algorithm is the default module in the `policy.conf` file.	`crypt_unix`(5)

Algorithms Configuration in the `policy.conf` File

The following shows the default `policy.conf` file:

```
#
# Copyright 1999-2002 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# /etc/security/policy.conf
#
# security policy configuration for user attributes. see policy.conf(4)
#
#ident  "@(#)policy.conf        1.6     02/06/07 SMI"
#
AUTHS_GRANTED=solaris.device.cdrw
PROFS_GRANTED=Basic Solaris User

# crypt(3c) Algorithms Configuration
```

```
#
# CRYPT_ALGORITHMS_ALLOW specifies the algorithms that are allowed to
# be used for new passwords.  This is enforced only in crypt_gensalt(3c).
#
CRYPT_ALGORITHMS_ALLOW=1,2a,md5

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

# The Solaris default is the traditional UNIX algorithm.  This is not
# listed in crypt.conf(4) since it is internal to libc.  The reserved
# name __unix__ is used to refer to it.
#
CRYPT_DEFAULT=__unix__
```

When you change the value for `CRYPT_DEFAULT`, the passwords of new users are encrypted with the algorithm that is associated with the new value. When current users change their passwords, how their old password was encrypted affects which algorithm is used to encrypt the new password.

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

Initial Password Algorithm	Changed Password Algorithm	Explanation
`crypt_bsdmd5`	`crypt_bsdmd5`	The identifier of `crypt_bsdmd5` is `1`, the value of `CRYPT_DEFAULT`. The user's password continues to be encrypted with the `crypt_bsdmd5` algorithm.
`crypt_bsdbf`	`crypt_bsdbf`	The identifier of `crypt_bsdbf` is `2a`. Because `2a` is in the `CRYPT_ALGORITHMS_ALLOW` list, the new password is encrypted with the `crypt_bsbdf` algorithm.
`crypt_md5`	`crypt_md5`	The identifier of `crypt_md5` is `md5`. Because `md5` is in the `CRYPT_ALGORITHMS_ALLOW` list, the new password is encrypted with the `crypt_md5` algorithm.
`crypt_unix`	`crypt_bsdmd5`	The identifier of `crypt_unix` is `__unix__`. The `__unix__` identifier is not in the `CRYPT_ALGORITHMS_ALLOW` list. Therefore, the `crypt_unix` algorithm cannot be used. The new password is encrypted with the `CRYPT_DEFAULT` algorithm.

For more information on the syntax for configuring the algorithm choices, see the `policy.conf`(4) man page. For information on how to use the new password encryption algorithms, see "Changing the Default Algorithm for Password Encryption" on page 53.

Special System Logins

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

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

TABLE 2–2 System Logins

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

Remote Logins

Remote logins offer a tempting avenue for intruders. The Solaris operating environment provides a number of commands to monitor, limit, and disable remote logins. For procedures, see "Securing Logins and Passwords" on page 49.

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

Dial-up Logins

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

Only superuser can create or change a dial-up password. To ensure the integrity of the system, the password should be changed about once a month. The most effective use of this feature is to require a dial-up password to gain access to a gateway system. For how to set up dial-up passwords, see "How to Create a Dial-up Password" on page 52.

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

- If the user's login shell in `/etc/passwd` matches an entry in `/etc/d_passwd`, the user must supply a dial-up password.
- If the user's login shell in `/etc/passwd` is not found in `/etc/d_passwd`, the password entry for `/usr/bin/sh` is used.

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

Controlling Access to Machine Resources

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

Limiting and Monitoring Superuser

Your system requires a `root` password for superuser mode. In the default configuration, a user cannot remotely log in to a system as `root`. When logging in remotely, a user must log in with the user's username and then use the `su` command to become `root`. For security reasons, you can monitor who has been using the `su` command, especially those users who are trying to gain superuser access. For procedures that monitor superuser and limit access to superuser, see "Monitoring and Restricting Superuser" on page 57.

Configuring Role-Based Access Control to Replace `root`

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

Each role requires that a known user log in with their username and password. After logging in, the user then assumes the role with a specific role password. Someone who learns the root password, then, has limited ability to damage your system. For more on RBAC, see Chapter 5.

Preventing Unintentional Misuse of Machine Resources

You can prevent you and your users from unintentional error. You can keep from running a Trojan horse by setting the `PATH` variable correctly. You can prevent user error by steering users to those parts of the system that the users need for their jobs. In fact, through careful setup, you can ensure that users see only those parts of the system that help the users work efficiently.

Setting the Path Variable

You should take care to set the path variable correctly. Otherwise, you can accidentally run a program that was introduced by someone else. The intruding program can corrupt your data or harm your system. This kind of program, which creates a security hazard, is referred to as a "Trojan horse." For example, a substitute `su` program could be placed in a public directory where you, as system administrator, might run the substitute program. Such a script would look just like the regular `su` command. Since the script removes itself after execution, you would have little evidence to show that you have actually run a Trojan horse.

The path variable is automatically set at login time. The path is set through the startup files: `.login`, `.profile`, and `.cshrc`. When you set up the user search path so that the current directory (.) comes last, you are protected from running this type of Trojan horse. The path variable for superuser should not include the current directory at all.

The Automated Security Enhancement Tool (ASET) examines the startup files to ensure that the path variable is set up correctly. ASET also makes sure that the path variable does not contain a dot (.) entry.

Assigning a Restricted Shell

The standard shell allows a user to open files, execute commands, and so on. The restricted shell is invoked with the `/usr/lib/rsh` command. The restricted shell can be used to limit the ability of a user to change directories and to execute commands. Note that the restricted shell is not the remote shell, which is `/usr/sbin/rsh`. The restricted shell differs from the standard shell in the following ways:

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

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

For information about the restricted shell, see the `rsh`(1M) man page.

A more secure alternative to the restricted shell is the Secure Shell, the `ssh` command. The Secure Shell enables users to securely access a remote host over an unsecured network. For information about using the Secure Shell, see Chapter 12.

Restricting Access to Data in Files

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

After you have established login restrictions, you can control access to the data on your machine. You might want to allow some users to read some files, and give other users permission to change or delete some files. You might have some data that you do not want anyone else to see. Chapter 4 discusses how to set file permissions.

Restricting `setuid` Executable Files

Executable files can be security risks. Many executable programs have to be run as `root`, that is, as superuser, to work properly. These programs run with the user ID set to `0`, that is, `setuid=0`. Anyone who is running these programs runs the programs with the `root` ID. A program that runs with the `root` ID creates a potential security problem if the program was not written with security in mind.

Except for the executables that Sun ships with the `setuid` bit set to `root`, you should disallow the use of `setuid` programs. If you cannot disallow the use of `setuid` programs, then you should at least restrict their use. Secure administration requires few `setuid` programs.

Using the Automated Security Enhancement Tool (ASET)

The ASET security package provides automated administration tools that enable you to control and monitor your system's security. You specify an ASET security level. ASET provides three security levels: low, medium, and high. At each higher level, ASET's file-control functions increase to reduce file access and tighten your machine's security.

For more information, see Chapter 8.

Using the Resource Manager

Solaris software provides a sophisticated resource management tool, the Resource Manager. The Resource Manager can help prevent denial of service attacks. With the Resource Manager, you can designate resources for particular projects. You can prevent scripts from overrunning the machine's resources. You can limit the space that a project can occupy. For a description and an extensive example of how to use the Resource Manager, see "Solaris 9 Resource Manager Topics" in *System Administration Guide: Resource Management and Network Services*.

Monitoring Use of Machine Resources

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

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

With this kind of knowledge, you can use the available tools to audit machine use and monitor the activities of individual users. Monitoring is very useful when there is a suspected breach in security. For more information on the auditing module, see Chapter 20.

Controlling Access to Files

The Solaris operating environment is a multiuser environment. In a multiuser environment, all the users who are logged in to a machine can read files that belong to other users. With the appropriate file permissions, users can also use files that belong to other users. Table 2–3 describes the commands for file system security. For step-by-step instructions on securing files, see Chapter 4.

Commands for File System Security

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

TABLE 2–3 Commands for File System Security

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

File Encryption

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

Fortunately, an additional layer of security is available to all users of Solaris software in the United States, the Solaris Encryption Kit. The encryption kit includes the `crypt` command, which scrambles the data to disguise the text. For more information, see the `crypt`(1) man page.

Access Control Lists (ACLs)

ACLs, pronounced “ackkls”, can provide greater control over file permissions. You add ACLs when the traditional UNIX file protection in the Solaris operating environment is not sufficient. The traditional UNIX file protection provides read, write, and execute permissions for the three user classes: owner, group, and other. An ACL provides finer-grained file security. ACLs enable you to define the following file permissions:

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

For step–by–step instructions on using ACLs, see “Using Access Control Lists (ACLs)” on page 76.

The following table lists the commands for administering ACLs on files or directories.

TABLE 2–4 Access Control List (ACL) Commands

Command	Description	Man Page
`setfacl`	Sets, adds, modifies, and deletes ACL entries	`setfacl`(1)
`getfacl`	Displays ACL entries	`getfacl`(1)

Sharing Files Across Machines

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

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

Restricting `root` Access to Shared Files

In general, superuser is not allowed `root` access to file systems that are shared across the network. The NFS system prevents `root` access to mounted file systems by changing the user of the requester to the user `nobody` with user ID `60001`. The access

rights of user `nobody` are the same as those access rights that are given to the public. The user `nobody` has the access rights of a user without credentials. For example, if the public has only execute permission for a file, then user `nobody` can only execute that file.

An NFS server can grant superuser privileges on a shared file system on a per-host basis. To grant these privileges, use the `root=`*hostname* option to the `share` command. You should use this option with care.

Controlling Network Access

Computers are often part of a configuration of computers. The configuration is called a *network*. A network allows connected computers to exchange information. Networked computers can access data and other resources from other computers on the network. Networking has created a powerful and sophisticated way of computing. However, networking also jeopardizes computer security.

For instance, within a network of computers, individual machines are open to allow the sharing of information. Also, because many people have access to the network, unwanted access is more likely, especially through user error. For example, a poor use of passwords can allow unwanted access.

Network Security Mechanisms

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

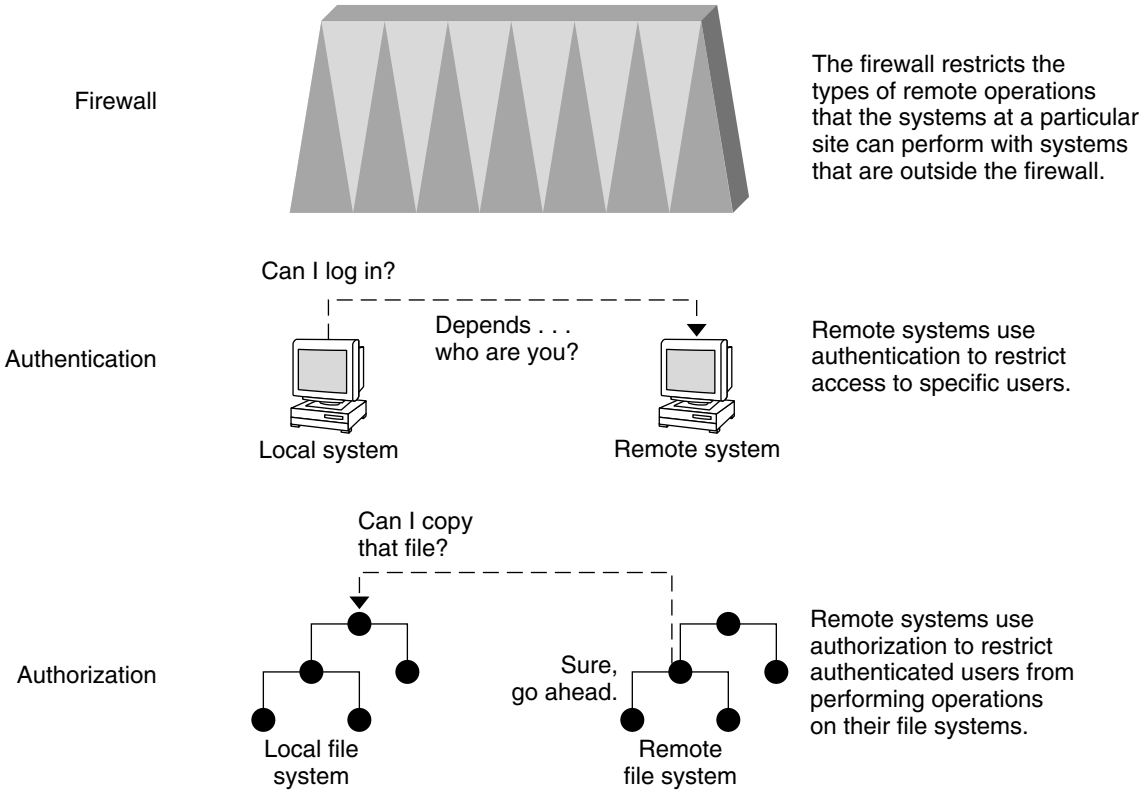


FIGURE 2–1 Security Restrictions for Remote Operations

Authentication and Authorization for Remote Access

Authentication is a way to restrict access to specific users when these users access a remote machine. Authentication can be set up at both the machine level and the network level. Once a user gains access to a remote machine, *authorization* is a way to restrict operations that the user can perform on the remote system. The following table lists the types of authentications and authorizations that can help protect your machines on the network against unauthorized use.

TABLE 2–5 Types of Authentication and Authorization for Remote Access

Type	Description	Where to Find Information
LDAP and NIS+	The LDAP directory service and the NIS+ name service can provide both authentication and authorization at the network level.	*System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* and *System Administration Guide: Naming and Directory Services (FNS and NIS+)*
Remote login commands	The remote login commands enable users to log in to a remote machine over the network and use its resources. The remote login commands are `rlogin`, `rcp`, `ftp`. If you are a "trusted host," authentication is automatic. Otherwise, you are asked to authenticate yourself.	"Accessing Remote Systems (Tasks)" in *System Administration Guide: Resource Management and Network Services*
Secure RPC	Secure RPC improves the security of network environments by authenticating users who make requests on remote machines. You can use either the UNIX, DES, or Kerberos authentication system for Secure RPC.	"Overview of Secure RPC" on page 161
	Secure RPC can also be used to provide additional security to the NFS environment. An NFS environment with secure RPC is called Secure NFS.	"NFS Services and Secure RPC" on page 161
DES encryption	The Data Encryption Standard (DES) encryption functions use a 56-bit key to encrypt a secret key.	"DES Encryption" on page 162
Diffie-Hellman authentication	This authentication method is based on the ability of the sending machine to use a common key to encrypt the current time. The receiving machine decrypts the common key. The machine then checks the time against its current time.	"Diffie-Hellman Authentication" on page 162
Kerberos	Kerberos uses DES encryption to authenticate a user when logging in to the system.	See "How to Configure a Master KDC" on page 229 for an example.

Using Privileged Ports Between Solaris Systems

If you do not want to run Secure RPC, a possible substitute is the Solaris "privileged port" mechanism. A privileged port is assigned with a port number of less than 1024. After a client system has authenticated the client's credential, the client builds a connection to the server by using the privileged port. The server then verifies the client credential by examining the connection's port number.

Non-Solaris clients, however, might be unable to communicate by using the privileged port. If the clients cannot communicate over the port, you see an error message that is similar to the following:

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

Firewall Systems

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

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

A firewall can also be useful between some internal networks. For example, you can set up a firewall or secure gateway computer to restrict the transfer of packets. The gateway can forbid packet exchange between two networks unless the gateway computer is the origin address or the destination address of the packet. A firewall should also be set up to forward packets for particular protocols only. For example, you can allow packets for transferring mail, but not allow packets for the `telnet` or the `rlogin` command. ASET, when run at high security, disables the forwarding of Internet Protocol (IP) packets.

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

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

A firewall system should not have any *trusted hosts*. A trusted host is a host from which a user can log in without being required to type in a password. A firewall system should not share any of its file systems, or mount any file systems from other servers.

ASET can be used to harden a machine into a firewall. ASET enforces high security on a firewall system, as described in Chapter 8. Similarly, IPsec provides firewall protection. For more information on using IPsec to protect network traffic, see "IPsec (Overview)" in *System Administration Guide: IP Services*.

Packet Smashing

Most local area networks transmit data between computers in blocks that are called packets. Through a procedure that is called *packet smashing*, unauthorized users can corrupt data. Data can also be destroyed. Packet smashing involves capturing the packets before the packets reach their destination. The intruder then injects arbitrary data into the contents, and sends the packets back on their original course. On a local area network, packet smashing is impossible because packets reach all machines, including the server, at the same time. Packet smashing is possible on a gateway, however, so make sure that all gateways on the network are protected.

The most dangerous attacks are those attacks that affect the integrity of the data. Such attacks involve changing the contents of the packets or impersonating a user. Attacks that involve eavesdropping do not compromise data integrity. An eavesdropper records conversations for later replay. An eavesdropper does not impersonate a user. While eavesdropping attacks do not attack data integrity, the attacks do affect privacy. You can protect the privacy of sensitive information by encrypting data that goes over the network. For how to encrypt IP datagrams, see "Internet Key Exchange" in *System Administration Guide: IP Services*.

Reporting Security Problems

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

CHAPTER 3

Securing Machines (Tasks)

This chapter describes the procedures for securing machines in the Solaris environment. The procedures are introduced in the following section:

- "Securing Machines (Task Map)" on page 47

For overview information about machine security, see Chapter 2.

Securing Machines (Task Map)

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

Task	Description	For Instructions
Display a user's login status	Use the `logins` command to view a user's login status information.	"How to Display a User's Login Status" on page 49
Find users who do not have passwords	Use the `logins` command to find only those users whose accounts do not require a password.	"How to Display Users Without Passwords" on page 50
Disable logins temporarily	Deny user logins to a machine as part of system shutdown or routine maintenance.	"How to Temporarily Disable User Logins" on page 50
Provide strong password encryption	Specify algorithms for password encryption.	"How to Specify an Algorithm for Password Encryption" on page 54

Task	Description	For Instructions
Provide strong password encryption with a name service	Specify algorithms for password encryption when you are using a name service.	"How to Specify a New Password Algorithm for an NIS+ Domain" on page 55 "How to Specify a New Password Algorithm for an NIS Domain" on page 55 "How to Specify a New Password Algorithm for an LDAP Domain" on page 55
Add new password encryption module	Add third-party algorithms.	"How to Install a Password Encryption Module From a Third Party" on page 56
Save failed login attempts	Create a log of users who failed to provide the correct password after five attempts.	"How to Save Failed Login Attempts" on page 51
Create a dial-up password	Require an additional password for users who log in remotely through a modem or dial-up port.	"How to Create a Dial-up Password" on page 52
Disable dial-up entry temporarily	Prevent users from dialing in remotely through a modem or port.	"How to Temporarily Disable Dial-up Logins" on page 53
Monitor who is using the `su` command	Read the `sulog` file on a regular basis.	"How to Monitor Who Is Using the `su` Command" on page 57
Display superuser activity on the console	Monitor superuser access attempts.	"How to Display Superuser (`root`) Access Attempts to the Console" on page 58
Prevent remote access to the console as superuser	Require remote users to log in with their username and then become root.	"How to Prevent Remote Login by Superuser (`root`)" on page 58
Prevent users from changing machine parameters	Prevent users from changing PROM settings.	"How to Require a Password for Hardware Access" on page 59
Disable the abort sequence	Prevent users from accessing the PROM.	"How to Disable or Enable a System's Abort Sequence" on page 60

Securing Logins and Passwords

This section describes how to control and monitor logins.

▼ How to Display a User's Login Status

1. **Become superuser or assume an equivalent role.**
2. **Display a user's login status by using the `logins` command.**

```
# logins -x -l username
```

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

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

Example—Displaying a User's Login Status

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

```
# logins -x -l rimmer
rimmer       500     staff           10   Annalee J. Rimmer
                     /export/home/rimmer
                     /bin/sh
                     PS 010170 10 7 -1
```

`rimmer`	Identifies the user's login name.
`500`	Identifies the user ID (UID).
`staff`	Identifies the user's primary group.
`10`	Identifies the group ID (GID).
`Annalee J. Rimmer`	Identifies the comment.
`/export/home/rimmer`	Identifies the user's home directory.

`/bin/sh`	Identifies the login shell.
`PS 010170 10 7 -1`	Specifies the password aging information: ■ Last date that the password was changed ■ Number of days that are required between changes ■ Number of days before a change is required ■ Warning period

▼ How to Display Users Without Passwords

1. **Become superuser or assume an equivalent role.**
2. **Display all users who have no passwords by using the `logins` command.**

   ```
   # logins -p
   ```

   The `-p` option displays a list of users with no passwords. The `logins` command can use the password databases on the local machine and on the network. The command can use the local `/etc/passwd` file. The command can use the password databases for the name services to obtain a user's login status.

Example—Displaying Users Without Passwords

The following example shows that the user `pmorph` does not have a password.

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

▼ How to Temporarily Disable User Logins

1. **Become superuser or assume an equivalent role.**
2. **Create the `/etc/nologin` file by using an editor.**

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

3. **Include a message about system availability.**
4. **Close and save the file.**

   Create this file to disallow user logins during system shutdown or routine maintenance. If a user attempts to log in to a system where the `nologin` file exists, the contents of this file are displayed. Then, the user login is terminated.

   Superuser logins are not affected. For more information, see the `nologin`(4) man page.

Example—Disabling User Logins

This example shows how to notify users of system unavailability.

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

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

***The system will be unavailable until 12 noon.***
```

You can also bring the system to run level 0, single-user mode. For information on bringing the system to single-user mode, see "Shutting Down a System (Tasks)" in *System Administration Guide: Basic Administration*.

▼ How to Save Failed Login Attempts

1. **Become superuser or assume an equivalent role.**
2. **Create the `loginlog` file in the `/var/adm` directory.**

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

3. **Set read and write permissions for `root` on the `loginlog` file.**

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

4. **Change group membership to `sys` on the `loginlog` file.**

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

5. **Make sure that the log works by attempting to log into the system five times with the wrong password. Then, display the `/var/adm/loginlog` file.**

   ```
   # more /var/adm/loginlog
   rimmer:/dev/pts/1:Wed Jan 16 09:22:31 2002
   rimmer:/dev/pts/1:Wed Jan 16 09:22:39 2002
   rimmer:/dev/pts/1:Wed Jan 16 09:22:45 2002
   rimmer:/dev/pts/1:Wed Jan 16 09:22:53 2002
   rimmer:/dev/pts/1:Wed Jan 16 09:23:01 2002
   #
   ```

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

   The `loginlog` file might grow quickly. To use this file in a timely manner, you should check and clear its contents occasionally. A `loginlog` file that shows a lot of activity can indicate an attempt to break into the computer system. For more information, see the `loginlog`(4) man page.

▼ How to Create a Dial-up Password

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

1. **Become superuser or assume an equivalent role.**
2. **Create an `/etc/dialups` file that contains a list of serial devices. Include all the ports that are being protected with dial-up passwords.**

   The `/etc/dialups` file should look like the following:

   ```
   /dev/term/a
   /dev/term/b
   /dev/term/c
   ```

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

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

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

   You are going to add the encrypted password for each login program later in the procedure.

4. **Set ownership to `root` on the two files.**

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

5. **Set group ownership to `root` on the two files.**

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

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

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

7. **Create the encrypted passwords.**

   a. **Create a temporary user.**

      ```
      # useradd username
      ```

   b. **Create a password for the temporary user.**

      ```
      # passwd username
      ```

c. **Capture the encrypted password.**

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

d. **Edit the *username*`.temp` file.**

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

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

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

e. **Delete the temporary user.**

```
# userdel username
```

8. **Copy the encrypted password from *username*`.temp` file into the `/etc/d_passwd` file.**

You can create a different password for each login shell, or use the same password for each login shell.

9. **Inform your dial-up users of the password.**

You should ensure that your means of informing the users cannot be tampered with.

▼ How to Temporarily Disable Dial-up Logins

1. **Become superuser or assume an equivalent role.**
2. **Put the following single-line entry into the `/etc/d_passwd` file:**

```
/usr/bin/sh:*:
```

Changing the Default Algorithm for Password Encryption

By default, user passwords are encrypted with the `crypt_unix` algorithm. In the Solaris 9 12/02 release, you can use a stronger encryption algorithm, such as MD5 or Blowfish, by changing the default password encryption algorithm. The next time that your users change their password, the algorithm that you specified encrypts the password.

Note – The following procedures do not work if you are running a Solaris environment from an earlier release. This functionality works only on machines that are running the Solaris 9 12/02 release and later releases of the Solaris operating environment.

▼ How to Specify an Algorithm for Password Encryption

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

1. **Become superuser or assume an equivalent role.**
2. **Specify the identifier for the encryption algorithm as the value for the `CRYPT_DEFAULT` variable in the `/etc/security/policy.conf` file.**

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

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

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

   For more information on the syntax for configuring the algorithm choices, see the `policy.conf`(4) man page.

Example—Using the Blowfish Algorithm for Password Encryption

In this example, the identifier for the Blowfish algorithm, `2a`, is specified as the value for the `CRYPT_DEFAULT` variable. The `policy.conf` entries that control password encryption would look like the following:

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

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

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

1. **Specify the password encryption algorithm in the `/etc/security/policy.conf` file on the NIS+ master.**
2. **To minimize confusion, copy the NIS+ master's `/etc/security/policy.conf` file to every host in the NIS+ domain.**

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

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

1. **Specify the password encryption algorithm in the `/etc/security/policy.conf` file on the NIS client.**
2. **Copy the modified `/etc/security/policy.conf` file to every client machine in the NIS domain.**
3. **To minimize confusion, copy the modified `/etc/security/policy.conf` file to the NIS root server and to the slave servers.**

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

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

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

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

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

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

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

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

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

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

▼ How to Install a Password Encryption Module From a Third Party

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

1. **Add the software by using the `pkgadd` command.**

   For detailed instructions on how to add software, see "Adding or Removing a Software Package" in *System Administration Guide: Basic Administration*.

2. **Read the `/etc/security/crypt.conf` file to confirm that the new module and module identifier are in the list of encryption algorithms.**

   For example, the following lines show a `crypt.conf` file that was modified by a package that installed the `crypt_rot13` algorithm.

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

```
rot13 /usr/lib/security/$ISA/crypt_rot13.so

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

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

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

```
# Copyright 1999-2002 Sun Microsystems, Inc.  All rights reserved.
# ...
#ident  "@(#)policy.conf        1.6     02/06/07 SMI"
# ...
# crypt(3c) Algorithms Configuration
CRYPT_ALGORITHMS_ALLOW=1,2a,md5,rot13
#CRYPT_ALGORITHMS_DEPRECATE=__unix__
CRYPT_DEFAULT=md5
```

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

Monitoring and Restricting Superuser

An alternative to using the superuser account is to set up role-based access control. Role-based access control is called RBAC. For overview information on RBAC, see Chapter 5. For how to set up RBAC, see Chapter 6.

▼ How to Monitor Who Is Using the `su` Command

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

1. **Become superuser or assume an equivalent role.**
2. **Monitor the contents of the `/var/adm/sulog` file on a regular basis.**

```
# more /var/adm/sulog
SU 12/20 16:26 + pts/0 nathan-root
SU 12/21 10:59 + pts/0 nathan-root
SU 01/12 11:11 + pts/0 root-janedoe
```

```
SU 01/12 14:56 + pts/0 pmorph-root
SU 01/12 14:57 + pts/0 pmorph-root
```

The entries display the following information:

- The date and time that the command was entered
- If the attempt was successful

  A + indicates a successful attempt. A - indicates an unsuccessful attempt.
- The port from which the command was issued
- The name of the user and the name of the switched identity

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

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

▼ How to Display Superuser (`root`) Access Attempts to the Console

1. **Become superuser or assume an equivalent role.**
2. **Edit the `/etc/default/su` file.**
3. **Uncomment the following line:**

   ```
   CONSOLE=/dev/console
   ```

4. **Use the `su` command to become `root`.**

   Verify that a message is printed on the system console.

   This method immediately detects someone who is trying to gain superuser access to the system that you are on.

▼ How to Prevent Remote Login by Superuser (`root`)

Note – Superuser login is restricted to the console by default when you install the Solaris release.

---

1. **Become superuser or assume an equivalent role.**
2. **Edit the `/etc/default/login` file.**
3. **Uncomment the following line:**

```
CONSOLE=/dev/console
```

When superuser access is restricted to the console, you can log in to a system as superuser only from the console. Any users who try to remotely log in to this system must first log in with their user login. After logging in with their user name, users then use the `su` command to become superuser.

4. **Attempt to log in remotely as superuser to this system, and verify that the operation fails.**

Securing the Hardware

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

▼ How to Require a Password for Hardware Access

1. **Become superuser or assume an equivalent role.**
2. **In a terminal, enter the PROM security mode. Type the following:**

```
# eeprom security-mode=command

Changing PROM password:
    New password: password
    Retype new password: password
```

Choose the value `command` or `full`. See the `eeprom`(1M) man page for more details.

3. **If you are not prompted to enter a PROM password, the system already has a PROM password. To change the PROM password, run the command:**

```
# eeprom security-password=<Type the Return key>
Changing PROM password:
New password: password
Retype new password: password
```

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

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

▼ How to Disable or Enable a System's Abort Sequence

Use the following procedure to disable a machine's abort sequence. The default system behavior is that a system's abort sequence is enabled.

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

1. **Become superuser or assume an equivalent role.**
2. **Change the value of `KEYBOARD_ABORT` to `disable`.**

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

   ```
   # vi /etc/default/kbd
   ...
   # KEYBOARD_ABORT affects the default behavior of the keyboard abort
   # sequence, see kbd(1) for details.  The default value is "enable".
   # The optional value is "disable".  Any other value is ignored.
   ...
   #KEYBOARD_ABORT=enable
   KEYBOARD_ABORT=disable
   ```

3. **Update the keyboard defaults.**

   ```
   # kbd -i
   ```

CHAPTER 4

Securing Files (Tasks)

This chapter describes the procedures for securing files in the Solaris environment.

The following is a list of the step-by-step instructions in this chapter.

File Security Features

This section describes the features that constitute a file's security.

User Classes

For each file, there are three classes of users that specify the levels of security:

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

Only the owner of the file or `root` can assign or modify file permissions.

File Permissions

The following table lists and describes the permissions that you can give to each user class for a file.

TABLE 4–1 File Permissions

Symbol	Permission	Description
`r`	Read	Designated users can open and read the contents of a file.
`w`	Write	Designated users can write to the file (modify its contents), add to it, or delete it.
`x`	Execute	Designated users can execute the file (if it is a program or shell script), or run the program with one of the `exec(2)` system calls.
`-`	Denied	Designated users cannot read, write, or execute the file.

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

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

Directory Permissions

The following table lists and describes the permissions that you can give to each user class for a directory.

TABLE 4–2 Directory Permissions

Symbol	Permission	Description
`r`	Read	Designated users can list files in the directory.
`w`	Write	Designated users can add or remove files or links in the directory.

TABLE 4–2 Directory Permissions *(Continued)*

Symbol	Permission	Description
x	Execute	Designated users can open or execute files in the directory. They also can make the directory and the directories beneath it current.

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

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

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

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

You should monitor your system for any unauthorized use of the `setuid` and `setgid` permissions to gain superuser privileges. To search for and list all of the files that use these permissions, see "How to Find Files With `setuid` Permissions" on page 74. A suspicious listing grants ownership of such a program to a user rather than to `root` or `bin`.

`setuid` Permission

When set-user identification (`setuid`) permission is set on an executable file, a process that runs this file is granted access based on the owner of the file (usually `root`), rather than the user who is running the executable file. This special permission allows a user to access files and directories that are normally only available to the owner. For example, the `setuid` permission on the `passwd` command makes it possible for a user to change passwords, assuming the permissions of the `root` ID:

```
-r-sr-sr-x   3 root     sys       104580 Sep 16 12:02 /usr/bin/passwd
```

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

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

`setgid` Permission

The set-group identification (`setgid`) permission is similar to `setuid`, except that the process's effective group ID (GID) is changed to the group owner of the file, and a user is granted access based on permissions granted to that group. The `/usr/bin/mail` command has `setgid` permissions:

```
-r-x--s--x   1 root     mail       63628 Sep 16 12:01 /usr/bin/mail
```

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

You should monitor your system for any unauthorized use of the `setuid` and `setgid` permissions to gain superuser privileges. To search for and list all of the files that use these permissions, see "How to Find Files With `setuid` Permissions" on page 74. A suspicious listing grants group ownership of such a program to a user rather than to `root` or `bin`.

Sticky Bit

The *sticky bit* is a permission bit that protects the files within a directory. If the directory has the sticky bit set, a file can be deleted only by the owner of the file, the owner of the directory, or by `root`. This special permission prevents a user from deleting other users' files from public directories such as `/tmp`:

```
drwxrwxrwt 7  root  sys   400 Sep  3 13:37 tmp
```

Be sure to set the sticky bit manually when you set up a public directory on a TMPFS file system.

Default `umask` Setting

When you create a file or directory, it has a default set of permissions. These default permissions are determined by the `umask` setting in the `/etc/profile` file, or in your `.cshrc` or `.login` file. By default, the system sets the permissions on a text file to `666`, which grants read and write permission to user, group, and others, and to `777` on a directory or executable file.

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

The following table shows some typical `umask` settings, and the effect on an executable file.

TABLE 4–3 `umask` Settings for Different Security Levels

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

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

Displaying File Information

This section describes how to display file information.

▼ How to Display File Information

Display information about all the files in a directory by using the `ls` command.

- **Type the following command to get a long listing of all files in the current directory.**

```
% ls -la
```

`-l`	Displays the long format that includes user and group ownership and file permissions.
`-a`	Displays all files, including hidden files that begin with a dot (.).

Each line in the display has the following information about a file:

- Type of file

A file can be one of seven types. The following table lists the possible file types.

TABLE 4–4 File Types

Symbol	Type
-	Text or program
D	Door
d	Directory
b	Block special file
c	Character special file
p	Named pipe (FIFO)
l	Symbolic link
s	Socket

- Permissions; see "Setting the Path Variable" on page 36 and Table 4–2 for descriptions
- Number of hard links
- Owner of the file
- Group of the file
- Size of the file, in bytes
- Date the file was created or the last date that the file was changed
- Name of the file

Example—Displaying File Information

The following example displays the partial list of the files in the `/sbin` directory.

```
% cd /sbin
% ls -la
total 13456
drwxr-xr-x   2 root     sys          512 Sep  1 14:11 .
drwxr-xr-x  29 root     root        1024 Sep  1 15:40 ..
-r-xr-xr-x   1 root     bin       218188 Aug 18 15:17 autopush
lrwxrwxrwx   1 root     root          21 Sep  1 14:11 bpgetfile -> ...
-r-xr-xr-x   1 root     bin       505556 Aug 20 13:24 dhcpagent
-r-xr-xr-x   1 root     bin       456064 Aug 20 13:25 dhcpinfo
-r-xr-xr-x   1 root     bin       272360 Aug 18 15:19 fdisk
-r-xr-xr-x   1 root     bin       824728 Aug 20 13:29 hostconfig
-r-xr-xr-x   1 root     bin       603528 Aug 20 13:21 ifconfig
-r-xr-xr-x   1 root     sys       556008 Aug 20 13:21 init
-r-xr-xr-x   2 root     root      274020 Aug 18 15:28 jsh
-r-xr-xr-x   1 root     bin       238736 Aug 21 19:46 mount
```

```
-r-xr-xr-x   1 root     sys         7696 Aug 18 15:20 mountall
   .
   .
   .
```

Changing File Ownership

This section describes how to change the ownership and group ownership of a file.

By default, the owner cannot use the `chown` command to change the owner of a file or directory. However, you can enable the owner to use the `chown` command by adding the following line to the system's `/etc/system` file and rebooting the system.

```
set rstchown = 0
```

For more information, see `chown`(1).

In addition, the owner can only use the `chgrp` command to change the group of a file to a group in which the owner belongs by default. For example, if the owner of a file only belongs to the `staff` and `sysadm` groups, the owner can only change the group of a file to `staff` or `sysadm` group.

However, you can enable the owner to change the group of a file to a group in which the owner doesn't belong by adding the following line to the system's `/etc/system` file and rebooting the system.

```
set rstchown = 0
```

For more information, see `chgrp`(1).

Also, be aware that there can be other restrictions on changing ownership and groups on NFS-mounted file systems.

▼ How to Change the Owner of a File

Use the following procedure to change the ownership of a file.

1. **Become superuser or assume an equivalent role.**
2. **Change the owner of a file by using the `chown` command.**

   # **`chown`** *new-owner filename*

new-owner	Specifies the user name or UID of the new owner of the file or directory.
filename	Specifies the file or directory.

3. **Verify that the owner of the file has changed.**

```
# ls -l filename
```

Example—Changing the Owner of a File

In the following example, the ownership on `myfile` is changed to the user `rimmer`.

```
# chown rimmer myfile
# ls -l myfile
-rw-r--r--   1 rimmer   scifi   112640 May 24 10:49 myfile
```

▼ How to Change Group Ownership of a File

Use the following procedure to change the group ownership of a file.

1. **Become superuser or assume an equivalent role.**
2. **Change the group owner of a file by using the `chgrp` command.**

```
$ chgrp group filename
```

group	Specifies the group name or GID of the new group of the file or directory.
filename	Specifies the file or directory.

For information on setting up groups, see "Managing User Accounts and Groups (Overview)" in *System Administration Guide: Basic Administration*.

3. **Verify that the group owner of the file has changed.**

```
$ ls -l filename
```

Example—Changing Group Ownership of a File

In the following example, the group ownership on `myfile` is changed to the group `scifi`.

```
$ chgrp scifi myfile
$ ls -l myfile
-rwxrw-- 1 rimmer scifi 12985 Nov 12 16:28 myfile
```

Changing File Permissions

The `chmod` command enables you to change the permissions on a file. You must be superuser or the owner of a file or directory to change its permissions.

You can use the `chmod` command to set permissions in either of two modes:

- **Absolute Mode –** Use numbers to represent file permissions (the method most commonly used to set permissions). When you change permissions by using the absolute mode, you represent permissions for each triplet by an octal mode number.
- **Symbolic Mode –** Use combinations of letters and symbols to add or remove permissions.

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

TABLE 4–5 Setting File Permissions in Absolute Mode

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

You can set special permissions on a file in absolute or symbolic modes. However, you cannot set or remove `setuid` permissions on a directory by using absolute mode. You must use symbolic mode. In absolute mode, you set special permissions by adding a new octal value to the left of the permission triplet. The following table lists the octal values to set special permissions on a file.

TABLE 4–6 Setting Special Permissions in Absolute Mode

Octal Value	Special Permissions Set
1	Sticky bit
2	setgid
4	setuid

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

TABLE 4–7 Setting File Permissions in Symbolic Mode

Symbol	Function	Description
u	Who	User (owner)
g	Who	Group
o	Who	Others
a	Who	All
=	Operator	Assign
+	Operator	Add
-	Operator	Remove
r	Permission	Read
w	Permission	Write
x	Permission	Execute
l	Permission	Mandatory locking, setgid bit is on, group execution bit is off
s	Permission	setuid or setgid bit is on
S	Permission	suid bit is on, user execution bit is off
t	Permission	Sticky bit is on, execution bit for others is on
T	Permission	Sticky bit is on, execution bit for others is off

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

who Specifies whose permissions are to be changed.

operator	Specifies the operation to be performed.
permissions	Specifies what permissions are to be changed.

▼ How to Change Permissions in Absolute Mode

Use the following procedure to change permissions in absolute mode.

1. **If you are not the owner of the file or directory, become superuser or assume an equivalent role.**

   Only the current owner or superuser can use the `chmod` command to change file permissions on a file or directory.

2. **Change permissions in absolute mode by using the `chmod` command.**

   `%` **`chmod`** *nnn filename*

nnn	Specifies the octal values that represent the permissions for the file owner, file group, and others, in that order. See Table 4–5 for the list of valid octal values.
filename	Specifies the file or directory.

Note – If you use the `chmod` command to change the file group permissions on a file with ACL entries, both the file group permissions and the ACL mask are changed to the new permissions. Be aware that the new ACL mask permissions can change the permissions for additional users and groups who have ACL entries on the file. Use the `getfacl` command to make sure that the appropriate permissions are set for all ACL entries. For more information, see the `getfacl`(1) man page.

---

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

   `%` **`ls -l`** *filename*

Example—Changing Permissions in Absolute Mode

In the following example, the permissions of a public directory are changed from `744` (read, write, execute; read-only; and read-only) to `755` (read, write, execute; read and execute; and read and execute).

```
# ls -ld public_dir
drwxr--r--  1 ignatz   staff    6023 Aug  5 12:06 public_dir
# chmod 755 public_dir
# ls -ld public_dir
drwxr-xr-x  1 ignatz   staff    6023 Aug  5 12:06 public_dir
```

In the following example, the permissions of an executable shell script are changed from read and write to read, write, and execute.

```
% ls -l my_script
-rw------- 1 ignatz   staff    6023 Aug  5 12:06 my_script
% chmod 700 my_script
% ls -l my_script
-rwx------ 1 ignatz   staff    6023 Aug  5 12:06 my_script
```

▼ How to Change Special Permissions in Absolute Mode

Use the following procedure to change special permissions in absolute mode.

1. **If you are not the owner of the file or directory, become superuser or assume an equivalent role.**

   Only the current owner or superuser can use the `chmod` command to change the special permissions on a file or directory.

2. **Change special permissions in absolute mode by using the `chmod` command.**

   `% chmod` *nnnn filename*

nnnn	Specifies the octal values that change the permissions on the file or directory. The first octal value on the left sets the special permissions on the file. For the list of valid octal values for the special permissions, see Table 4–6.
filename	Specifies the file or directory.

   Note – If you use the `chmod` command to change the file group permissions on a file with ACL entries, both the file group permissions and the ACL mask are changed to the new permissions. Be aware that the new ACL mask permissions can change the permissions for additional users and groups who have ACL entries on the file. Use the `getfacl` command to make sure that the appropriate permissions are set for all ACL entries. For more information, see the `getfacl`(1) man page.

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

   `% ls -l` *filename*

Examples—Setting Special Permissions in Absolute Mode

In the following example, the `setuid` permission is set on the `dbprog` file.

```
# chmod 4555 dbprog
# ls -l dbprog
-r-sr-xr-x   1 db     staff        12095 May  6 09:29 dbprog
```

In the following example, the `setgid` permission is set on the `dbprog2` file.

```
# chmod 2551 dbprog2
# ls -l dbprog2
-r-xr-s--x   1 db     staff       24576 May  6 09:30 dbprog2
```

In the following example, the sticky bit permission is set on the `public_dir` directory.

```
# chmod 1777 public_dir
# ls -ld public_dir
drwxrwxrwt   2 ignatz   staff          512 May 15 15:27 public_dir
```

▼ How to Change Permissions in Symbolic Mode

Use the following procedure to change permissions in symbolic mode.

1. **If you are not the owner of the file or directory, become superuser.**

   Only the current owner or superuser can use the `chmod` command to change file permissions on a file or directory.

2. **Change permissions in symbolic mode by using the `chmod` command.**

   `% chmod` *who operator permission filename*

who operator permission	*who* specifies whose permissions are to be changed, *operator* specifies the operation to be performed, and *permission* specifies what permissions are to be changed. For the list of valid symbols, see Table 4–7.
filename	Specifies the file or directory.

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

   `% ls -l` *filename*

Examples—Changing Permissions in Symbolic Mode

In the following example, `read` permission are taken away from others.

```
% chmod o-r filea
```

In the following example, `read` and `execute` permissions are added for user, group, and others.

```
$ chmod a+rx fileb
```

In the following example, `read`, `write`, and `execute` permissions are assigned to group.

```
$ chmod g=rwx filec
```

Searching for Special Permissions

You should monitor your system for any unauthorized use of the `setuid` and `setgid` permissions on programs to gain superuser privileges. A suspicious listing grants ownership of such a program to a user rather than to `root` or `bin`.

▼ How to Find Files With `setuid` Permissions

Use the following procedure to find files with `setuid` permissions.

1. **Become superuser or assume an equivalent role.**
2. **Find files with `setuid` permissions by using the `find` command.**

   # **find** *directory* **-user root -perm -4000 -exec ls -ldb {} \; >/tmp/***filename*

`find` *directory*	Checks all mounted paths starting at the specified *directory*, which can be root (`/`), `sys`, `bin`, or `mail`.
`-user root`	Displays files owned only by `root`.
`-perm -4000`	Displays files only with permissions set to 4000.
`-exec ls -ldb`	Displays the output of the `find` command in `ls -ldb` format.
`>/tmp/`*filename*	Writes results to this file.

3. **Display the results in `/tmp/`*filename*.**

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

   If you need background information about `setuid` permissions, see "`setuid` Permission" on page 63.

Example—Finding Files With `setuid` Permissions

```
# find / -user root -perm -4000 -exec ls -ldb {} \; > /tmp/ckprm
# cat /tmp/ckprm
-r-sr-xr-x 1 root bin 38836 Aug 10 16:16 /usr/bin/at
```

```
-r-sr-xr-x 1 root bin 19812 Aug 10 16:16 /usr/bin/crontab
---s--x--x 1 root sys 46040 Aug 10 15:18 /usr/bin/ct
-r-sr-xr-x 1 root sys 12092 Aug 11 01:29 /usr/lib/mv_dir
-r-sr-sr-x 1 root bin 33208 Aug 10 15:55 /usr/lib/lpadmin
-r-sr-sr-x 1 root bin 38696 Aug 10 15:55 /usr/lib/lpsched
---s--x--- 1 root rar 45376 Aug 18 15:11 /usr/rar/bin/sh
-r-sr-xr-x 1 root bin 12524 Aug 11 01:27 /usr/bin/df
-rwsr-xr-x 1 root sys 21780 Aug 11 01:27 /usr/bin/newgrp
-r-sr-sr-x 1 root sys 23000 Aug 11 01:27 /usr/bin/passwd
-r-sr-xr-x 1 root sys 23824 Aug 11 01:27 /usr/bin/su
```

This output shows that a user named `rar` has made a personal copy of `/usr/bin/sh`, and has set the permissions as `setuid` to `root`. As a result, `rar` can execute `/usr/rar/bin/sh` and become the privileged user. If you want to save this output for future reference, move the file out of the `/tmp` directory.

Executable Stacks and Security

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

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

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

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

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

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

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

Because of hardware limitations, the capability of catching and reporting executable stack problems is only available on sun4m and sun4u platforms.

▼ How to Disable Programs From Using Executable Stacks

1. **Become superuser or assume an equivalent role.**
2. **Edit the `/etc/system` file and add the following line:**

   ```
   set noexec_user_stack=1
   ```

3. **Reboot the system.**

   ```
   # init 6
   ```

▼ How to Disable Executable Stack Message Logging

1. **Become superuser or assume an equivalent role.**
2. **Edit the `/etc/system` file and add the following line:**

   ```
   set noexec_user_stack_log=0
   ```

3. **Reboot the system.**

   ```
   # init 6
   ```

Using Access Control Lists (ACLs)

Traditional UNIX file protection provides read, write, and execute permissions for the three user classes: file owner, file group, and other. An ACL provides better file security by enabling you to define file permissions for the file owner, file group, other, specific users and groups, and default permissions for each of those categories.

For example, if you wanted everyone in a group to be able to read a file, you would simply give group read permissions on that file. Now, assume you wanted only one person in the group to be able to write to that file. Standard UNIX doesn't provide that level of file security. However, this dilemma is perfect for ACLs.

ACL entries are the way to define an ACL on a file, and they are set through the `setfacl` command. ACL entries consist of the following fields separated by colons:

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

entry-type	Is the type of ACL entry on which to set file permissions. For example, *entry-type* can be `user` (the owner of a file) or `mask` (the ACL mask). For a listing of ACL entries, see Table 4–8 and Table 4–9.
uid	Is the user name or user ID (UID).
gid	Is the group name or group ID (GID).
perms	Represents the permissions that are set on *entry-type*. *perms* can be indicated by the symbolic characters `rwx` or a number (the same permissions numbers that are used with the `chmod` command).

The following example shows an ACL entry that sets read and write permissions for the user `nathan`.

```
user:nathan:rw-
```

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

ACL Entries for Files

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

TABLE 4–8 ACL Entries for Files

ACL Entry	Description
`u[ser]::`*perms*	File owner permissions.
`g[roup]::`*perms*	File group permissions.
`o[ther]:`*perms*	Permissions for users other than the file owner or members of the file group.

TABLE 4–8 ACL Entries for Files *(Continued)*

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

ACL Entries for Directories

In addition to the ACL entries that are described in Table 4–8, you can set default ACL entries on a directory. Files or directories created in a directory that has default ACL entries will have the same ACL entries as the default ACL entries. Table 4–9 lists the default ACL entries for directories.

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

TABLE 4–9 Default ACL Entries for Directories

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

▼ How to Set an ACL on a File

1. **Set an ACL on a file by using the `setfacl` command.**

```
% setfacl -s user::perms,group::perms,other:perms,mask:perms,acl-entry-list  filename ...
```

`-s`	Sets an ACL on the file. If a file already has an ACL, it is replaced. This option requires at least the file owner, file group, and other entries.
`user::`*perms*	Specifies the file owner permissions.
`group::`*perms*	Specifies the file group permissions.
`other:`*perms*	Specifies the permissions for users other than the file owner or members of the file group.
`mask:`*perms*	Specifies the permissions for the ACL mask. The mask indicates the maximum permissions that are allowed for users (other than the owner) and for groups.
acl-entry-list	Specifies the list of one or more ACL entries to set for specific users and groups on the file or directory. You can also set default ACL entries on a directory. Table 4–8 and Table 4–9 show the valid ACL entries.
filename ...	Specifies one or more files or directories on which to set the ACL, separated by a space.

Caution – If an ACL already exists on the file, the `-s` option will replace the entire ACL with the new ACL.

For more information, see the `setfacl`(1) man page.

2. **Verify that an ACL was set on the file or verify which ACL entries were set on the file.**

```
% getfacl filename
```

For more information, see "How to Check If a File Has an ACL" on page 81.

Examples—Setting an ACL on a File

In the following example, the file owner permissions are set to read and write, file group permissions are set to read only, and other permissions are set to none on the `ch1.doc` file. In addition, the user `george` is given read and write permissions on the file, and the ACL mask permissions are set to read and write, which means that no user or group can have execute permissions.

```
% setfacl -s user::rw-,group::r--,other:---,mask:rw-,user:george:rw- ch1.doc
% ls -l
```

```
total 124
-rw-r-----+  1 nathan  sysadmin   34816 Nov 11 14:16 ch1.doc
-rw-r--r--   1 nathan  sysadmin   20167 Nov 11 14:16 ch2.doc
-rw-r--r--   1 nathan  sysadmin    8192 Nov 11 14:16 notes
% getfacl ch1.doc
# file: ch1.doc
# owner: nathan
# group: sysadmin
user::rw-
user:george:rw-    #effective:rw-
group::r--         #effective:r--
mask:rw-
other:---
```

In the following example, the file owner permissions are set to read, write, and execute, file group permissions are set to read only, other permissions are set to none, and the ACL mask permissions are set to read on the `ch2.doc` file. In addition, the user `george` is given read and write permissions. However, due to the ACL mask, the permissions for `george` are read only.

```
% setfacl -s u::7,g::4,o:0,m:4,u:george:7 ch2.doc
% getfacl ch2.doc
# file: ch2.doc
# owner: nathan
# group: sysadmin
user::rwx
user:george:rwx         #effective:r--
group::r--              #effective:r--
mask:r--
other:---
```

▼ How to Copy an ACL

- **Copy a file's ACL to another file by redirecting the `getfacl` output.**

```
% getfacl filename1 | setfacl -f -filename2
```

filename1	Specifies the file from which to copy the ACL.
filename2	Specifies the file on which to set the copied ACL.

Example—Copying an ACL

In the following example, the ACL on `ch2.doc` is copied to `ch3.doc`.

```
% getfacl ch2.doc | setfacl -f - ch3.doc
```

▼ How to Check If a File Has an ACL

- **Check if a file has an ACL by using the `ls` command.**

  `% ls -l` *filename*

  filename specifies the file or directory.

  In the output, a plus sign (+) to the right of the mode field indicates that the file has an ACL.

Note – Unless you have added ACL entries for additional users or groups on a file, a file is considered to be a "trivial" ACL and the plus sign (+) will not display.

Example—Checking If a File Has an ACL

The following example shows that the `ch1.doc` file has an ACL, because the listing has a plus sign (+) to the right of the mode field.

```
% ls -l ch1.doc
-rwxr-----+  1 nathan   sysadmin      167 Nov 11 11:13 ch1.doc
```

▼ How to Modify ACL Entries on a File

1. **Modify ACL entries on a file by using the `setfacl` command.**

   `% setfacl -m` *acl-entry-list filename ...*

`-m`	Modifies the existing ACL entry.
acl-entry-list	Specifies the list of one or more ACL entries to modify on the file or directory. You can also modify default ACL entries on a directory. Table 4–8 and Table 4–9 show the valid ACL entries.
filename ...	Specifies one or more files or directories, separated by a space.

2. **Verify that the ACL entries were modified on the file by using the `getfacl` command.**

   `% getfacl` *filename*

Examples—Modifying ACL Entries on a File

In the following example, the permissions for the user `george` are modified to read and write.

```
% setfacl -m user:george:6 ch3.doc
% getfacl ch3.doc
# file: ch3.doc
# owner: nathan
# group: staff
user::rw-
user::george:rw-         #effective:r--
group::r-                #effective:r--
mask:r--
other:r-
```

In the following example, the default permissions for the group `staff` are modified to read and the default ACL mask permissions are modified to read and write on the `book` directory.

```
% setfacl -m default:group:staff:4,default:mask:6 book
```

▼ How to Delete ACL Entries From a File

1. **Delete ACL entries from a file by using the `setfacl` command.**

   `% setfacl -d` *acl-entry-list filename ...*

`-d`	Deletes the specified ACL entries.
acl-entry-list	Specifies the list of ACL entries (without specifying the permissions) to delete from the file or directory. You can only delete ACL entries and default ACL entries for specific users and groups. Table 4–8 and Table 4–9 show the valid ACL entries.
filename ...	Specifies one or more files or directories, separated by a space.

   Alternately, you can use `setfacl -s` to delete all the ACL entries on a file and replace them with the new ACL entries that are specified.

2. **To verify that the ACL entries were deleted from the file, by using the `getfacl` command.**

   `% getfacl` *filename*

Example—Deleting ACL Entries on a File

In the following example, the user `george` is deleted from the `ch4.doc` file.

```
% setfacl -d user:george ch4.doc
```

▼ How to Display ACL Entries for a File

- **Display ACL entries for a file by using the `getfacl` command.**

```
% getfacl [-a | -d] filename ...
```

`-a`	Displays the file name, file owner, file group, and ACL entries for the specified file or directory.
`-d`	Displays the file name, file owner, file group, and default ACL entries for the specified directory.
filename ...	Specifies one or more files or directories, separated by a space.

If you specify multiple file names on the command line, the ACL entries are displayed with a blank line between each entry.

Examples—Displaying ACL Entries for a File

The following example shows all the ACL entries for the `ch1.doc` file. The `#effective:` note beside the user and group entries indicates what the permissions are after being modified by the ACL mask.

```
% getfacl ch1.doc

# file: ch1.doc
# owner: nathan
# group: sysadmin
user::rw-
user:george:r--         #effective:r--
group::rw-              #effective:rw-
mask:rw-
other:---
```

The following example shows the default ACL entries for the `book` directory.

```
% getfacl -d book

# file: book
# owner: nathan
# group: sysadmin
user::rwx
user:george:r-x         #effective:r-x
group::rwx              #effective:rwx
mask:rwx
other:---
default:user::rw-
default:user:george:r--
default:group::rw-
default:mask:rw-
default:other:---
```

CHAPTER 5

Role-Based Access Control (Overview)

This chapter describes role-based access control (RBAC), a security feature for controlling access to tasks that would normally be restricted to superuser. The following is a list of the overview information in this chapter.

For information on RBAC tasks, see Chapter 6. For reference information on the RBAC elements and tools, see Chapter 7.

RBAC: Replacing the Superuser Model

In conventional UNIX systems, the `root` user (also referred to as superuser) is all-powerful, with the ability to read and write to any file, run all programs, and send kill signals to any process. Effectively, anyone who can become superuser can modify a site's firewall, alter the audit trail, read payroll and other confidential records, and shut down the entire network.

Role-based access control (RBAC) is an alternative to the all-or-nothing superuser model. RBAC uses the security principle of least privilege, which is that no user should be given more privilege than necessary for performing his or her job. RBAC allows an organization to separate superuser's capabilities and assign them to special user accounts that are called *roles*. Roles can be assigned to specific individuals, according to their job needs.

The flexibility in setting up roles enables a variety of security policies. Three recommended roles that can be easily configured are available:

- **Primary Administrator –** A powerful role that is equivalent to `root`.
- **System Administrator –** A less strong role for administration that is not related to security. This role does not allow the user to set passwords.
- **Operator –** A junior administrator role for operations such as backups and restores, and printer management.

There is no requirement that these specific roles be implemented. Roles are a function of an organization's security needs. Roles can be set up for special-purpose administrators in areas such as security, networking, or firewall administration. Another strategy is to create a single strong administrator role along with an advanced user role for those users who are permitted to fix portions of their own systems.

Solaris RBAC Elements

In the RBAC model in the Solaris operating environment, users log in as themselves and assume roles that enable them to run restricted administration graphical tools and commands. The RBAC model introduces these elements to the Solaris operating environment:

- **Privileged Application –** An application that can override system controls and check for specific user IDs (UIDs), group IDs (GIDs), or authorizations (see "Privileged Applications" on page 88).
- **Role –** A special identity for running privileged applications that can be assumed by assigned users only.
- **Authorization –** A permission that can be assigned to a role or user (or embedded in a rights profile) for performing a class of actions that are otherwise prohibited by security policy.
- **Rights Profile –** A collection of overrides that can be assigned to a role or user. A rights profile can consist of authorizations, commands with `setuid` or `setgid` permissions (referred to as security attributes), and other rights profiles.

The following figure shows how the RBAC elements work together.

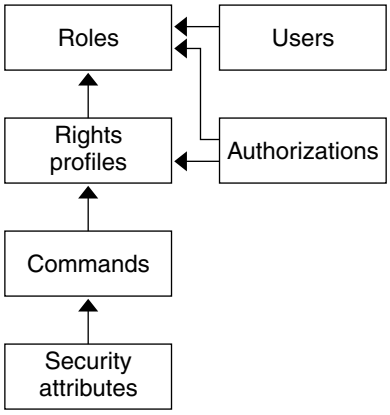


FIGURE 5–1 Solaris RBAC Element Relationships

In RBAC, users are assigned to roles. Roles get their capabilities from rights profiles and authorizations. Authorizations are generally assigned to the rights profiles with which they are logically associated but can be assigned directly to roles.

Note – Rights profiles and authorizations can also be assigned directly to users. This practice is discouraged because it enables users to make mistakes through inadvertent use of their privileges.

Commands with security attributes, that is, real or effective UIDs or GIDs, can be assigned to rights profiles.

The following figure uses the Operator role and the Printer Management rights profiles as examples to demonstrate RBAC relationships.

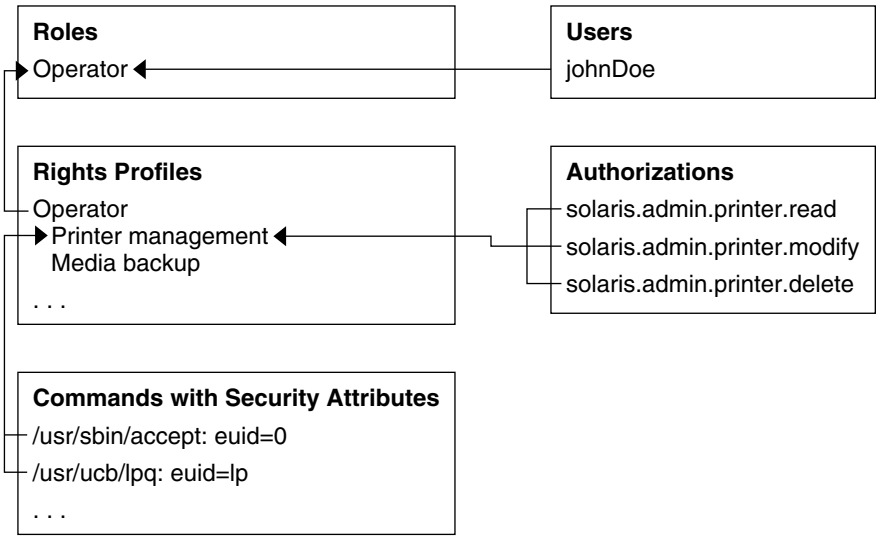


FIGURE 5–2 How Solaris RBAC Elements Relate

The Operator role is used to maintain printers and perform media backup. The user `johnDoe` is assigned to the Operator role and can assume it by supplying the Operator password.

The Operator rights profile has been assigned to the Operator role. The Operator rights profile has two supplementary profiles assigned to it, Printer Management and Media Backup, which reflect the Operator role's primary tasks.

The Printer Management rights profile is for managing printers, print daemons, and spoolers. Three authorizations are assigned to the Printer Management rights profile: `solaris.admin.printer.read`, `solaris.admin.printer.delete`, and `solaris.admin.printer.modify`. These authorizations allows users to manipulate information in the printer queue. The Printer Management profile also has a number of commands with security attributes that are assigned to it, such as `/usr/sbin/accept` with `euid=0` and `/usr/ucb/lpq` with `euid=lp`.

Privileged Applications

Applications that can override system controls are considered *privileged applications*.

Applications That Check UIDs and GIDs

Privileged applications that check for `root` or some other special UID or GID have long existed in UNIX. The RBAC rights profile mechanism enables you to specify the UID or GID for specific commands. Instead of changing the ID on a command that anyone can access, you can isolate the command with execution security attributes in the rights profile. A user or role with that rights profile can then run the program without having to become `root`.

IDs can be specified as real or effective. Assigning effective IDs is preferred over assigning real IDs. Effective IDs are equivalent to the `setuid` feature in the file permission bits and identify the UID for auditing. However, because some shell scripts and programs require a real UID of `root`, real IDs can be set as well. For example, the `pkgadd` command requires a real rather than an effective UID. If you encounter a command where the effective UID is not sufficient to run the command, you need to change the privilege to a real UID by using the Set Security Attributes option in the SMC Right Properties dialog boxes as described in "Creating or Changing a Rights Profile" on page 110.

Applications That Check Authorizations

RBAC additionally provides commands that check authorizations. By definition, `root` has all authorizations and thus can run any application. Currently, the applications that check for authorizations include the following:

- The entire Solaris Management Console suite of tools
- The batch job-related commands – `at`, `atq`, `batch`, and `crontab`
- Device-oriented commands – `allocate`, `deallocate`, `list_devices`, and `cdrw`.

Profile Shell

Authorized users can obtain privileged applications from the Solaris Management Console launcher or on the command line from a *profile shell*. A profile shell is a special kind of shell that enables access to the privileged applications that are assigned to the profile. Profile shells are launched when the user runs `su` to assume a role. The profile shells are `pfsh`, `pfcsh`, and `pfksh`. They correspond to Bourne shell (`sh`), C shell (`csh`), and Korn shell (`ksh`), respectively.

RBAC Roles

A *role* is a special type of user account from which you can run privileged applications. Roles are created in the same general manner as user accounts, with a home directory, groups, password, and so on. The capabilities of a role are a function of the rights profiles and authorizations that are assigned to it. Roles do not have inheritance.

When a user assumes a role, the role's attributes replace all user attributes. Role information is stored in the `passwd`, `shadow`, `user_attr`, and `audit_user` databases. For detailed information on setting up roles, see "Configuring Recommended Roles" on page 117, "Creating Roles" on page 105, and "Changing Role Properties" on page 108.

All users who can assume the same role have the same role home directory, operate in the same environment, and have access to the same files. Users can assume roles from the command line by running `su` and supplying the role name and password. Users can also assume a role when they open a Solaris Management Console tool.

Users cannot log in directly to a role. For this reason, it is useful to make `root` a role to prevent anonymous `root` login. See "Making `Root` a Role" on page 101. Users must log in to their user account first. A user cannot assume a role directly from another role. A user's real UID can always be audited.

No predefined roles are shipped with the Solaris 9 software. As stated earlier in this chapter, you can easily configure the three recommended roles.

RBAC Authorizations

An *authorization* is a discrete right that can be granted to a role or user. RBAC-compliant applications can check a user's authorizations prior to granting access to the application or specific operations within it. This check replaces the check in conventional UNIX applications for `UID=0`. For more information on authorizations, see "Authorizations" on page 122, "The `auth_attr` Database" on page 126, and "Commands That Require Authorizations" on page 131.

RBAC Rights Profiles

A *rights profile* is a collection of system overrides that can be assigned to a role or user. A rights profile can contain commands with effective or real UIDs or GIDs defined, authorizations, and other rights profiles. Rights profile information is split between the `prof_attr` and `exec_attr` databases. For more information on rights profiles, see "Contents of Rights Profiles" on page 118, "The `prof_attr` Database" on page 128, and "The `exec_attr` Database" on page 129.

Name Service Scope

Name service scope is an important concept for understanding RBAC. The scope in which a role can operate might apply to an individual host or to all hosts that are served by a name service such as NIS, NIS+, or LDAP. The precedence of local configuration files versus distributed databases is specified in the file `/etc/nsswitch.conf`. A lookup stops at the first match. For example, if a profile exists in two scopes, only the entries in the first scope are used.

CHAPTER 6

Role-Based Access Control (Tasks)

This chapter covers tasks that you can use to manage RBAC elements. The following is a list of the task maps in this chapter. To find the tasks for the initial setup of RBAC, see "Configuring RBAC (Task Map)" on page 94. For general management of the RBAC elements, see Chapter 5.

The following is a list of information in this chapter:

The preferred method for performing RBAC-related tasks is through the Solaris Management Console. The console tools for managing the RBAC elements are contained in the User Tool Collection.

You can also operate on local files with the Solaris Management Console command-line interface and other command-line interfaces. The Solaris Management Console commands require authentication to connect to the server. As a result, they are not practical in scripts. The other commands require superuser or a role, and cannot be applied to databases in a name service.

Tip – Another drawback to using the command line to manage RBAC information is that the edit may not take effect immediately. To enable the edit, you need to stop and restart the name service cache daemon, `nscd`(1M).

Configuring RBAC (Task Map)

Task	Description	For Instructions
1. Plan for RBAC	Learn the concepts behind RBAC, examine your site's security needs, and plan how to integrate RBAC into your operation.	"How to Plan Your RBAC Implementation" on page 94
2. Start the User tools from the Solaris Management Console	All RBAC tasks can be performed by the User tools.	"How to Run the User Tool Collection" on page 96
3. Install initial users if needed	One or more existing users must be available for assignment to the first role.	"How to Create Initial Users by Using the User Accounts Tool" on page 97
4. Install the first role	The first role, typically Primary Administrator, needs to be installed by `root` user.	"How to Run the User Tool Collection" on page 96
5. (Optional) Make `root` a role	To eliminate anonymous `root` login, `root` can be made a role.	"How to Make `Root` a Role" on page 101

Planning for RBAC

RBAC can be an integral part of how an organization manages its information resources. Planning requires a thorough knowledge of the RBAC capabilities as well as the security requirements of the organization.

▼ How to Plan Your RBAC Implementation

1. **Learn the basic RBAC concepts.**

   Read Chapter 5. Using RBAC to administer a system is very different from using conventional UNIX. You should be familiar with the RBAC concepts before you start your implementation. For greater detail, see Chapter 7.

2. **Examine your security policy.**

   Your organization's security policy should detail the potential threats to your system, measure the risk of each threat, and have a strategy to counter these threats. Isolating the security-relevant tasks through RBAC can be a part of the strategy. Although you

can install the suggested roles and their configurations as is, you might need to customize your RBAC configuration to adhere to your security policy.

3. **Decide how much RBAC your organization needs.**

   Depending on your security needs, you can use varying degrees of RBAC, as follows:

   - **No RBAC** – You can perform all tasks as `root` user. In this instance, you log in as yourself and when you select a console tool, you type **`root`** as the user.
   - **Root as a Role** – This method eliminates anonymous `root` logins by preventing all users from logging in as `root`. Instead, they must log in as normal users prior to assuming the `root` role. See "Making `Root` a Role" on page 101.
   - **Single Role Only** – This method adds the Primary Administrator role only and is similar to the superuser model.
   - **Suggested Roles** – Three suggested roles that can be easily configured are available: Primary Administrator, System Administrator, and Operator. These roles are suitable for organizations with administrators at different levels of responsibility whose job capabilities fit the suggested roles.
   - **Custom Roles** – You can create your own roles to meet the security requirements of your organization. The new roles can be based on existing or customized rights profiles.

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

   Review the capabilities of the suggested roles and default rights profiles. Three rights profiles are available for configuring the suggested roles:

   - **Primary Administrator rights profile** – For creating a role that can perform all administrative tasks, granting rights to others, and editing rights that are associated with administrative roles. A user in this role can assign the Primary Administrator role and the ability to grant rights to other users.
   - **System Administrator rights profile** – For creating a role that can perform most nonsecurity administrative tasks. For example, the System Administrator can add new user accounts, but cannot set passwords or grant rights to other users.
   - **Operator rights profile** – For creating a role that can perform simple administrative tasks, such as backup and restore, and printer maintenance.

   These rights profiles enable administrators to configure the suggested roles by using a single rights profile instead of having to mix and match rights profiles.

   To further examine rights profiles, use the Rights tool to display the contents. You can also refer to "Contents of Rights Profiles" on page 118 for a summary of some typical rights profiles. With the console tools, you can customize the roles and rights profiles that are provided to meet the needs of your organization.

5. **Decide if any additional roles or rights profiles are appropriate for your organization.**

   Look for other applications or families of applications at your site that might benefit from restricted access. Applications that affect security, that can cause denial-of-service problems, or that require special administrator training are good candidates for RBAC.

a. **Determine which commands are needed for the new task.**

b. **Decide which rights profile is appropriate for this task.**

   Check if an existing rights profile can handle this task or if a separate rights profile needs to be created.

c. **Determine which role is appropriate for this rights profile.**

   Decide if the rights profile for this task should be assigned to an existing role or if a new role should be created. If you use an existing role, check that the other rights profiles are appropriate for users who are assigned to this role.

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

   According to the principle of least privilege, you should assign users to roles that are appropriate to their level of trust. Keeping users away from tasks that they do not need to use reduces potential problems.

First-Time Use of the User Tool Collection

To install the initial users to their assigned roles, you first log in as yourself. When you authenticate yourself to the Solaris Management Console, specify `root` user.

▼ How to Run the User Tool Collection

1. **Log in as a normal user and start the Solaris Management Console.**

```
% whoami
johnDoe
% /usr/sadm/bin/smc&
```

2. **Navigate to the User Tool Collection and click the icon, as follows:**

   a. **Find the icon that is labeled This Computer under Management Tools in the navigation pane.**

   b. **Click the turner icon to its left.**

      The turner icon is shaped like a lever. When the lever is horizontal, the contents of the folder are hidden. When the lever is vertical, the contents are displayed. Clicking the turner icon toggles the folder between the hidden and displayed states.

   c. **Click the turner icon next to the System Configuration folder to display its contents.**

d. Click the User icon to open the User Tool Collection.

The user login dialog box is displayed.

3. Type `root` and the `root` password in the Login: User Name dialog box. Click OK.

Generally, you should type your user name here and then assume a role. However, for the first time, you need to be `root` user because no roles exist yet. This step opens the User Tool Collection (see the following figure).

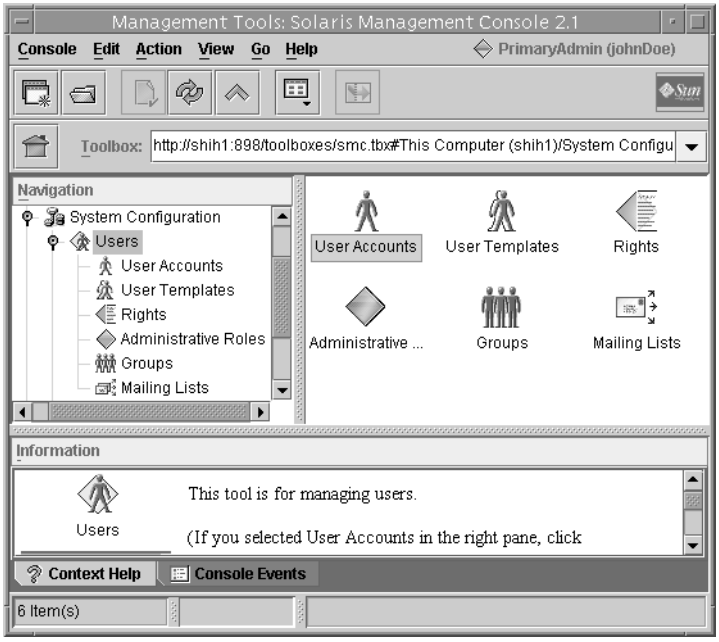


FIGURE 6–1 The User Tool Collection

Setting Up Initial Users

If all users who are assigned to roles are already installed on this system, you can skip this task and go to "Setting Up Initial Roles" on page 99.

▼ How to Create Initial Users by Using the User Accounts Tool

1. Click the User Accounts tool icon in either the navigation pane or the view pane of the User Tool Collection.

The User Accounts tool is started. The Action menu now provides options for this tool.

2. **Select Add User->With Wizard from the Action menu.**

   This step starts the Add User wizard, a series of dialog boxes that request information that is necessary for configuring a user. Use the Next and Back buttons to navigate between dialog boxes. Note that the Next button does not become active until all required fields have been filled in. The last dialog box is for reviewing the entered data, at which point you can go back to change entries or click Finish to save the new role.

   The following figure shows the first dialog box, Step 1: Enter a user name.

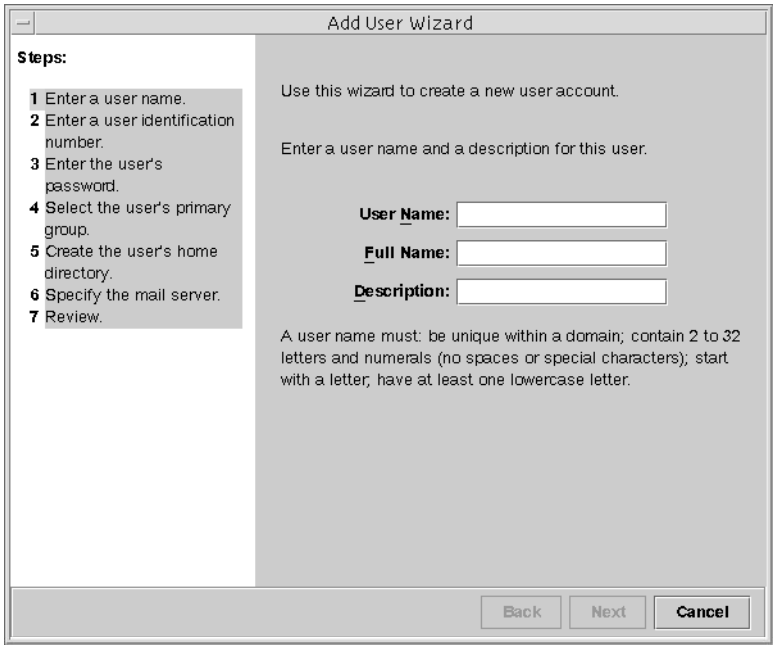


FIGURE 6–2 Add User Wizard

3. **Type the name of the first user and the other identification information.**

4. **In the Step 2: Enter a User Identification Number dialog box, type the UID.**

   This entry should match the existing UID for the user.

5. **In the Step 3: Enter the User's Password dialog box, indicate whether you or the user will be setting the password.**

   If you are setting up this account for yourself, click the second option. Then, type and confirm your password.

6. **In the Step 4: Select the User's Primary Group dialog box, select the appropriate group.**

7. **In the Step 5: Create the User's Home Directory dialog box, specify the path for the home directory.**
8. **In the Step 6: Specify the Mail Server dialog box, check out the default mail server and mailbox.**

   You can change these settings later in the User Properties dialog box.
9. **Check the information in the Review dialog box. Click Finish to save, or click Back to reenter information.**

   If you discover missing or incorrect information, click the Back button repeatedly to display the dialog box where the incorrect information is displayed. Then, click Next repeatedly to return to the Review dialog box.

Setting Up Initial Roles

The first role to create is the role that is responsible for managing users and roles, typically the Primary Administrator. First, you should install the users and the roles on your local host. After you have set up a toolbox for the name service scope, you need to create the same users and roles in the name service. See "Using the Solaris Management Tools in a Name Service Environment (Task Map)" in *System Administration Guide: Basic Administration*. After the first role is established and assigned to you, then you can run the console tools by assuming a role instead of becoming `root`.

▼ How to Create the First Role (Primary Administrator) by Using the Administrative Roles Tool

To install the first role, you should log in as yourself. When you authenticate yourself to the Solaris Management Console, specify `root` user. You should first install the role on your local host. After the first role is established and assigned to you, you can run the console tools by assuming a role instead of as `root` user.

1. **Type `root` and the `root` password in the Login: User Name dialog box. Click OK.**
2. **Click the Administrative Roles icon in either the navigation pane or the view pane of the User Tool Collection.**

   The Administrative Roles tool is started. The Action menu now provides options for this tool.
3. **Select Add Administrative Role from the Action menu.**

This step starts the Add Administrative Role wizard, a series of dialog boxes that request information that is necessary for configuring a role. Use the Next and Back buttons to navigate between dialog boxes. Note that the Next button does not become active until all required fields have been filled in. The last dialog box is for reviewing the entered data, at which point you can go back to change entries or click Finish to save the new role.

The following figure shows the first dialog box, Step 1: Enter a Role Name.

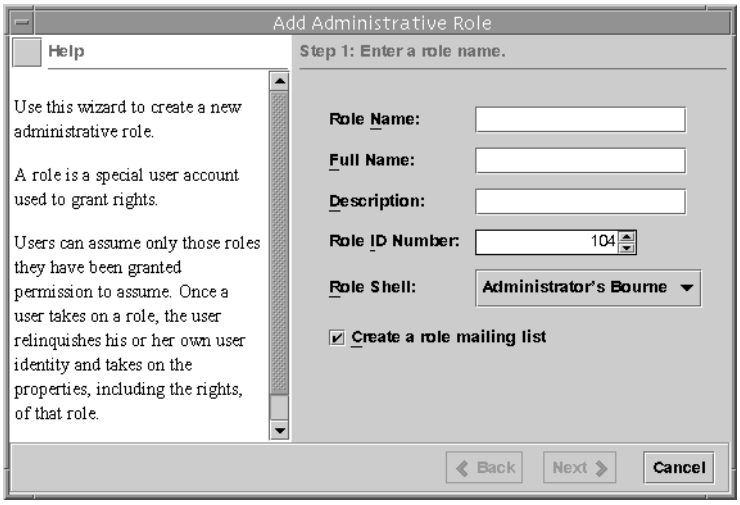


FIGURE 6–3 Add Administrative Role Wizard

4. **Type `primaryadmin`, or whatever role name you are using, and the other identification information.**

   If you select the role mailing list option, you can create an alias of users who can assume this role.

5. **In the Step 2: Enter a Role Password dialog box, type the password for the new role in the Role Password field and again in the Confirm Password field.**

   Confirmation helps prevent a misspelled password from being saved.

6. **In the Step 3: Enter Role Rights dialog box, select the Primary Administrator rights profile.**

   Double-click the Primary Administrator rights profile in the Available Rights column (on the left). The rights profiles in the Granted Rights column (on the right) are the rights profiles that are assigned to this role. In this instance, only the Primary Administrator rights profile is needed.

7. **In the Step 4: Select a Home Directory dialog box, specify the server and path for the home directory.**

8. **In the Step 5: Assign Users to This Role dialog box, type the login names for any users to be assigned to the Primary Administrator role.**

   Any users that you add must be defined in the same name service scope in which you are working. If you selected the role mailing list option in the Step 1: Enter a Role Name dialog box, these users will receive email that is addressed to the Primary Administrator role.

9. **Check the information in the Review dialog box. Click Finish to save, or click Back to reenter information.**

   If you discover missing or incorrect information, click the Back button repeatedly to display the dialog box where the incorrect information is displayed. Then, click Next repeatedly to return to the Review dialog box.

10. **Open a terminal window, become root, and start and stop the name service cache daemon.**

    The new role does not take effect until the name service cache daemon is restarted. After becoming root, type as follows:

    ```
    # /etc/init.d/nscd stop
    # /etc/init.d/nscd start
    ```

Making `Root` a Role

This procedure shows how to change `root` from a user to a role within a local scope. Changing `root` to a role prevents users from logging in to that server directly as `root`. Users must first log in as themselves so their UIDs are available for auditing.

Caution – If you make `root` a role without assigning it to a valid user or without a currently existing role equivalent to `root`, no one can become `root`.

▼ How to Make `Root` a Role

1. **Log in to the target server.**

2. **Become superuser.**

3. **Edit the `/etc/user_attr` file.**

   Here is an excerpt from a typical `user_attr` file.

   ```
   root::::type=normal;auths=solaris.*,solaris.grant;profiles=All
   johnDoe::::type=normal
   ```

4. **Check that your name is in the file.**
5. **Add root to the roles that are assigned to your record.**

   Assign the `root` role to any applicable users. If you intend to use `primaryadmin` as your most powerful role, you do not have to assign `root` to any users.

   ```
   johnDoe::::type=normal;roles=root
   ```

6. **Go to the root record in the file and change `type=normal` to `type=root`.**

   ```
   root::::type=role;auths=solaris.*,solaris.grant;profiles=All
   ```

7. **Save the file.**

Managing RBAC Information (Task Map)

The following task map shows where to obtain information for performing specific RBAC tasks.

Task	Description	For Instructions
Use privileged applications	To run applications that can affect security or system operations requires becoming superuser or assuming a role.	"How to Assume a Role in the Console Tools" on page 104 "How to Assume a Role at the Command Line" on page 103
Create roles	To add new roles, that is, special identities for running privileged applications.	"How to Create a Role by Using the Administrative Roles Tool" on page 105 "How to Create a Role From the Command Line" on page 106
Change role properties	To change the properties of a role, that is, the assigned users, rights profiles, and authorizations that are assigned to a role.	"How to Change a Role by Using the Administrative Roles Tool" on page 108 "How to Change a Role From the Command Line" on page 109
Create or change rights profiles	To add or change a rights profile, including the assignment of authorizations, commands with security attributes, and supplementary rights profiles.	"How to Create or Change a Rights Profile by Using the Rights Tool" on page 110 "How to Change Rights Profiles From the Command Line" on page 113

Task	Description	For Instructions
Change a user's RBAC properties	To change the roles, rights profiles, or authorizations that are assigned to a user.	"How to Modify a User's RBAC Properties by Using the User Accounts Tool" on page 114 "How to Modify a User's RBAC Properties From the Command Line" on page 114
Secure legacy applications	To turn on the set ID permissions for legacy applications. Scripts can contain commands with set IDs. Legacy applications can check for authorizations, if appropriate.	"How to Add Security Attributes to a Legacy Application" on page 115 "How to Add Security Attributes to Commands in a Script" on page 115 "How to Check for Authorizations in a Script or Program" on page 115

These procedures manage the elements that are used in role-based access control (RBAC). For user management procedures, refer to "Managing User Accounts and Groups (Tasks)" in *System Administration Guide: Basic Administration*.

Using Privileged Applications

To run privileged applications, you must first become superuser or assume a role. Although running privileged applications as a normal user is possible, it is discouraged to avoid errors that are caused by users who inadvertently exercise this privilege.

▼ How to Assume a Role at the Command Line

1. **Use the `su` command as follows:**

   ```
   % su my-role
   Password: my-role-password
   #
   ```

   Typing `su` by itself lets you become superuser. Typing `su` with a role name lets you assume that role (if it has been assigned to you). You must supply the appropriate password. Assuming a role switches the command line to the profile shell for that role. The profile shell has been modified to run commands with the security attributes that are assigned in the role's rights profiles.

2. **Type a command in the shell.**

   The command is executed with any assigned security attributes and `setuid` or `setgid` permissions.

▼ How to Assume a Role in the Console Tools

1. **Start the Solaris Management Console.**

   Use one of the following methods:

   - Type `smc` at the command line.
   - Click the Solaris Management Console icon in the Tools subpanel.
   - Double-click the Solaris Management Console icon in the Application Manager.

   All Solaris Management Console tools have extensive context-sensitive help that document each field. In addition, you can access various help topics from the Help menu. Note that it does not matter whether you are logged in as `root` or as a normal user when you start the console.

2. **Select the toolbox for your task.**

   Navigate to the toolbox that contains the tool or collection in the appropriate scope and click the icon. The scopes are files (local), NIS, NIS+, and LDAP. If the appropriate toolbox is not displayed in the navigation pane, choose Open Toolbox from the Console menu and load the relevant toolbox.

3. **Select the tool.**

   Navigate to the tool or collection to be used and click the icon. The tools for managing the RBAC elements are all part of the User Tool Collection.

4. **Authenticate yourself in the Login: User Name dialog box.**

   Your choices are the following:

   - Type your user name and password to assume a role or to operate as a normal user.
   - Type `root` and the `root` password to operate as superuser.

   Note that if you have not yet set up any roles or if the roles that are set up cannot perform the appropriate tasks, you need to log in as `root`. If you authenticate yourself as `root` (or as a user with no roles assigned), the tools are loaded into the console and you can proceed to Step 6.

5. **Authenticate yourself in the Login: Role dialog box.**

   The Role option menu in the dialog box displays the roles that are assigned to you. Choose a role and type the role password. If you are to operate as a normal user, type your user name and password.

6. **Navigate to the tool to be run and click the icon.**

Creating Roles

To create a role, you must either assume a role that has the Primary Administrator rights profile assigned to it or run as `root` user. See "RBAC Roles" on page 90 and "Configuring Recommended Roles" on page 117 to learn more about roles.

▼ How to Create a Role by Using the Administrative Roles Tool

1. **Start the Administrative Roles tool.**

   Run the Administrative Roles tool, start the Solaris Management Console, as described in "How to Assume a Role in the Console Tools" on page 104. Then, open the User Tool Collection, and click the Administrative Roles icon.

2. **Start the Add Administrative Role wizard.**

   Select Add Administrative Role from the Action menu to start the Add Administrative Role wizard for configuring roles.

3. **Fill in the fields in the series of dialog boxes. Click Finish when done.**

   Use the Next and Back buttons to navigate between dialog boxes. Note that the Next button does not become active until all required fields have been filled in. The last dialog box is for reviewing the entered data, at which point you can go back to change entries or click Finish to save the new role. Table 6–1 summarizes the dialog boxes.

4. **Open a terminal window, become `root`, and start and stop the name service cache daemon.**

   The new role does not take effect until the name service cache daemon is restarted. After becoming `root`, type as follows:

   ```
   # /etc/init.d/nscd stop
   # /etc/init.d/nscd start
   ```

TABLE 6–1 Add Administrative Role Wizard: Dialog Boxes and Fields

Dialog Box	Fields	Field Description
Step 1: Enter a role name	Role Name	Short name of the role.
	Full Name	Long version of the name.
	Description	Description of the role.

TABLE 6–1 Add Administrative Role Wizard: Dialog Boxes and Fields *(Continued)*

Dialog Box	Fields	Field Description
	Role ID Number	UID for the role, automatically incremented.
	Role Shell	The profile shells that are available to roles: Administrator's C, Administrator's Bourne, or Administrator's Korn shell.
	Create a role mailing list	Makes a mailing list for users who are assigned to this role.
Step 2: Enter a role password	Role Password	********
	Confirm Password	********
Step 3: Select role rights	Available Rights / Granted Rights	Assigns or removes a role's rights profiles.
		Note that the system does not prevent you from typing multiple occurrences of the same command. The attributes that are assigned to the first occurrence of a command in a rights profile have precedence and all subsequent occurrences are ignored. Use the Up and Down arrows to change the order.
Step 4: Select a home directory	Server	Server for the home directory.
	Path	Home directory path.
Step 5: Assign users to this role	Add	Adds users who can assume this role. Must be in the same scope.
	Delete	Deletes users who are assigned to this role.

▼ How to Create a Role From the Command Line

1. **Become superuser or assume a role that is capable of creating other roles.**
2. **Select a method for creating a role:**
   - For roles in the local scope, use the `roleadd` command to specify a new local role and its attributes.
   - Alternatively, for roles in the local scope, edit the `user_attr` file to add a user with type=role.

This method is recommended for emergencies only, as it is easy to make mistakes while you are typing.

- For roles in a name service, use the `smrole` command to specify the new role and its attributes.

  This command requires authentication by superuser or a role that is capable of creating other roles. You can apply the `smrole` to all name services. This command runs as a client of the Solaris Management Console server.

3. **Start and stop the name service cache daemon.**

   New roles do not take effect until the name service cache daemon is restarted. As `root`, type as follows:

   ```
   # /etc/init.d/nscd stop
   # /etc/init.d/nscd start
   ```

EXAMPLE 6–1 Creating a Custom Operator Role by Using the `smrole` Command

The following sequence demonstrates how a role is created with the `smrole` command. In this example, a new version of the Operator role is created that has assigned to it the standard Operator rights profile and the Media Restore rights profile.

```
% su primaryadmin
# /usr/sadm/bin/smrole add -H myHost -- -c "Custom Operator" -n oper2 -a johnDoe \
-d /export/home/oper2 -F "Backup/Restore Operator" -p "Operator" -p "Media Restore"
Authenticating as user: primaryadmin

Type /? for help, pressing <enter> accepts the default denoted by [ ]
Please enter a string value for: password ::     <type primaryadmin password>

Loading Tool: com.sun.admin.usermgr.cli.role.UserMgrRoleCli from myHost
Login to myHost as user primaryadmin was successful.
Download of com.sun.admin.usermgr.cli.role.UserMgrRoleCli from myHost was successful.

Type /? for help, pressing <enter> accepts the default denoted by [ ]
Please enter a string value for: password ::    <type oper2 password>

# /etc/init.d/nscd stop
# /etc/init.d/nscd start
```

To view the newly created role (and any other roles), use `smrole` with the `list` subcommand, as follows:

```
# /usr/sadm/bin/smrole list --
Authenticating as user: primaryadmin

Type /? for help, pressing <enter> accepts the default denoted by [ ]
Please enter a string value for: password ::     <type primaryadmin password>

Loading Tool: com.sun.admin.usermgr.cli.role.UserMgrRoleCli from myHost
Login to myHost as user primaryadmin was successful.
Download of com.sun.admin.usermgr.cli.role.UserMgrRoleCli from myHost was successful.
root                    0               Super-User
```

EXAMPLE 6–1 Creating a Custom Operator Role by Using the `smrole` Command *(Continued)*

```
primaryadmin            100             Most powerful role
sysadmin                101             Performs non-security admin tasks
oper2                   102             Backup/Restore Operator
```

Changing Role Properties

To change a role, you must either assume a role that has the Primary Administrator rights profile assigned to it, or run the User Tool Collection as `root` user if roles have not yet been set up.

▼ How to Change a Role by Using the Administrative Roles Tool

1. **Start the Administrative Roles tool.**

   To run the Administrative Roles tool, you need to start the Solaris Management Console, as described in "How to Assume a Role in the Console Tools" on page 104. Then, open the User Tool Collection, and click the Administrative Roles icon.

   After the Administrative Roles tool starts, the icons for the existing roles are displayed in the view pane.

2. **Click the role to be changed and select the appropriate item from the Action menu, as follows:**

   - To change users who are assigned to a role, select Assign Administrative Role.

     The Assign Administrative Role dialog box is displayed. The Assign Administrative Role dialog box is a modified version of the Role Properties dialog box and has a Users tab only. Use the Add field to assign a user in the current scope to this role. Use the Delete field to remove a user's role assignment. Click OK to save.

   - To change rights that are assigned to a role, select Assign Rights to Role.

     The Assign Rights to Role dialog box is displayed. The Assign Rights to Role dialog box is a modified version of the Role Properties dialog box and has a Rights tab only. Use the Available Rights and Granted Rights columns to add or remove rights profiles for the selected role. Click OK to save.

   - To change any of the role's properties, select Properties (or simply double-click the role icon).

The Role Properties dialog box is displayed, which provides access to all role properties (see the following figure and table). Use the tabs to navigate to any information to be changed, make your changes, and click OK to save.

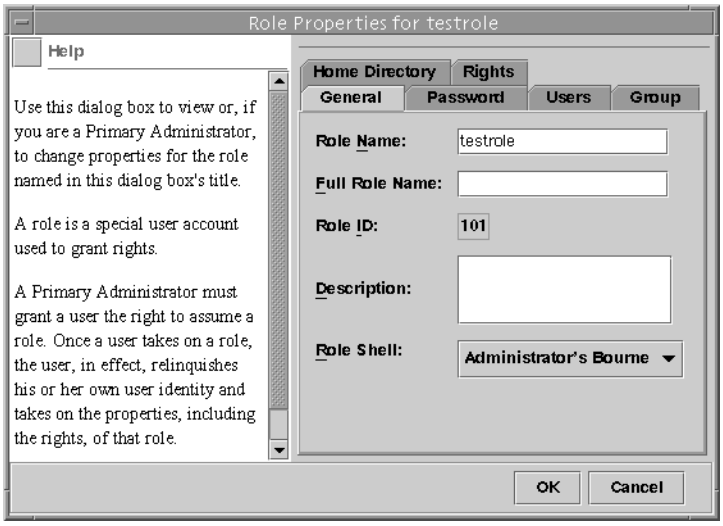


FIGURE 6–4 Role Properties Dialog Box

TABLE 6–2 Role Properties Summary

Tab	Tab Description
General	Specifies the role identification information and the default login shell.
Password	Specifies the role password.
Users	Specifies the users who are assigned to the role.
Group	Sets the role's primary groups and secondary groups for the purpose of accessing and creating files and directories.
Home Directory	Specifies the role's home directory, home directory server, automounting, and directory access.
Rights	Allows rights profiles to be assigned to the role. The precedence of the assigned rights profiles can be changed here.

▼ How to Change a Role From the Command Line

1. **Become superuser or assume a role that is capable of changing other roles.**
2. **Use the command that is appropriate for the task:**

- Use the `rolemod` command to modify the attributes of a role that are defined locally.
- Use the `roledel` command to delete a role that is defined locally.
- Edit the `user_attr` file to change the authorizations or rights profiles that are assigned to a local role.

  This method is recommended for emergencies only, as it is easy to make a mistake while you are typing.
- Use the `smrole` command to modify the attributes of a role in a name service.

  This command requires authentication as superuser or as a role that is capable of changing other roles. The `smrole` command runs as a client of the Solaris Management Console server.

Creating or Changing a Rights Profile

To create or change a rights profile, you must either assume a role that has the Primary Administrator rights profile assigned to it, or run the User Tool Collection as `root` user if roles have not yet been set up. To learn more about rights profiles, see "RBAC Roles" on page 90 and "Configuring Recommended Roles" on page 117.

▼ How to Create or Change a Rights Profile by Using the Rights Tool

1. **Start the Rights tool.**

   To run the Rights tool, you need to start the Solaris Management Console as described in "How to Assume a Role in the Console Tools" on page 104. Then, open the User Tool Collection, and click the Rights icon.

   After the Rights tool starts, the icons for the existing rights profiles are displayed in the view pane.

2. **Take the appropriate action for creating or changing a rights profile:**

   - To create a new rights profile, select Add Right from the Action menu.
   - To change an existing rights profile, click the rights profile icon and select Properties from the Action menu (or simply double-click the rights profile icon).

   Both actions display a version of the Rights Properties dialog box. The Add Right version (which follows) has a writable Name field. The standard Rights Properties dialog box has a read-only Name field because the name of a rights profile cannot be changed after it has been defined.

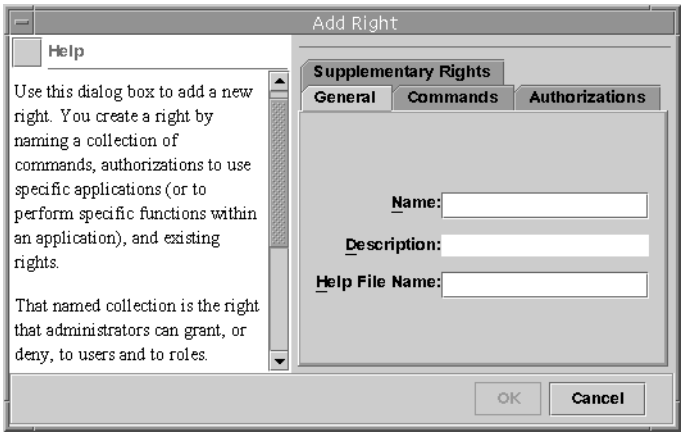


FIGURE 6–5 Add Right Dialog Box

3. **Type the new information. Click OK to save the rights profile.**
   The following table lists the tabs and fields in the Right Properties dialog box.

Tab	Field	Field Description
General	Name	Name of the new rights profile.
	Description	Description of the new rights profile.
	Help File Name	Name of the HTML help file for the new rights profile.
Commands	Add Directory	Opens a dialog box for adding directories that are not already in the Commands Denied or Commands Permitted columns.
	Commands Denied / Commands Permitted	Assigns or removes a rights profile's commands.
	Set Security Attributes	Opens a dialog box for assigning or removing a command's security attributes, that is, real or effective UIDs or GIDs (see Figure 6–6). **Note –** Assigning effective IDs is preferred over assigning real IDs. Use real IDs only when they are required by the command, such as `pkgadd`.
	Find (command)	Searches the two command lists for the specified string.
Authorizations	Authorizations Excluded / Authorizations Included	Assigns or removes a rights profile's authorizations.

Tab	Field	Field Description
Supplementary Rights	Rights Excluded / Rights Included	Assigns or removes a rights profile's supplementary rights profiles.

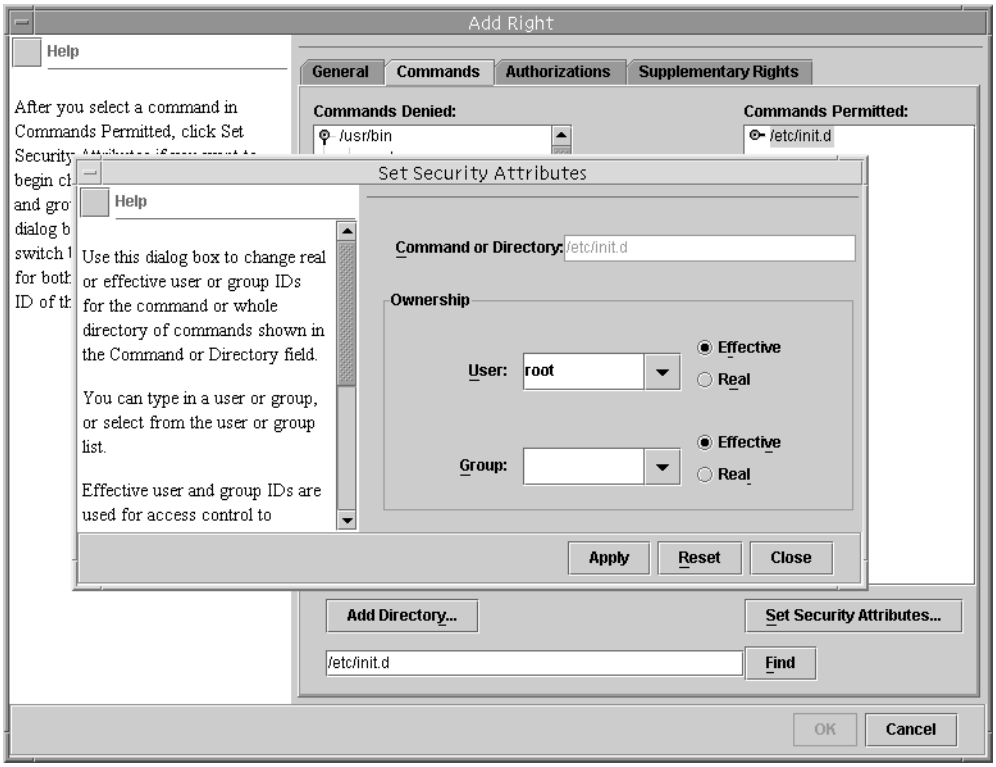


FIGURE 6–6 Adding Security Attributes to Commands

EXAMPLE 6–2 Creating a New Rights Profile With the Rights Tool

The data in the following table shows sample data for a hypothetical rights profile that is called "Restart" could be created. The example rights profile, Restart, has the commands in the subdirectory `/etc/init.d` assigned to it. These commands have an effective UID of 0. This rights profile would be useful for administrators who are permitted to stop and start the daemons in `/etc/init.d`.

Tab	Field	Example
General	Name	Restart

EXAMPLE 6–2 Creating a New Rights Profile With the Rights Tool *(Continued)*

Tab	Field	Example
	Description	For starting and stopping daemons in `/etc/init.d`
	Help File Name	`Restart.html`
Commands	Add Directory	Click Add Directory, type `/etc/init.d` in the dialog box, and click OK.
	Commands Denied / Commands Permitted	Select `/etc/init.d` and click Add to move the command to the Commands Permitted column.
	Set Security Attributes	Select `/etc/init.d`, click Set Security Attributes, and set Effective UID = root (see Figure 6–6).
	Find (command)	
Authorizations	Authorizations Excluded / Authorizations Included	
Supplementary Rights	Rights Excluded / Rights Included	

▼ How to Change Rights Profiles From the Command Line

1. **Become superuser or assume a role with the PrimaryAdministration rights profile.**
2. **Use the subcommand of `smprofile` that is appropriate for the task.**

   This command requires authentication. You can apply the command to all name services. `smprofile` runs as a client of the Solaris Management Console server.

   - To add a new profile, use `smprofile` with the `add` subcommand.
   - To change an existing profile, use `smprofile` with the `modify` subcommand.

Modifying a User's RBAC Properties

To modify a user's properties, you must either be running the User Tool Collection as `root` user or assume a role that has the Primary Administrator rights profile assigned to it.

▼ How to Modify a User's RBAC Properties by Using the User Accounts Tool

1. **Start the User Accounts tool.**

   To run the User Accounts tool, you need to start the Solaris Management Console, as described in "How to Assume a Role in the Console Tools" on page 104. Then, open the User Tool Collection, and click the User Accounts icon.

   After the User Accounts tool starts, the icons for the existing user accounts are displayed in the view pane.

2. **Click the user account icon to be changed and select Properties from the Action menu (or simply double-click the user account icon).**

3. **Click the appropriate tab in the dialog box for the property to be changed, as follows:**

   - To change the roles that are assigned to the user, click the Roles tab and move the role assignment to be changed to the appropriate column: Available Roles or Assigned Roles.
   - To change the rights profiles that are assigned to the user, click the Rights tab and move it to the appropriate column: Available Rights or Assigned Rights.

   Note – It is not good practice to assign rights profiles directly to users. The preferred approach is to force users to assume roles in order to perform privileged applications. This strategy avoids the possibility of normal users abusing privileges.

▼ How to Modify a User's RBAC Properties From the Command Line

1. **Become superuser or assume a role that can modify user files.**

2. **Use the appropriate command:**

   - To change the authorizations, roles, or rights profiles that are assigned to a user who is defined in the local scope, use the `usermod` command.
   - Alternatively, to change the authorizations, roles, or rights profiles that are assigned to a user who is defined in the local scope, edit the `user_attr` file.

     This method is recommended for emergencies only, as it is easy to make a mistake while you are typing.
   - To change the authorizations, roles, or rights profiles that are assigned to a user who is defined in a name service, use the `smuser` command.

This command requires authentication as superuser or as a role that is capable of changing user files. You can apply `smuser` to all name services. `smuser` runs as a client of the Solaris Management Console server.

Securing Legacy Applications

This section discusses how to make legacy applications more secure. To add legacy applications to the Solaris Management Console, see "Adding Tools to the Solaris Management Console" in *System Administration Guide: Basic Administration*.

How to Add Security Attributes to a Legacy Application

You add security attributes to a legacy application in the same way as you would for any command. You need to add the command (or its directory) to the Commands Denied column in the Commands tab of the Rights property dialog box. Then move the command to the Commands Permitted column.

How to Add Security Attributes to Commands in a Script

If a command in a script needs to have the setUID bit set to run, simply add the security attributes to that command in the same rights profile. See "How to Create or Change a Rights Profile by Using the Rights Tool" on page 110.

How to Check for Authorizations in a Script or Program

To have a script for authorizations, you need to add a test that is based on the `auths` command (see the `auths`(1) man page). For example, the following line would test if the user has the authorization entered as the `$1` argument:

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

To be more complete, the test should include logic that checks for other authorizations that use wildcards. For example, to test if the user has the `solaris.admin.usermgr.write` authorization, you need to check for the strings: `solaris.admin.usermgr.write`, `solaris.admin.usermgr.*`, `solaris.admin.*`, and `solaris.*`.

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

CHAPTER 7

Role-Based Access Control (Reference)

This chapter provides additional information that supplements Chapter 5.

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

For information on RBAC tasks, see Chapter 6.

RBAC Elements: Reference Information

This section describes the role-based access control (RBAC) elements in detail.

Configuring Recommended Roles

No predefined roles are shipped with the Solaris 9 software. Management at a customer site must decide what types of roles should be set up. However, three recommended roles can be readily configured by assigning the appropriate predefined rights profile to the corresponding roles:

- **Primary Administrator rights profile –** For creating a role that can perform all administrative tasks, granting rights to others, and editing rights that are associated with administrative roles. A user in this role can assign the Primary Administrator role and the ability to grant rights to other users.
- **System Administrator rights profile –** For creating a role that can perform most nonsecurity administrative tasks. For example, the System Administrator can add new user accounts, but cannot set passwords or grant rights to other users.
- **Operator rights profile –** For creating a role that can perform simple administrative tasks, such as backup and restore, and printer maintenance.

These rights profiles enable administrators to configure the suggested roles by using a single rights profile instead of having to mix and match rights profiles.

Those sites that customize roles should closely check the order of the rights profiles that are assigned to the role. The system does not prevent someone from typing multiple occurrences of the same command. The attributes that are assigned to the first occurrence of a command in a rights profile take precedence and all subsequent occurrences are ignored.

Note – You can also set up `root` as a role through a manual process. This method prevents users from logging in directly as `root`, forcing them to log in as themselves first. See "Making `Root` a Role" on page 101.

Contents of Rights Profiles

This section describes some typical rights profiles.

- The All rights profile provides role access to commands without security attributes.
- The Primary Administrator rights profile is designed specifically for the Primary Administrator role. The Primary Administrator rights profile allows the use of wildcards.
- The System Administrator rights profile is designed specifically for the System Administrator role. The System Administrator rights profile uses discrete supplementary profiles to create a powerful role.
- The Operator rights profile is designed specifically for the Operator role. The Operator rights profile uses a few discrete supplementary profiles to create a simple role.
- The Basic Solaris User rights profile shows how the `policy.conf` file can be used to assign tasks to users that are not related to security.
- The Printer Management rights profile exemplifies a profile that is dedicated to a single area of administration.

The tables in the following sections show the purpose and the contents of these rights profiles, including the commands, authorizations, supplementary rights, rights profiles, and associated help files.

Help files are in HTML and can be readily customized, if required. These files reside in the `/usr/lib/help/auths/locale/C` directory.

The Solaris Management Console Rights tool provides another way of inspecting the contents of the rights profiles.

All Rights Profile

The All rights profile uses the wildcard to include all commands, except for those commands without security attributes. This rights profile provides a role with access to all commands that are not explicitly assigned in other rights profiles. Without the All rights profile or some other rights profiles that use wildcards, a role has access to explicitly assigned commands only, which is not very practical.

Because commands in rights profiles are interpreted in the order in which they occur, any wildcard settings should be positioned last so that explicit attribute assignments are not inadvertently overridden. The All rights profile, if used, should be the final rights profile that is assigned.

TABLE 7–1 Contents of All Rights Profile

Purpose	Contents
To execute any command as the user or role	**Commands**: *
	Help File: `RtAll.html`

Primary Administrator Rights Profile

The Primary Administrator rights profile is assigned the most powerful role on the system, effectively providing that role with superuser capabilities.

- The `solaris.*` authorization effectively assigns all of the authorizations that are provided by the Solaris software.
- The `solaris.grant` authorization lets a role assign any authorization to any rights profile, role, or user.
- The command assignment `*:uid=0;gid=0` provides the ability to run any command with `UID=0` and `GID=0`.

The help file `RtPriAdmin.html` is identified so that a site can modify it if necessary. Help files are stored in the `/usr/lib/help/auths/locale/C` directory.

Note also that if the Primary Administrator rights profile is not consistent with a site's security policy, it can be modified or not assigned at all. However, the security capabilities in the Primary Administrator rights profile would need to be handled in one or more other rights profiles.

TABLE 7–2 Contents of Primary Administrator Rights Profile

Purpose	Contents
To perform all administrative tasks	**Commands**: *
	Authorizations: `solaris.*,solaris.grant`
	Help File: `RtPriAdmin.html`

System Administrator Rights Profile

The System Administrator rights profile is intended for the System Administrator role. Because the System Administrator does not have the broad powers of the Primary Administrator, no wildcards are used. Instead, discrete administrative rights profiles that do not deal with security are assigned. The commands that are assigned to the supplementary rights profiles are not shown in the following table.

Notice that the All rights profile is assigned at the end of the list of supplementary rights profiles.

TABLE 7–3 Contents of System Administrator Rights Profile

Purpose	Contents
To perform most nonsecurity administrative tasks	**Supplementary rights profiles**: Audit Review, Printer Management, Cron Management, Device Management, File System Management, Mail Management, Maintenance and Repair, Media Backup, Media Restore, Name Service Management, Network Management, Object Access Management, Process Management, Software Installation, User Management, All
	Help File: `RtSysAdmin.html`

Operator Rights Profile

The Operator rights profile is a less powerful administrative rights profile that provides the ability to do backups and printer maintenance. The ability to restore files has more security consequences, and the default is not to assign it to this rights profile.

TABLE 7–4 Contents of Operator Rights Profile

Purpose	Contents
To perform simple administrative tasks	**Supplementary rights profiles**: Printer Management, Media Backup, All **Help File**: `RtOperator.html`

Basic Solaris User Rights Profile for User

By default, the Basic Solaris User rights profile is assigned automatically to all users through the `policy.conf` file. This rights profile provides basic authorizations that are useful in normal operations. Note that the convenience that is offered by the Basic Solaris User rights profile must be balanced against site security requirements. Those sites that need stricter security might prefer to remove this rights profile from the `policy.conf` file.

TABLE 7–5 Contents of Basic Solaris User Rights Profile

Purpose	Contents
To automatically assign rights to all users	**Authorizations**: `solaris.profmgr.read`, `solaris.admin.usermgr.read`, `solaris.admin.logsvc.read`, `solaris.admin.fsmgr.read`, `solaris.admin.serialmgr.read`, `solaris.admin.diskmgr.read`, `solaris.admin.procmgr.user`, `solaris.compsys.read`, `solaris.admin.printer.read`, `solaris.admin.prodreg.read`, `solaris.admin.dcmgr.read` **Supplementary rights profiles**: All **Help File**: `RtDefault.html`

Printer Management Rights Profile

Printer Management is a typical rights profile that is intended for a specific task area. Both authorizations and commands are assigned to the Printer Management rights profile. The following table shows a partial list of commands.

TABLE 7–6 Contents of Printer Management Rights Profile

Purpose	Contents
To manage printers, daemons, and spooling	**Authorizations**: `solaris.admin.printer.delete`, `solaris.admin.printer.modify`, `solaris.admin.printer.read` **Commands**: `/usr/sbin/accept:euid=lp`, `/usr/ucb/lpq:euid=0`, `/etc/init.d/lp:euid=0`, `/usr/bin/lpstat:euid=0`, `/usr/lib/lp/lpsched:uid=0`, `/usr/sbin/lpfilter:euid=lp` **Help File**: `RtPrntMngmnt.html`

Authorizations

An *authorization* is a discrete right that can be granted to a role or user. Authorizations are checked by RBAC-compliant applications before a user gets access to the application or specific operations within it. This check replaces the tests in conventional UNIX applications for `UID=0`.

Authorization Naming Convention

An authorization has a name that is used internally and in files (for example, `solaris.admin.usermgr.pswd`), and a short description, which appears in the graphical user interfaces (for example, `Change Passwords`).

By convention, authorization names consist of the reverse order of the Internet name of the supplier, the subject area, any subareas, and the function, which are all separated by dots. An example would be `com.xyzcorp.device.access`. Exceptions to this convention are the authorizations from Sun Microsystems, Inc., which use the prefix `solaris` instead of an Internet name. Sun's convention enables administrators to apply authorizations in a hierarchical fashion by using a wildcard (*) to represent any strings to the right of a dot.

Example of Authorization Granularity

As an example of how authorizations are used, consider the following. A user in the Operator role might be limited to the `solaris.admin.usermgr.read` authorization, which provides read but not write access to users' configuration files. The System Administrator role naturally has the `solaris.admin.usermgr.read` and also the `solaris.admin.usermgr.write` authorization for making changes to users' files. However, without the `solaris.admin.usermgr.pswd` authorization, the System Administrator cannot change passwords. The Primary Administrator has all three of these authorizations.

The `solaris.admin.usermgr.pswd` authorization is required to make password changes in the Solaris Management Console User Tool. This authorization is also required for using the password modification options in the `smuser`, `smmultiuser`, and `smrole` commands.

Delegating Authorizations

An authorization that ends with the suffix `grant` permits a user or role to delegate to other users any assigned authorizations that begin with the same prefix.

For example, a role with the authorizations `solaris.admin.usermgr.grant` and `solaris.admin.usermgr.read` can delegate the `solaris.admin.usermgr.read` authorization to another user. A role with the `solaris.admin.usermgr.grant` and `solaris.admin.usermgr.*` can delegate any of the authorizations with the `solaris.admin.usermgr` prefix to other users.

Databases That Support RBAC

The following four databases store the data for the RBAC elements:

- `user_attr` (extended user attributes database) – Associates users and roles with authorizations and rights
- `auth_attr` (authorization attributes database) – Defines authorizations and their attributes, and identifies the associated help file
- `prof_attr` (rights profile attributes database) – Defines rights profiles, lists the rights profile's assigned authorizations, and identifies the associated help file
- `exec_attr` (execution attributes database) – Identifies the commands with security attributes that are assigned to specific rights profiles

---

Note – The commands can also indicate a security policy. Currently, the only security policy that is available for the Solaris operating environment is `suser` (short for superuser). The `suser` policy is the default and it accommodates both the ID attributes and authorizations. The Trusted Solaris environment, which can interoperate with the Solaris environment, uses a policy called `tsol`. Additional policies might be available in future releases.

The `policy.conf` database is also important to the RBAC implementation. This database can contain authorizations and rights profiles that are to be applied by default to all users.

RBAC Database Relationships

The following figure illustrates how the RBAC databases work together.

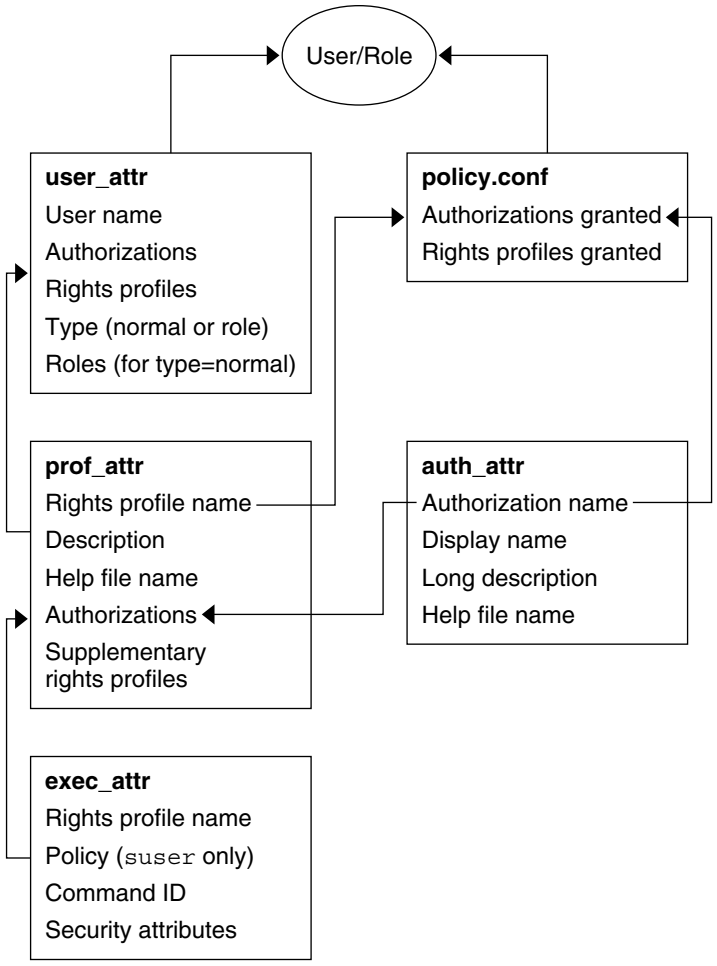


FIGURE 7–1 RBAC Database Relations

The `user_attr` database stores the basic definitions for both users and roles, which are differentiated by the type field. The `user_attr` database contains the attributes that are shown in the figure, which includes a comma-separated list of rights profile names. The definitions of the rights profiles are split between two databases. The `prof_attr` database contains rights profile identification information, authorizations that are assigned to the profile, and supplementary profiles. The `exec_attr` database identifies the security policy and contains commands with their associated security

attributes. The auth_attr database supplies authorization information to the Sun Management Console tools. The policy.conf database supplies default authorizations and rights profiles that are to be applied to all users.

Each database uses a key=value syntax for storing attributes. This method accommodates future expansion of the databases and enables a system to continue if it encounters a key that is unknown to its policy.

The scope of the RBAC databases can apply to individual hosts or to all hosts that are served by a name service such as NIS, NIS+, or LDAP. The precedence of local configuration files versus distributed databases for the user_attr database is set by the precedence that is specified for the passwd entry in the file /etc/nsswitch.conf. The precedence for the prof_attr and auth_attr databases are individually set in /etc/nsswitch.conf. The exec_attr database uses the same precedence as prof_attr. For example, if a command with security attributes is assigned to a profile that exists in two scopes, only the entry in the first scope is used.

The databases can reside on a local system or can be administered by the NIS, NIS+, or LDAP name service.

You can edit the databases manually or manipulate them with the commands that are described in "Command-Line Applications for Managing RBAC" on page 130.

The user_attr Database

The user_attr database contains user and role information that supplements the passwd and shadow databases. The user_attr database contains extended user attributes such as authorizations, rights profiles, and assigned roles. The fields in the user_attr database are separated by colons, as follows:

```
user:qualifier:res1:res2:attr
```

The following table describes these fields.

Field Name	Description
user	The name of the user or role as specified in the passwd database.
qualifier	Reserved for future use.
res1	Reserved for future use.
res2	Reserved for future use.

Field Name	Description
attr	An optional list of semicolon-separated (;) key-value pairs that describes the security attributes to be applied when the user runs commands. The four valid keys are type, auths, profiles, and roles. ■ The type key can be set to normal, if this account is for a normal user, or to role, if this account is for a role. ■ The auths key specifies a comma-separated list of authorization names that are chosen from names that are defined in the auth_attr database. Authorization names can include the asterisk (*) character as a wildcard. For example, solaris.device.* means all of the Solaris device authorizations. ■ The profiles key specifies an ordered, comma-separated list of rights profile names from the prof_attr database. The order of rights profiles works similarly to UNIX search paths. The first rights profile in the list that contains the command to be executed defines which (if any) attributes are to be applied to the command. ■ The roles key can be assigned to the user through a comma-separated list of role names. Note that roles are defined in the same user_attr database. Roles are indicated by setting the type value to role. Roles cannot be assigned to other roles.

The following example demonstrates how the Operator role is defined in a typical user_attr database and how it is assigned to user johnDoe. Roles and users are differentiated by the type keyword.

```
% grep operator /etc/user_attr
johnDoe::::type=normal;roles=sysadmin,operator
operator::::profiles=Operator;type=role
```

The auth_attr Database

All authorizations are stored in the auth_attr database. Authorizations can be assigned directly to users (or roles) in the user_attr database. Authorizations can also be assigned to rights profiles, which are assigned to users.

The fields in the auth_attr database are separated by colons, as follows:

```
authname:res1:res2:short_desc:long_desc:attr
```

The following table describes these fields.

Field Name	Description
authname	A unique character string that is used to identify the authorization in the format *prefix*.[*suffix*]. Authorizations for the Solaris operating environment use solaris as a prefix. All other authorizations should use a prefix that begins with the reverse-order Internet domain name of the organization that creates the authorization (for example, com.xyzcompany). The suffix indicates what is being authorized, which is typically the functional area and operation. When the authname consists of a prefix and functional area and ends with a period, the authname serves as a heading to be used by applications in their GUIs, rather than as an actual authorization. The authname of solaris.printmgr. is an example of a heading. When authname ends with the word "grant," the authname serves as a grant authorization and lets the user delegate authorizations with the same prefix and functional area to other users. The authname of solaris.printmgr.grant is an example of a grant authorization. solaris.printmgr.grant gives the user the right to delegate such authorizations as solaris.printmgr.admin and solaris.printmgr.nobanner to other users.
res1	Reserved for future use.
res2	Reserved for future use.
short_desc	A terse name for the authorization that is suitable for display in user interfaces, such as in a scrolling list in a GUI.
long_desc	A long description. This field identifies the purpose of the authorization, the applications in which it is used, and the type of user who might be interested in using it. The long description can be displayed in the help text of an application.
attr	An optional list of semicolon-separated (;) key-value pairs that describe the attributes of an authorization. Zero or more keys can be specified. The keyword help identifies a help file in HTML. Help files can be accessed from the index.html file in the /usr/lib/help/auths/locale/C directory.

The following example shows an auth_attr database with some typical values.

```
% grep printer /etc/security/auth_attr
solaris.admin.printer.:::Printer Information::help=AuthPrinterHeader.html
solaris.admin.printer.delete:::Delete Printer Information::help=AuthPrinterDelete.html
solaris.admin.printer.modify:::Update Printer Information::help=AuthPrinterModify.html
solaris.admin.printer.read:::View Printer Information::help=AuthPrinterRead.html
```

Note that solaris.admin.printer. is defined to be a heading, because it ends in a dot (.). Headings are used by the GUIs to organize families of authorizations.

The prof_attr Database

The prof_attr database stores the name, description, help file location, and authorizations that are assigned to rights profiles. The commands and security attributes that are assigned to rights profiles are stored in the exec_attr database (see "The exec_attr Database" on page 129). The fields in the prof_attr database are separated by colons:

```
profname:res1:res2:desc:attr
```

The following table describes these fields.

Field Name	Description
profname	The name of the rights profile. Rights profile names are case-sensitive. This name is also used by the user_attr database to indicate rights profiles that are assigned to roles and users.
res1	Reserved for future use.
res2	Reserved for future use.
desc	A long description. This field should explain the purpose of the rights profile, including what type of user would be interested in using it. The long description should be suitable for display in the help text of an application.
attr	An optional list of key-value pairs that are separated by semicolons (;) that describes the security attributes to apply to the object on execution. Zero or more keys can be specified. The two valid keys are help and auths. The keyword help identifies a help file in HTML. Help files can be accessed from the index.html file in the /usr/lib/help/auths/locale/C directory. The keyword auths specifies a comma-separated list of authorization names that are chosen from those names that are defined in the auth_attr database. Authorization names can be specified with the asterisk (*) character as a wildcard.

The following example shows a typical prof_attr database. Note that the Printer Management rights profile is a supplementary rights profile that is assigned to the Operator rights profile.

```
% grep 'Printer Management' /etc/security/prof_attr
Printer Management:::Manage printers, daemons, spooling:help=RtPrntAdmin.html; \
auths=solaris.admin.printer.read,solaris.admin.printer.modify,solaris.admin.printer.delete \
Operator:::Can perform simple administrative tasks:profiles=Printer Management,\
Media Backup,All;help=RtOperator.html
...
```

The exec_attr Database

An execution attribute is a command that is associated with a specific UID or GID and that is assigned to a rights profile. The command with its security attributes can be run by users or roles to whom the rights profile is assigned.

The exec_attr database stores the definitions of the execution attributes.

The fields in the exec_attr database are separated by colons:

```
name:policy:type:res1:res2:id:attr
```

The following table describes these fields.

Field Name	Description
name	The name of the rights profile. Rights profile names are case-sensitive. The name refers to a rights profile in the prof_attr database.
policy	The security policy that is associated with this entry. Currently, suser (the superuser policy model) is the only valid entry.
type	The type of entity that is specified. Currently, the only valid entity type is cmd (command).
res1	Reserved for future use.
res2	Reserved for future use.
id	A string that identifies the entity. Commands should have the full path or a path with a wildcard. To specify arguments, write a script with the arguments and point the id to the script.
attr	An optional list of semicolon (;) separated key-value pairs that describes the security attributes to apply to the entity on execution. Zero or more keys can be specified. The list of valid keywords depends on the policy that is enforced. The four valid keys are euid, uid, egid, and gid. The euid and uid keywords contain a single user name or a numeric user ID (UID). Commands that are designated with euid run with the effective UID indicated, which is similar to setting the setuid bit on an executable file. Commands that are designated with uid run with both the real and effective UIDs. The egid and gid keywords contain a single group name or numeric group ID (GID). Commands that are designated with egid run with the effective GID indicated, which is similar to setting the setgid bit on an executable file. Commands that are designated with gid run with both the real and effective GIDs.

The following example shows some typical values from an exec_attr database.

```
% grep 'Printer Management' /etc/security/exec_attr
Printer Management:suser:cmd:::/usr/sbin/accept:euid=lp
Printer Management:suser:cmd:::/usr/ucb/lpq:euid=0
Printer Management:suser:cmd:::/etc/init.d/lp:euid=0
```

.
.
.

The `policy.conf` File

The `policy.conf` file provides a way of granting specific rights profiles and authorizations to all users. The two types of entries in the file consist of key-value pairs. They are the following:

- `AUTHS_GRANTED=`*authorizations* – Refers to one or more authorizations
- `PROFS_GRANTED=`*right profiles* – Refers to one or more rights profiles

The following example shows some typical values from a `policy.conf` database.

```
# grep AUTHS /etc/security/policy
AUTHS_GRANTED=solaris.device.cdrw

# grep PROFS /etc/security/policy
PROFS_GRANTED=Basic Solaris User
```

RBAC Commands

This section lists commands that are used to administer RBAC. Also provided is a table of commands whose access can be controlled by authorizations.

Command-Line Applications for Managing RBAC

In addition to editing the RBAC databases directly, the following commands are available for managing access to tasks with RBAC.

TABLE 7–7 RBAC Administration Commands

Command	Description
`auths`(1)	Displays authorizations for a user.
`makedbm`(1M)	Makes a `dbm` file.
`nscd`(1M)	Name service cache daemon, useful for caching the `user_attr`, `prof_attr`, and `exec_attr` databases.

TABLE 7–7 RBAC Administration Commands *(Continued)*

Command	Description
pam_roles(5)	Role account management module for PAM. Checks for the authorization to assume role.
pfexec(1)	Used by profile shells to execute commands with attributes that are specified in the exec_attr database.
policy.conf(4)	Configuration file for security policy. Lists granted authorizations.
profiles(1)	Displays rights profiles for a specified user.
roles(1)	Displays roles that are granted to a user.
roleadd(1M)	Adds a role to the system.
roledel(1M)	Deletes a role from the system.
rolemod(1M)	Modifies a role's properties on the system.
smattrpop(1M)	Merges the source security attribute database into the target database. For use in situations where local databases need to be merged into a name service and in upgrades where conversion scripts are not supplied.
smexec(1M)	Manages entries in the exec_attr database. Requires authentication.
smmultiuser(1M)	Manages bulk operations on user accounts. Requires authentication.
smuser(1M)	Manages user entries. Requires authentication.
smprofile(1M)	Manages rights profiles in the prof_attr and exec_attr databases. Requires authentication.
smrole(1M)	Manages roles and users in role accounts. Requires authentication.
useradd(1M)	Adds a user account to the system. The -P option assigns a role to a user's account.
userdel(1M)	Deletes a user's login from the system.
usermod(1M)	Modifies a user's account properties on the system.

Commands That Require Authorizations

The following table provides examples of how authorizations are used to limit command options in the Solaris environment. See also "Authorizations" on page 122.

TABLE 7–8 Commands and Associated Authorizations

Commands	Authorization Requirements
at(1)	solaris.jobs.user required for all options (when neither at.allow nor at.deny files exist)

TABLE 7–8 Commands and Associated Authorizations *(Continued)*

Commands	Authorization Requirements
`atq`(1)	`solaris.jobs.admin` required for all options
`crontab`(1)	`solaris.jobs.user` required for the option to submit a job (when neither `crontab.allow` nor `crontab.deny` files exist)
	`solaris.jobs.admin` required for the options to list or modify other users' `crontab` files
`allocate`(1) (with BSM enabled only)	`solaris.device.allocate` (or other authorization as specified in `device_allocate`(4)) required to allocate a device.
	`solaris.device.revoke` (or other authorization as specified in `device_allocate` file) required to allocate a device to another user (`-F` option)
`deallocate`(1) (with BSM enabled only)	`solaris.device.allocate` (or other authorization as specified in `device_allocate`(4)) required to deallocate another user's device.
	`solaris.device.revoke` (or other authorization as specified in `device_allocate`) required to force deallocation of the specified device (`-F` option) or all devices (`-I` option)
`list_devices`(1) (with BSM enabled only)	`solaris.device.revoke` required to list another user's devices (`-U` option)

CHAPTER 8

Using the Automated Security Enhancement Tool (Tasks)

This chapter describes how to use the Automated Security Enhancement Tool (ASET) to monitor or restrict access to system files and directories.

The following is a list of the step-by-step instructions in this chapter.

Automated Security Enhancement Tool (ASET)

The Solaris 9 release includes the Automated Security Enhancement Tool (ASET). ASET helps you monitor and control system security by automatically performing tasks that you would otherwise do manually.

The ASET security package provides automated administration tools that enable you to control and monitor your system's security. You specify a security level—low, medium, or high—at which ASET will run. At each higher level, ASET's file-control functions increase to reduce file access and tighten your system security.

There are seven tasks involved with ASET, each task performs specific checks and adjustments to system files. The ASET tasks tighten file permissions, check the contents of critical system files for security weaknesses, and monitor crucial areas. ASET can also safeguard a network by applying the basic requirements of a firewall system to a system that serves as a gateway system. (See "Firewall Setup" on page 137.)

ASET uses master files for configuration. Master files, reports, and other ASET files are in the `/usr/aset` directory. These files can be changed to suit the particular requirements of your site.

Each task generates a report that notes detected security weaknesses and any changes the task has made to the system files. When run at the highest security level, ASET will attempt to modify all system security weaknesses. If ASET cannot correct a potential security problem, it reports the existence of the problem.

You can initiate an ASET session by using the `/usr/aset` command interactively. Or, you can set up ASET to run periodically by putting an entry into the `crontab` file.

ASET tasks are disk-intensive and can interfere with regular activities. To minimize the impact on system performance, schedule ASET to run when system activity level is lowest, for example, once every 24 or 48 hours at midnight.

ASET Security Levels

ASET can be set to operate at one of three security levels: low, medium, or high. At each higher level, ASET's file-control functions increase to reduce file access and heighten system security. These functions range from monitoring system security without limiting users' file access, to increasingly tightening access permissions until the system is fully secured.

The following table outlines these three levels of security.

Security Level	Description
Low	Ensures that attributes of system files are set to standard release values. ASET performs several checks and reports potential security weaknesses. At this level, ASET takes no action and does not affect system services.
Medium	Provides adequate security control for most environments. ASET modifies some settings of system files and parameters, restricting system access to reduce the risks from security attacks. ASET reports security weaknesses and any modifications it makes to restrict access. At this level, ASET does not affect system services.
High	Renders a highly secure system. ASET adjusts many system files and parameter settings to minimum access permissions. Most system applications and commands continue to function normally. However, at this level, security considerations take precedence over other system behavior.

Note – ASET does not change the permissions of a file to make it less secure, unless you downgrade the security level or intentionally revert the system to the settings that existed prior to running ASET.

ASET Tasks

This section discusses what ASET does. You should understand each ASET task (what its objectives are, what operations it performs, and what system components it affects) to interpret and use the reports effectively.

ASET report files contain messages that describe as specifically as possible any problems that were discovered by each ASET task. These messages can help you diagnose and correct these problems. However, successful use of ASET assumes that you possess a general understanding of system administration and system components. If you are a novice administrator, you can refer to other Solaris system administration documentation and related manual pages to prepare yourself for ASET administration.

The `taskstat` utility identifies the tasks that have been completed and the tasks that are still running. Each completed task produces a report file. For a complete description of the `taskstat` utility, refer to `taskstat`(1M).

System Files Permissions Tuning

This task sets the permissions on system files to the security level you designate. This task is run when the system is installed. If you decide later to alter the previously established levels, run this task again. At low security, the permissions are set to values that are appropriate for an open information-sharing environment. At medium security, the permissions are tightened to produce adequate security for most environments. At high security, they are tightened to severely restrict access.

Any modifications that this task makes to system files permissions or parameter settings are reported in the `tune.rpt` file. For example of the files that ASET consults when it sets permissions, see "Tune Files" on page 149.

System Files Checks

This task examines system files and compares each file with a description of that file as it is listed in a master file. The master file is created the first time ASET runs this task. The master file contains the system file settings that are enforced by `checklist` for the specified security level.

A list of directories whose files are to be checked is defined for each security level. You can use the default list, or you can modify it, specifying different directories for each level.

For each file, the following criteria are checked:

- Owner and group
- Permission bits
- Size and checksum
- Number of links
- Last modification time

Any discrepancies found are reported in the `cklist.rpt` file. This file contains the results of comparing system file size, permission, and checksum values to the master file.

User and Group Checks

This task checks the consistency and integrity of user accounts and groups as they are defined in the `passwd` and `group` files. This task checks the local, and NIS or NIS+ password files. NIS+ password file problems are reported but not corrected. This task checks for the following violations:

- Duplicate names or IDs
- Entries in incorrect format
- Accounts without a password
- Invalid login directories
- The `nobody` account
- Null group password
- A plus sign (+) in the `/etc/passwd` file on an NIS (or NIS+) server

Discrepancies are reported in the `usrgrp.rpt` file.

System Configuration Files Check

During this task, ASET checks various system tables, most of which are in the `/etc` directory. These files are the following:

- `/etc/default/login`
- `/etc/hosts.equiv`
- `/etc/inetd.conf`
- `/etc/aliases`
- `/var/adm/utmpx`
- `/.rhosts`
- `/etc/vfstab`
- `/etc/dfs/dfstab`
- `/etc/ftpd/ftpusers`

ASET performs various checks and modifications on these files, and reports all problems in the `sysconf.rpt` file.

Environment Variables Check

This task checks how the `PATH` and `UMASK` environment variables are set for root, and other users, in the `/.profile`, `/.login`, and `/.cshrc` files.

The results of checking the environment for security are reported in the `env.rpt` file.

`eeprom` Check

This task checks the value of the `eeprom` security parameter to ensure that it is set to the appropriate security level. You can set the `eeprom` security parameter to `none`, `command`, or `full`.

ASET does not change this setting, but reports its recommendations in the `eeprom.rpt` file.

Firewall Setup

This task ensures that the system can be safely used as a network relay. This task protects an internal network from external public networks by setting up a dedicated system as a firewall, which is described in "Firewall Systems" on page 44. The firewall system separates two networks. In this situation, each network approaches the other network as untrusted. The firewall setup task disables the forwarding of Internet Protocol (IP) packets and hides routing information from the external network.

The firewall task runs at all security levels, but takes action only at the highest level. If you want to run ASET at high security, but find that your system does not require firewall protection, you can eliminate the firewall task by editing the `asetenv` file.

Any changes that are made are reported in the `firewall.rpt` file.

ASET Execution Log

ASET generates an execution log whether it runs interactively or in the background. By default, ASET generates the log file on standard output. The execution log confirms that ASET ran at the designated time, and also contains any execution error messages. The `aset -n` command directs the log to be delivered by electronic mail to a designated user. For a complete list of ASET options, refer to the `aset`(1M) man page.

Example of an ASET Execution Log File

```
ASET running at security level low

Machine=example; Current time = 0325_08:00
```

```
aset: Using /usr/aset as working directory

Executing task list...
        firewall
        env
        sysconfig
        usrgrp
        tune
        cklist
        eeprom
All tasks executed. Some background tasks may still be running.

Run /usr/aset/util/taskstat to check their status:
     $/usr/aset/util/taskstat     aset_dir
Where aset_dir is ASET's operating directory, currently=/usr/aset

When the tasks complete, the reports can be found in:
     /usr/aset/reports/latest/*.rpt
You can view them by:
more /usr/aset/reports/latest/*.rpt
```

The execution log first shows the system and time that ASET was run. Then, the execution log lists each task as it is started.

ASET invokes a background process for each of these tasks, which are described in "ASET Tasks" on page 135. The task is listed in the execution log when it starts. This listing does not indicate that it has been completed. To check the status of the background tasks, use the `taskstat` command.

ASET Reports

All report files that are generated from ASET tasks are stored in subdirectories under the `/usr/aset/reports` directory. This section describes the structure of the `/usr/aset/reports` directory, and provides guidelines on managing the report files.

ASET places the report files in subdirectories that are named to reflect the time and date when the reports are generated. This convention enables you to keep an orderly trail of records that document the system status as it varies between ASET executions. You can monitor and compare these reports to determine the soundness of your system's security.

The following figure shows an example of the `reports` directory structure.

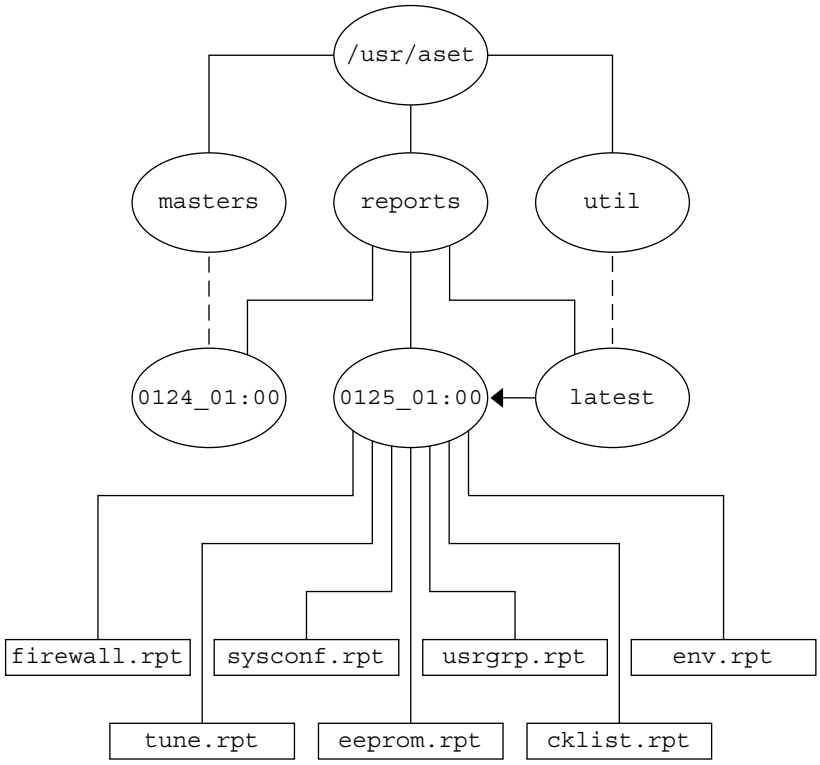


FIGURE 8–1 Structure of the ASET `reports` Directory

This example shows two report subdirectories.

- `0124_01:00`
- `0125_01:00`

The subdirectory names indicate the date and time that the reports were generated. Each report subdirectory name has the following format:

monthdate_hour:minute

where *month*, *date*, *hour,* and *minute* are all two-digit numbers. For example, `0125_01:00` represents January 25, at 1 a.m.

Each of the two report subdirectories contains a collection of reports that are generated from one execution of ASET.

The `latest` directory is a symbolic link that always points to the subdirectory that contains the latest reports. Therefore, to look at the latest reports that ASET has generated, you can go to the `/usr/aset/reports/latest` directory. There is a report file in this directory for each task that ASET performed during its most recent execution.

Format of ASET Report Files

Each report file is named after the task that generates it. See the following table for a list of tasks and their reports.

TABLE 8–1 ASET Tasks and Resulting Reports

Tasks	Report
System files permissions tuning (`tune`)	`tune.rpt`
System files checks (`cklist`)	`cklist.rpt`
User and group checks (`usrgrp`)	`usrgrp.rpt`
System configuration files check (`sysconf`)	`sysconf.rpt`
Environment variables check (`env`)	`env.rpt`
`eeprom` check (`eeprom`)	`eeprom.rpt`
Firewall setup (`firewall`)	`firewall.rpt`

Within each report file, messages are bracketed by a beginning and an ending banner line. Sometimes, a task terminates prematurely; for example, when a component of ASET is accidentally removed or damaged. In such cases, the report file usually contains a message near the end that indicates the reason for the premature termination.

The following is a sample report file, `usrgrp.rpt`.

```
*** Begin User and Group Checking ***

Checking /etc/passwd ...
Warning! Password file, line 10, no passwd
:sync::1:1::/:/bin/sync
..end user check; starting group check ...
Checking /etc/group...
*** End User And group Checking ***
```

Examining ASET Report Files

After you initially run or reconfigure ASET, you should examine the report files closely. Reconfiguration includes modifying the `asetenv` file or the master files in the `masters` subdirectory, or changing the security level at which ASET operates.

The reports record any errors that were introduced when you reconfigured ASET. By watching the reports closely, you can react to, and solve, problems as they arise.

Comparing ASET Report Files

After you monitor the report files for a period during which there are no configuration changes or system updates, you might find that the content of the reports begin to stabilize and that it contains little, if any, unexpected information. You can use the `diff` utility to compare reports.

ASET Master Files

ASET's master files, `tune.high, tune.low`, `tune.med`, and `uid_aliases`, are located in the `/usr/aset/masters` directory. ASET uses the master files to define security levels.

Tune Files

The `tune.low`, `tune.med`, and `tune.high` master files define the available ASET security levels. They specify the attributes of system files at each level and are used for comparison and reference purposes.

The `uid_aliases` File

The `uid_aliases` file contains a list of multiple user accounts that share the same user ID (UID). Normally, ASET warns about such multiple user accounts because this practice lessens accountability. You can allow for exceptions to this rule by listing the exceptions in the `uid_aliases` file. ASET does not report entries in the `passwd` file with duplicate UIDs if these entries are specified in the `uid_aliases` file.

Avoid having multiple user accounts (password entries) share the same UID. You should consider other methods of achieving your objective. For example, if you intend for several users to share a set of permissions, you could create a group account. The sharing of UIDs should be your last resort, used only when absolutely necessary and when other methods will not accomplish your objectives.

You can use the `UID_ALIASES` environment variable to specify an alternate aliases file. The default file is `/usr/aset/masters/uid_aliases`.

The Checklist Files

The master files that are used by the systems files checks are generated when you first execute ASET, or when you run ASET after you change the security level.

The following environment variables define the files that are checked by this task:

- CKLISTPATH_LOW
- CKLISTPATH_MED
- CKLISTPATH_HIGH

ASET Environment File (asetenv)

The environment file, asetenv, contains a list of environment variables that affect ASET tasks. Some of these variables can be changed to modify ASET operation.

Configuring ASET

This section discusses how ASET is configured and the environment under which it operates.

ASET requires minimum administration and configuration, and in most cases, you can run it with the default values. You can, however, fine-tune some of the parameters that affect the operation and behavior of ASET to maximize its benefit. Before you change the default values, you should understand how ASET works, and how it affects the components of your system.

ASET relies on four configuration files to control the behavior of its tasks:

- /usr/aset/asetenv
- /usr/aset/masters/tune.low
- /usr/aset/masters/tune.med
- /usr/aset/masters/tune.high

Modifying the Environment File (asetenv)

The /usr/aset/asetenv file has two main sections:

- A user-configurable environment variables section
- An internal environment variables section

You can alter the user-configurable parameters section. However, the settings in the internal environment variables section are for internal use only and should not be modified.

You can edit the entries in the user-configurable section to do the following:

- Choose which tasks to run
- Specify the directories for the system files checks task
- Schedule ASET execution
- Specify a UID aliases file
- Extend checks to NIS+ tables

Choose Which Tasks to Run: TASKS

Each task that ASET performs monitors a particular area of system security. In most system environments, all the tasks are necessary to provide balanced security coverage. However, you might decide to eliminate one or more tasks.

For example, the firewall task runs at all security levels, but takes action only at the high security level. You might want to run ASET at the high security level, but you do not require firewall protection.

You can set up ASET to run at the high security level without the firewall feature by editing the TASKS list of environment variables in the asetenv file. By default, the TASKS list contains all of the ASET tasks. To delete a task, remove the task-related environment variable from the file. In this case, you would delete the firewall environment variable from the list. The next time ASET runs, the excluded task will not be performed.

The following example shows the TASKS list with all of the ASET tasks.

```
TASKS="env sysconfig usrgrp tune cklist eeprom firewall"
```

Specify Directories for System Files Checks Task: CKLISTPATH

The system files check checks the attributes of files in selected system directories. You define which directories to check by using these checklist path environment variables.

The CKLISTPATH_LOW variable defines the directories to be checked at the low security level. CKLISTPATH_MED and CKLISTPATH_HIGH environment variables function similarly for the medium and high security levels.

The directory list that is defined by an environment variable at a lower security level should be a subset of the directory list that is defined at the next higher level. For example, all directories that are specified for CKLISTPATH_LOW should be included in CKLISTPATH_MED, and all the directories that are specified for CKLISTPATH_MED should be included in CKLISTPATH_HIGH.

Checks performed on these directories are not recursive. ASET only checks those directories that are explicitly listed in the environment variable. ASET does not check their subdirectories.

You can edit these environment variable definitions to add or delete directories that you want ASET to check. Note that these checklists are useful only for system files that do not normally change from day to day. A user's home directory, for example, is generally too dynamic to be a candidate for a checklist.

Schedule ASET Execution: PERIODIC_SCHEDULE

When you start ASET, you can start it interactively, or use the -p option to request that the ASET tasks run at a scheduled time. You can run ASET periodically, at a time when system demand is light. For example, ASET consults PERIODIC_SCHEDULE to determine how frequently to execute the ASET tasks, and at what time to run them. For detailed instructions about setting up ASET to run periodically, see "How to Run ASET Periodically" on page 152.

The format of PERIODIC_SCHEDULE follows the format of crontab entries. See crontab(1) for complete information.

Specify an Aliases File: UID_ALIASES

The UID_ALIASES variable specifies an aliases file that lists shared UIDs. The default file is /usr/aset/masters/uid_aliases.

Extend Checks to NIS+ Tables: YPCHECK

The YPCHECK environment variable specifies whether ASET should also check system configuration file tables. YPCHECK is a Boolean variable; you can specify only true or false for it. The default value is false, which disables NIS+ table checking.

To understand how this environment variable works, consider its effect on the passwd file. When set to false, ASET checks the local passwd file. When set to true, the task also checks the NIS+ passwd file for the domain of the system.

Note – Although ASET automatically repairs the local tables, it only reports potential problems in the NIS+ tables. ASET does not change them.

Modifying the Tune Files

ASET uses the three master tune files, tune.low, tune.med, and tune.high, to ease or tighten access to critical system files. These master files are located in the /usr/aset/masters directory, and they can be modified to suit your environment. For additional information, see "Tune Files" on page 149.

The tune.low file sets permissions to values that are appropriate for default system settings. The tune.med file further restricts these permissions and includes entries that are not present in tune.low. The tune.high file restricts permissions even further.

Note – Modify settings in the tune files by adding or deleting file entries. Setting a permission to a less restrictive value than the current setting has no effect. The ASET tasks do not relax permissions unless you downgrade your system security to a lower level.

Restoring System Files Modified by ASET

When ASET is executed for the first time, it saves and archives the original system files. The `aset.restore` utility reinstates these files. This utility also deschedules ASET, if it is currently scheduled for periodic execution. The `aset.restore` command is located in `/usr/aset`, the ASET operating directory.

Changes that are made to system files are lost when you run the `aset.restore` command.

You should use the `aset.restore` command in the following instances:

- When you want to remove ASET changes and restore the original system. If you want to deactivate ASET permanently, you can remove it from `cron` scheduling if the `aset` command had previously been added to root's `crontab`. For instructions on how to use `cron` to remove automatic execution, see "How to Stop Running ASET Periodically" on page 153.
- After a brief period of experimenting with ASET, to restore the original system state.
- When some major system feature is not working properly, and you suspect that ASET is causing the problem.

Network Operation With the NFS System

Generally, ASET is used in standalone mode, even on a system that is part of a network. As system administrator for your standalone system, you are responsible for the security of your system and for running and managing ASET to protect your system.

You can also use ASET in the NFS distributed environment. As a network administrator, you are responsible for installing, running, and managing various administrative tasks for all your clients. To facilitate ASET management across several client systems, you can make configuration changes that are applied globally to all clients, which eliminates the need for you to log in to each system to repeat the process.

When you are deciding how to set up ASET on your networked systems, you should consider how much you want users to control security on their own systems, and how much you want to centralize responsibility for security control.

Providing a Global Configuration for Each Security Level

A situation might arise where you want to set up more than one network configuration. For example, you might want to set up one configuration for clients that are designated with low security level, another configuration for those clients with medium level, and yet another configuration with high level.

If you need to create a separate ASET network configuration for each security level, you can create three ASET configurations on the server, one configuration for each level. You would export each configuration to the clients with the appropriate security level. Some ASET components that are common to all three configurations could be shared by using links.

Collecting ASET Reports

Not only can you centralize the ASET components on a server to be accessed by clients with or without superuser privilege, but you can also set up a central directory on a server to collect all reports that are produced by tasks that run on various clients. For instructions on setting up a collection mechanism, see "How to Collect ASET Reports on a Server" on page 153.

Setting up the collection of reports on a server allows you to review reports for all clients from one location. You can use this method whether or not a client has superuser privilege. Alternatively, you can leave the reports directory on the local system when you want users to monitor their own ASET reports.

ASET Environment Variables

The following table lists the ASET environment variables and the values that they specify.

TABLE 8–2 ASET Environment Variables and Their Meanings

Environment Variable	Value Specified
`ASETDIR`	ASET working directory
`ASETSECLEVEL`	Security level
`PERIODIC_SCHEDULE`	Periodic schedule
`TASKS`	Tasks to run
`UID_ALIASES`	Aliases file
`YPCHECK`	Whether to extend checks to NIS maps and NIS+ tables

TABLE 8–2 ASET Environment Variables and Their Meanings *(Continued)*

Environment Variable	Value Specified
CKLISTPATH_LOW	Directory lists for low security
CKLISTPATH_MED	Directory list for medium security
CKLISTPATH_HIGH	Directory list for high security

The environment variables that are listed in the following sections are found in the /usr/aset/asetenv file. The ASETDIR and ASETSECLEVEL variables are optional and can be set only through the shell by using the aset command. The other environment variables can be set by editing the file.

ASETDIR Environment Variable

ASETDIR specifies an ASET working directory.

From the C shell, type:

```
% setenv ASETDIR pathname
```

From the Bourne shell or the Korn shell, type:

```
$ ASETDIR=pathname
$ export ASETDIR
```

Set *pathname* to the full path name of the ASET working directory.

ASETSECLEVEL Environment Variable

The ASETSECLEVEL variable specifies a security level at which ASET tasks are executed.

From the C shell, type:

```
% setenv ASETSECLEVEL level
```

From the Bourne shell or the Korn shell, type:

```
$ ASETSECLEVEL=level
export ASETSECLEVEL
```

In these commands, *level* can be set to one of the following:

low	Low security level
med	Medium security level

`high` High security level

`PERIODIC_SCHEDULE` Environment Variable

The value of `PERIODIC_SCHEDULE` follows the same format as the `crontab` file. Specify the variable value as a string of five fields enclosed in double quotation marks, with each field separated by a space:

"*minutes hours day-of-month month day-of-week*"

TABLE 8–3 `Periodic_Schedule` Variable Values

Variable	Value
minutes hours	Specifies start time in number of minutes (0–59) after the hour and the hour (0–23)
day-of-month	Specifies the day of the month when ASET should be run, with values from 1–31
month	Specifies the month of the year when ASET should be run, with values from 1–12
day-of-week	Specifies the day of the week when ASET should be run, with values from 0–6; Sunday is day 0

The following rules apply:

- You can specify a list of values, each delimited by a comma, for any field.
- You can specify a value as a number, or you can specify it as a range; that is, a pair of numbers that are joined by a hyphen. A range states that the ASET tasks should be executed for every time that is included in the range.
- You can specify an asterisk (*) as the value of any field. An asterisk inclusively specifies all possible values of the field.

The default entry for the `PERIODIC_SCHEDULE` variable causes ASET to execute at 12:00 midnight every day:

```
PERIODIC_SCHEDULE=”0 0 * * *”
```

`TASKS` Environment Variable

The `TASKS` variable lists the tasks that ASET performs. The default is to list all seven tasks:

```
TASKS=”env sysconfig usrgrp tune cklist eeprom firewall”
```

UID_ALIASES Environment Variable

The UID_ALIASES variable specifies an aliases file. If present, ASET consults this file for a list of permitted multiple aliases. The format is UID_ALIASES=*pathname*, where *pathname* is the full path name of the aliases file.

The default is as follows:

```
UID_ALIASES=${ASETDIR}/masters/uid_aliases
```

YPCHECK Environment Variable

The YPCHECK variable extends the task of checking system tables to include NIS or NIS+ tables. This variable is a Boolean variable, which can be set to either true or false.

The default is false, which confines the checking to local system tables:

```
YPCHECK=false
```

CKLISTPATH_level Environment Variable

The three checklist path variables list the directories to be checked by the system files checks task. The following definitions of the variables are set by default. They illustrate the relationship between the variables at different levels:

```
CKLISTPATH_LOW=${ASETDIR}/tasks:${ASETDIR}/util:${ASETDIR}/masters:
/etc
CKLISTPATH_MED=${CKLISTPATH_LOW}:/usr/bin:/usr/ucb
CKLISTPATH_HIGH=${CKLISTPATH_MED}:/usr/lib:/sbin:/usr/sbin:/usr/ucblib
```

The values for the checklist path environment variables are similar to those values of the shell path variables, in that they are lists of directory names that are separated by colons. You use an equal sign (=) to connect the variable name to its value.

ASET File Examples

This section has examples of some ASET files, including the tune files and the aliases file.

Tune Files

ASET maintains three tune files. The following table describes the format of entries in all three tune files.

TABLE 8–4 Entry Format for Tune Files

Field Name	Description
pathname	The full path name of the file
mode	A five-digit number that represents the permission setting
owner	The owner of the file
group	The group owner of the file
type	The type of file

The following rules apply when you edit the tune files:

- You can use regular shell wildcard characters, such as an asterisk (`*`) and a question mark (`?`), in the path name for multiple references. See `sh`(1) for more information.
- *mode* represents the least restrictive value. If the current setting is already more restrictive than the specified value, ASET does not loosen the permission settings. For example, if the specified value is `00777`, the permission remains unchanged, because `00777` is always less restrictive than whatever the current setting is.

  This process is how ASET handles mode setting, unless the security level is being downgraded or you are removing ASET. When you decrease the security level from what it was for the previous execution, or when you want to restore the system files to the state they were in before ASET was first executed, ASET recognizes what you are doing and decreases the protection level.
- You must use names for *owner* and *group* instead of numeric IDs.
- You can use a question mark (`?`) in place of *owner*, *group*, and *type* to prevent ASET from changing the existing values of these parameters.
- *type* can be `symlink` (symbolic link), directory, or file (everything else).
- Higher security level tune files reset file permissions to be at least as restrictive as they are at lower levels. Also, at higher security levels, additional files are added to the list.
- A file can match more than one tune file entry. For example, `etc/passwd` matches the `etc/pass*` and `/etc/*` entries.
- Where two entries have different permissions, the file permission is set to the most restrictive value. In the following example, the permission of the `/etc/passwd` file will be set to `00755`, which is the more restrictive of `00755` and `00770`.

  ```
  /etc/pass* 00755 ? ? file
  /etc/* 00770 ? ? file
  ```
- If two entries have different *owner* or *group* designations, the last entry takes precedence. In the following example, the owner of `/usr/sbin/chroot` will be set to `root`.

```
/usr/sbin/chroot 00555 bin bin file
/usr/sbin/chroot 00555 root bin file
```

Aliases File

The aliases file contains a list of aliases that share the same user ID.

Each entry is in this form:

uid=alias1=alias2=alias3=. . .

uid	Shared UID.
aliasn	User account that share the UID.

For example, the following entry lists the UID `0` that is being shared by the `sysadm` and `root` accounts:

```
0=root=sysadm
```

Running ASET

This section describes how to run ASET either interactively or periodically.

▼ How to Run ASET Interactively

1. **Become superuser.**
2. **Run ASET interactively by using the `aset` command.**

   # **`/usr/aset/aset -l`** *level* **`-d`** *pathname*

level	Specifies the level of security. Valid values are `low`, `medium`, or `high`. The default setting is `low`. For detailed information about security levels see "ASET Security Levels" on page 134.
pathname	Specifies the working directory for ASET. The default is `/usr/aset`.

3. **Verify that ASET is running by viewing the ASET execution log that is displayed on the screen.**

The execution log message identifies which tasks are being run.

Example—Running ASET Interactively

The following example shows ASET being run at low security with the default working directory.

```
# /usr/aset/aset -l low
======= ASET Execution Log =======

ASET running at security level low

Machine = jupiter; Current time = 0111_09:26

aset: Using /usr/aset as working directory

Executing task list ...
    firewall
    env
    sysconf
    usrgrp
    tune
    cklist
    eeprom

All tasks executed. Some background tasks may still be running.

Run /usr/aset/util/taskstat to check their status:
 /usr/aset/util/taskstat [aset_dir]

where aset_dir is ASET's operating
directory,currently=/usr/aset.

When the tasks complete, the reports can be found in:
 /usr/aset/reports/latest/*.rpt

You can view them by:
 more /usr/aset/reports/latest/*.rpt
```

▼ How to Run ASET Periodically

1. **Become superuser.**
2. **If necessary, set up the time when you want ASET to run periodically.**

   You should have ASET run when system demand is light. The `PERIODIC_SCHEDULE` environment variable in the `/usr/aset/asetenv` file is used to set up the time for ASET to run periodically. By default, the time is set for every day at midnight.

   If you want to set up a different time, edit the `PERIODIC_SCHEDULE` variable in the `/usr/aset/asetenv` file. For detailed information about setting the PERIODIC_SCHEDULE variable see "`PERIODIC_SCHEDULE` Environment Variable"

on page 148.

3. **Add an entry to the `crontab` file by using the `aset` command.**

   ```
   # /usr/aset/aset -p
   ```

   The `-p` option inserts a line in the `crontab` file that starts ASET running at the time determined by the `PERIODIC_SCHEDULE` environment variable in the `/usr/aset/asetenv` file.

4. **Display the `crontab` entry to verify when ASET will run.**

   ```
   # crontab -l root
   ```

▼ How to Stop Running ASET Periodically

1. **Become superuser.**
2. **Edit the `crontab` file.**

   ```
   # crontab -e root
   ```

3. **Delete the ASET entry.**
4. **Save the changes and exit.**
5. **Display the `crontab` entry to verify that the ASET entry is deleted.**

   ```
   # crontab -l root
   ```

▼ How to Collect ASET Reports on a Server

1. **Become superuser.**
2. **Set up a directory on the server:**

   a. **Change to the `/usr/aset` directory.**

      ```
      mars# cd /usr/aset
      ```

   b. **Create a *rptdir* directory.**

      ```
      mars# mkdir rptdir
      ```

   c. **Change to the *rptdir* directory, and create a *client_rpt* directory.**

      This creates a subdirectory (*client_rpt*) for a client. Repeat this step for each client whose reports you need to collect.

      ```
      mars# cd rptdir
      mars# mkdir client_rpt
      ```

The following example shows the creation of the directory `all_reports`, and the subdirectories `pluto_rpt` and `neptune_rpt`.

```
mars# cd /usr/aset
mars# mkdir all_reports
mars# cd all_reports
mars# mkdir pluto_rpt
mars# mkdir neptune_rpt
```

3. **Add the *client_rpt* directories to the `/etc/dfs/dfstab` file.**

   The directories should have read and write options.

   For example, the following entries in the `dfstab` file are shared with read and write permissions.

   ```
   share -F nfs -o rw=pluto /usr/aset/all_reports/pluto_rpt
   share -F nfs -o rw=neptune /usr/aset/all_reports/neptune_rpt
   ```

4. **Make the resources in the `dfstab` file available to the clients.**

   ```
   # shareall
   ```

5. **On each client, mount the client subdirectory from the server at the mount point, `/usr/aset/masters/reports`.**

   ```
   # mount server:/usr/aset/client_rpt /usr/aset/masters/reports
   ```

6. **Edit the `/etc/vfstab` file to mount the directory automatically at boot time.**

   The following sample entry in `/etc/vfstab` on `neptune` lists the directory to be mounted from `mars`, `/usr/aset/all_reports/neptune_rpt`, and the mount point on `neptune`, `/usr/aset/reports`. At boot time, the directories that are listed in `vfstab` are automatically mounted.

   ```
   mars:/usr/aset/all_reports/neptune.rpt /usr/aset/reports nfs - yes hard
   ```

Troubleshooting ASET Problems

This section documents the error messages that are generated by ASET.

ASET Error Messages

`ASET failed: no mail program found.`

Cause: ASET is directed to send the execution log to a user, but no mail program can be found.

Solution: Install a mail program.

```
Usage: aset [-n user[@host]] in /bin/mail or /usr/ucb/mail.
Cannot decide current and previous security levels.
```

Cause: ASET cannot determine what the security levels are for the current and previous invocations.

Solution: Ensure the current security level is set either through the command-line option or the `ASETSECLEVEL` environment variable. Also, ensure that the last line of `ASETDIR/archives/asetseclevel.arch` correctly reflects the previous security level. If these values are not set or are incorrect, correct them.

```
ASET working directory undefined.
To specify, set ASETDIR environment variable or use command line
option -d.
ASET startup unsuccessful.
```

Cause: The ASET working (operating) directory is not defined, or it is defined incorrectly.

Solution: Use the `ASETDIR` environment variable or the `-d` command-line option to correct it, and restart ASET.

```
ASET working directory $ASETDIR missing.
ASET startup unsuccessful.
```

Cause: The ASET working (operating) directory is not defined, or it is defined incorrectly. This problem might be because the `ASETDIR` variable or the `-d` command-line option refers to a nonexistent directory.

Solution: Ensure that the correct directory, that is, the directory that contains the ASET directory hierarchy, is referred to correctly.

```
Cannot expand $ASETDIR to full pathname.
```

Cause: ASET cannot expand the directory name that is given by the `ASETDIR` variable or the `-d` command-line option to a full path name.

Solution: Ensure that the directory name is correct, and that it refers to an existing directory to which the user has access.

```
aset: invalid/undefined security level.
To specify, set ASETSECLEVEL environment variable or use command
line option -l, with argument= low/med/high.
```

Cause: The security level is not defined, or it is defined incorrectly. Only the values `low`, `med`, or `high` are acceptable.

Solution: Use the `ASETSECLEVEL` variable or the `-l` command-line option to specify one of the three values.

`ASET environment file asetenv not found in $ASETDIR.`

`ASET startup unsuccessful.`

Cause: ASET cannot locate an `asetenv` file in its working directory.

Solution: Ensure there is an `asetenv` file in ASET's working directory. See `asetenv`(4) for the details about this file.

`filename doesn't exist or is not readable.`

Cause: The file that is referred to by *filename* doesn't exist or is not readable. This problem can occur when you are using the `-u` option, where you can specify a file that contains a list of users whom you want to check.

Solution: Ensure that the argument to the `-u` option exists and is readable.

`ASET task list TASKLIST undefined.`

Cause: The ASET task list, which should be defined in the `asetenv` file, is not defined. This message can mean that your `asetenv` file is bad.

Solution: Examine your `asetenv` file. Ensure that the task list is defined in the `User Configurable` section. Also check other parts of the file to ensure that the file is intact. See `asetenv`(4) for the content of a valid `asetenv` file.

`ASET task list $TASKLIST missing.`

`ASET startup unsuccessful.`

Cause: The ASET task list, which should be defined in the `asetenv` file, is not defined. This message can mean that your `asetenv` file is bad.

Solution: Examine your `asetenv` file. Ensure that the task list is defined in the `User Configurable` section. Also check other parts of the file to ensure that the file is intact. See `asetenv`(4) for the content of a valid `asetenv` file.

`Schedule undefined for periodic invocation.`

`No tasks executed or scheduled. Check asetenv file.`

Cause: ASET scheduling is requested by using the `-p` option, but the environment variable `PERIODIC_SCHEDULE` is undefined in the `asetenv` file.

Solution: Check the `User Configurable` section of the `asetenv` file to ensure that the variable is defined and is in proper format.

`Warning! Duplicate ASET execution scheduled.`

`Check crontab file.`

Cause: ASET is scheduled to run more than once. In other words, ASET scheduling is requested while a schedule is already in effect. This message does not necessarily indicate an error if more than one schedule is indeed desired. In this instance, the messages servers only as a warning. If you want more than one schedule, you should use the proper scheduling format with the `crontab` command. For more information, see the `crontab`(1) man page.

Solution: Verify, through the `crontab` command, that the correct schedule is in effect. Ensure that no unnecessary `crontab` entries for ASET are in place.

PART III Authentication Services and Secure Communication

Chapter 9	Provides information about Diffie-Hellman authentication.
Chapter 10	Provides information about the Pluggable Authentication Module (PAM) framework.
Chapter 11	Provides an introduction to Solaris Secure Shell, as well as step-by-step instructions.
Chapter 12	Provides a description of the files that are used to configure Solaris Secure Shell.
Chapter 13	Provides overview information about Sun Enterprise Authentication Mechanism (SEAM).
Chapter 14	Provides a list of information that needs to be gathered and issues that need to be resolved before you configure SEAM.
Chapter 15	Provides step-by-step instructions for configuring SEAM.
Chapter 16	Provides a list of SEAM error messages, how to fix the conditions that generate the messages, and how to troubleshoot some error conditions.
Chapter 17	Provides step-by-step instructions for administering SEAM principals and policies with the `gkadmin` GUI and at the command line.
Chapter 18	Provides user instructions for SEAM.
Chapter 19	Provides reference information about SEAM.

CHAPTER 9

Using Authentication Services (Tasks)

This chapter provides information about the Diffie-Hellman authentication mechanism that can be used with Secure RPC.

The following is a list of the step-by-step instructions in this chapter.

Overview of Secure RPC

Secure RPC is an authentication method that authenticates both the host and the user who is making a request for a service. Secure RPC uses the Diffie-Hellman authentication mechanism. This authentication mechanism uses DES encryption. Applications that use Secure RPC include NFS and the NIS+ name service.

NFS Services and Secure RPC

NFS enables several hosts to share files over the network. Under the NFS service, a server holds the data and resources for several clients. The clients have access to the file systems that the server shares with the clients. Users who are logged in to the client machines can access the file systems by mounting the file systems from the server. To the user on the client machine, it appears as if the files are local to the client. One of the most common uses of NFS allows systems to be installed in offices, while keeping all user files in a central location. Some features of the NFS service, such as the `mount -nosuid` option, can be used to prohibit the opening of devices and file systems by unauthorized users.

The NFS service uses Secure RPC to authenticate users who make requests over the network. This process is known as Secure NFS. The authentication mechanism, `AUTH_DH`, uses DES encryption with Diffie-Hellman authentication to ensure authorized access. The `AUTH_DH` mechanism has also been called `AUTH_DES`.

- For how to set up and administer Secure NFS, see "Administering the Secure NFS System" in *System Administration Guide: Resource Management and Network Services*.
- For how to set up the NIS+ tables and enter names in the `cred` table, see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.
- For an outline of the steps that are involved in RPC authentication, see "Implementation of Diffie-Hellman Authentication" on page 163.

DES Encryption

The Data Encryption Standard (DES) encryption functions use a 56-bit key to encrypt data. If two credential users or principals know the same DES key, they can communicate in private by using the key to encipher and decipher text. DES is a relatively fast encryption mechanism. A DES chip makes the encryption even faster. However, if the chip is not present, a software implementation is substituted.

The risk of using just the DES key is that an intruder can collect enough cipher-text messages that were encrypted with the same key to be able to discover the key and decipher the messages. For this reason, security systems such as Secure NFS change the keys frequently.

Kerberos Authentication

Kerberos is an authentication system that was developed at MIT. Encryption in Kerberos is based on DES. Kerberos V4 support is no longer supplied as part of Secure RPC. However, a client-side implementation of Kerberos V5, which uses RPCSEC_GSS, is included with this release. For more information, see Chapter 13.

Diffie-Hellman Authentication

The Diffie-Hellman (DH) method of authenticating a user is nontrivial for an intruder to crack. The client and the server have their own private key, which they use with the public key to devise a common key. The private key is also known as the secret key. The client and the server use the common key to communicate with each other by using an agreed-on encryption function, such as DES. This method was identified as DES authentication in previous Solaris releases.

Authentication is based on the ability of the sending system to use the common key to encrypt the current time. Then the receiving system can decrypt and check against its current time. Make sure to synchronize the time on the client and the server.

The public keys and private keys are stored in an NIS or NIS+ database. NIS stores the keys in the `publickey` map. NIS+ stores the keys in the `cred` table. These files contain the public key and the private key for all potential users.

The system administrator is responsible for setting up NIS maps or NIS+ tables, and generating a public key and a private key for each user. The private key is stored in encrypted form with the user's password. This process makes the private key known only to the user.

Implementation of Diffie-Hellman Authentication

This section describes the series of transactions in a client-server session that use DH authentication (`AUTH_DH`).

Generating the Public Keys and Secret Keys

Sometime prior to a transaction, the administrator runs either the `newkey` or `nisaddcred` command to generate a public key and a secret key. Each user has a unique public key and secret key. The public key is stored in a public database. The secret key is stored in encrypted form in the same database. To change the key pair, use the `chkey` command.

Running the `keylogin` Command

Normally, the login password is identical to the secure RPC password. In this case, the `keylogin` command is not required. However, if the passwords are different, the users have to log in, and then run a `keylogin` command explicitly.

The `keylogin` command prompts the user for a secure RPC password. The command then uses the password to decrypt the secret key. The `keylogin` command then passes the decrypted secret key to a program that is called the keyserver. The keyserver is an RPC service with a local instance on every computer. The keyserver saves the decrypted secret key and waits for the user to initiate a secure RPC transaction with a server.

If both the login password and the RPC password are the same, the login process passes the secret key to the keyserver. If the passwords are required to be different and the user must always run the `keylogin` command, then the `keylogin` command can be included in the user's environment configuration file, such as the `~/.login`, `~/.cshrc`, or `~/.profile` file. Then the `keylogin` command runs automatically whenever the user logs in.

Generating the Conversation Key

When the user initiates a transaction with a server, the following occurs:

1. The keyserver randomly generates a conversation key.

2. The kernel uses the conversation key to encrypt the client's time stamp, among other things.
3. The keyserver looks up the server's public key in the public key database. See the `publickey`(4) man page for more information.
4. The keyserver uses the client's secret key and the server's public key to create a common key.
5. The keyserver encrypts the conversation key with the common key.

First Contact With the Server

The transmission, which includes the encrypted time stamp and the encrypted conversation key, is then sent to the server. The transmission includes a credential and a verifier. The credential contains three components:

- The client's net name
- The conversation key, which is encrypted with the common key
- A "window," which is encrypted with the conversation key

The window is the difference in time that the client says should be allowed between the server's clock and the client's time stamp. If the difference between the server's clock and the time stamp is greater than the window, the server rejects the client's request. Under normal circumstances, this rejection does not happen, because the client first synchronizes with the server before starting the RPC session.

The client's verifier contains the following:

- The encrypted time stamp
- An encrypted verifier of the specified window, which is decremented by 1

The window verifier is needed in case somebody wants to impersonate a user. The impersonator can write a program that, instead of filling in the encrypted fields of the credential and verifier, just stuffs in random bits. The server decrypts the conversation key into some random key. The server then uses the key to try to decrypt the window and the time stamp. The result is random numbers. After a few thousand trials, however, the random window/time stamp pair is likely to pass the authentication system. The window verifier makes the process of guessing the right credential much more difficult.

Decrypting the Conversation Key

When the server receives the transmission from the client, the following occurs:

1. The keyserver that is local to the server looks up the client's public key in the public key database.
2. The keyserver uses the client's public key and the server's secret key to deduce the common key. The common key is the same common key that is computed by the client. Only the server and the client can calculate the common key because the calculation requires knowing one of the secret keys.

3. The kernel uses the common key to decrypt the conversation key.
4. The kernel calls the keyserver to decrypt the client's time stamp with the decrypted conversation key.

Storing Information on the Server

After the server decrypts the client's time stamp, the server stores four items of information in a credential table:

- The client's computer name
- The conversation key
- The window
- The client's time stamp

The server stores the first three items for future use. The server stores the time stamp to protect against replays. The server accepts only time stamps that are chronologically greater than the last time stamp seen, so any replayed transactions are guaranteed to be rejected.

Note – Implicit in these procedures are the name of the caller, who must be authenticated in some manner. The keyserver cannot use DES authentication to authenticate the caller because it would create a deadlock. To solve this problem, the keyserver stores the secret keys by user ID (UID) and grants requests only to local root processes.

Returning the Verifier to the Client

The server returns a verifier to the client, which includes the following:

- The index ID, which the server records in its credential cache
- The client's time stamp minus 1, which is encrypted by the conversation key

The reason for subtracting 1 from the time stamp is to ensure that the time stamp is invalid. So, the time stamp cannot be reused as a client verifier.

Client Authenticates the Server

The client receives the verifier and authenticates the server. The client knows that only the server could have sent the verifier because only the server knows what time stamp the client sent.

Additional Transactions

With every transaction after the first transaction, the client returns the index ID to the server in its second transaction and sends another encrypted time stamp. The server sends back the client's time stamp minus 1, which is encrypted by the conversation key.

Administering Diffie-Hellman Authentication

A system administrator can implement policies that help secure the network. The level of security that is required differs with each site. This section provides instructions for some tasks that are associated with network security.

▼ How to Restart the Keyserver

1. **Become superuser or assume an equivalent role.**
2. **Verify whether the `keyserv` daemon is running.**

```
# ps -ef | grep keyserv
root   100      1  16   Apr 11    ?         0:00 /usr/sbin/keyserv
root  2215   2211   5   09:57:28  pts/0     0:00 grep keyserv
```

3. **Start the keyserver if the process isn't running.**

```
# /usr/sbin/keyserv
```

▼ How to Set Up a `root` Key in NIS+ Credentials for Diffie-Hellman Authentication

For detailed description of NIS+ security, see *System Administration Guide: Naming and Directory Services (FNS and NIS+)*.

1. **Become superuser or assume an equivalent role.**
2. **Edit the `/etc/nsswitch.conf` file, and add the following line:**

```
publickey: nisplus
```

3. **Initialize the NIS+ client.**

```
# nisinit -cH hostname
```

hostname is the name of a trusted NIS+ server that contains an entry in its tables for the client machine.

4. **Add the client to the `cred` table by typing the following commands:**

```
# nisaddcred local
# nisaddcred des
```

5. **Verify the setup by using the `keylogin` command.**

If you are prompted for a password, the procedure has succeeded.

Example—Setting Up a New Key for `root` on an NIS+ Client

The following example uses the host `pluto` to set up `earth` as an NIS+ client. You can ignore the warnings. The `keylogin` command is accepted, verifying that `earth` is correctly set up as a secure NIS+ client.

```
# nisinit -cH pluto
NIS Server/Client setup utility.
This machine is in the North.Abc.COM. directory.
Setting up NIS+ client ...
All done.
# nisaddcred local
# nisaddcred des
DES principal name : unix.earth@North.Abc.COM
Adding new key for unix.earth@North.Abc.Com (earth.North.Abc.COM.)

Network password: xxx      Press Return
Warning, password differs from login password.
Retype password: xxx      Press Return

# keylogin
Password:
#
```

▼ How to Set Up a New User Key That Uses NIS+ Credentials for Diffie-Hellman Authentication

1. **Add the user to the `cred` table on the root master server by typing the following command:**

```
# nisaddcred -p unix.UID@domain-name -P username.domain-name. des
```

Note that, in this case, the *username.domain-name* must end with a dot (.)

2. **Verify the setup by logging in as the client and typing the `keylogin` command.**

Example—Setting Up a New Key for an NIS+ User

The following example shows how DES authorization is given to a user who is named `george`.

```
# nisaddcred -p unix.1234@North.Abc.com -P george.North.Abc.COM. des
DES principal name : unix.1234@North.Abc.COM
Adding new key for unix.1234@North.Abc.COM (george.North.Abc.COM.)

Password:
Retype password:

# rlogin rootmaster -l george
# keylogin
Password:
#
```

▼ How to Set Up a `root` Key by Using NIS Credentials With Diffie-Hellman Authentication

1. **Become superuser on the client or assume an equivalent role.**
2. **Edit the `/etc/nsswitch.conf` file, and add the following line:**

   ```
   publickey: nis
   ```

3. **Create a new key pair by using the `newkey` command.**

   # **`newkey -h`** *hostname*

   hostname is the name of the client.

Example—Setting Up a New Key for `root` on a NIS Client

The following example shows how to set up `earth` as a secure NIS client.

```
# newkey -h earth
Adding new key for unix.earth@North.Abc.COM
New Password:
Retype password:
Please wait for the database to get updated...
Your new key has been successfully stored away.
#
```

▼ How to Create a New User Key That Uses NIS Credentials With Diffie-Hellman Authentication

1. **Log in to the NIS master server as superuser or assume an equivalent role.**

Only system administrators, when logged in to the NIS master server, can generate a new key for a user.

2. **Create a new key for a user.**

   # **newkey -u** *username*

   username is the name of the user. The system prompts for a password. You can type a generic password. The private key is stored in an encrypted form by using the generic password.

```
# newkey -u george
Adding new key for unix.12345@Abc.North.Acme.COM
New Password:
Retype password:
Please wait for the database to get updated...
Your new key has been successfully stored away.
#
```

3. **Tell the user to log in and type the `chkey -p` command.**

   This command allows the user to re-encrypt his or her private key with a password known only to the user.

```
earth% chkey -p
Updating nis publickey database.
Reencrypting key for unix.12345@Abc.North.Acme.COM
Please enter the Secure-RPC password for george:
Please enter the login password for george:
Sending key change request to pluto...
#
```

Note – The `chkey` command can be used to create a new key-pair for a user.

▼ How to Share and Mount Files With Diffie-Hellman Authentication

Prerequisite

The Diffie-Hellman `publickey` authentication must be enabled on the network. See "How to Set Up a `root` Key in NIS+ Credentials for Diffie-Hellman Authentication" on page 166 and "How to Set Up a `root` Key by Using NIS Credentials With Diffie-Hellman Authentication" on page 168.

To share a file system with Diffie-Hellman authentication:

1. **Become superuser or assume an equivalent role.**
2. **Share the file system with Diffie-Hellman authentication.**

```
# share -F nfs -o sec=dh /filesystem
```

To mount a file system with Diffie-Hellman authentication:

1. **Become superuser or assume an equivalent role.**
2. **Mount the file system with Diffie-Hellman authentication.**

   ```
   # mount -F nfs -o sec=dh server:resource mountpoint
   ```

   The `-o sec=dh` option mounts the file system with `AUTH_DH` authentication.

CHAPTER 10

Using PAM

This chapter covers the Pluggable Authentication Module (PAM) framework. PAM provides a method to "plug in" authentication services and provides support for multiple authentication services.

PAM (Overview)

The Pluggable Authentication Module (PAM) framework lets you "plug in" new authentication technologies without changing system entry services, such as `login`, `ftp`, `telnet`, and so on. You can also use PAM to integrate UNIX login with other security mechanisms like Kerberos. Mechanisms for account, session, and password management can also be "plugged in" by using this framework.

Benefits of Using PAM

The PAM framework allows you to configure the use of system entry services (`ftp`, `login`, `telnet`, or `rsh`, for example) for user authentication. Some benefits that PAM provides are as follows:

- Flexible configuration policy
  - Per application authentication policy
  - The ability to choose a default authentication mechanism
  - Multiple passwords on high-security systems
- Ease of use for the end user

- No retyping of passwords if the passwords are the same for different mechanisms.
- The ability to prompt the user for passwords for multiple authentication methods without having the user enter multiple commands.

- The ability to pass optional parameters to the user authentication services

PAM Components

The PAM software consists of a library, several modules, and a configuration file. New versions of several commands or daemons that take advantage of the PAM interfaces are also included.

The following figure illustrates the relationship between the applications, the PAM library, the `pam.conf` file, and the PAM modules.

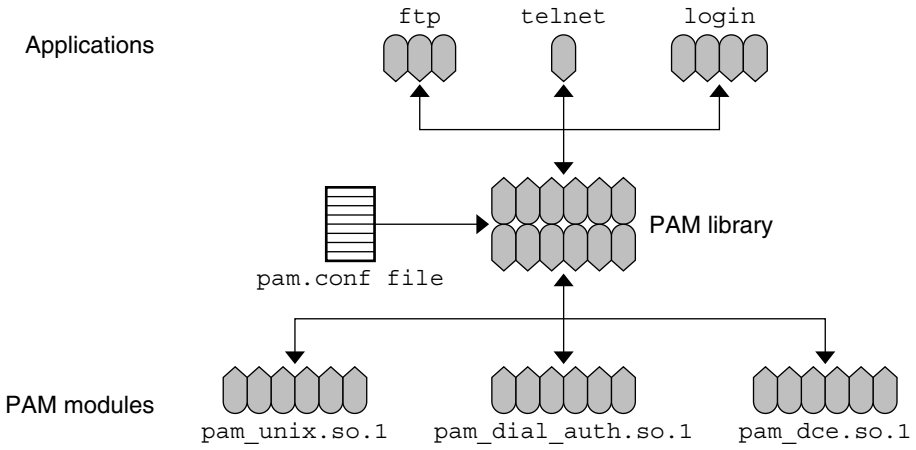


FIGURE 10–1 How PAM Works

The applications, such as `ftp`, `telnet`, and `login`, use the PAM library to call they configuration policy. The `pam.conf` file defines which modules to use, and in what order the modules are to be used with each application. Responses from the modules are passed back through the library to the application.

The following sections describe the relationship between the PAM components and the applications.

PAM Library

The PAM library provides the framework to load the appropriate modules and to manage the stacking process. The PAM library provides a generic structure to which all of the modules can plug in. See the pam.3PAM XREF man page for more information.

Password-Mapping Feature

The stacking feature can require that a user remembers several passwords. With the *password-mapping* feature, the primary password is used to decrypt the other passwords. The user does not need to remember or enter multiple passwords. The other option is to synchronize the passwords across each authentication mechanism. This strategy could increase the security risk, because the mechanism security is limited by the least secure password method that is used in the stack.

Changes to PAM for the Solaris 9 Release

The Solaris 9 release includes several enhancements to the PAM service. The following list highlights the most important changes:

- To accommodate proper stacking, the `pam_unix` module is broken into single service modules. These modules provide the same capabilities as in the `pam_unix` module. The capabilities are provided by the following modules:
  - `pam_authtok_get`
  - `pam_authtok_check`
  - `pam_authtok_store`
  - `pam_unix_auth`
  - `pam_dhkeys`
  - `pam_passwd_auth`

  See "PAM Modules" on page 177 for information about the new modules.
- The `ssh` service name was added. See "Valid Service Names for PAM" on page 180 for information about the PAM services.
- The PAM configuration file was updated. See "Generic `pam.conf` File" on page 183 for information about the configuration file.

Changes to PAM for the Solaris 9 Update 2 Release

Update 2 includes a new binding control flag. This flag provides the ability to skip additional authentication if the service module returns success and if no preceding required modules have failed. The control flag is documented in the `pam.conf`(4) man page and in "PAM Control Flags" on page 181.

PAM (Tasks)

This section discusses some tasks that might be required to make the PAM framework fully functional. In particular, you should be aware of some security issues that are associated with the PAM configuration file.

PAM (Task Map)

Task	Description	For Instructions
Plan for your PAM Installation	Consider configuration issues and make decisions about them before you start the software configuration process.	"Planning for PAM" on page 174
Add new PAM modules	Sometimes, site-specific modules must be written and installed to cover requirements that are not part of the generic software. This procedure covers the installation process.	"How to Add a PAM Module" on page 175
Block access through `~/.rhosts`	Steps to further increase security by preventing access through `~/.rhosts`.	"How to Prevent Unauthorized Access From Remote Systems With PAM" on page 176
Initiate error reporting	Steps to start the reporting of PAM error messages through `syslog`.	"How to Initiate PAM Error Reporting" on page 176

Planning for PAM

When you are deciding how best to use PAM in your environment, start by focusing on these issues:

- Determine what your needs are, especially which modules you should select.
- Identify the services that need special attention. Use `OTHER` if appropriate.
- Decide on the order in which the modules should be run.
- Select the control flag for each module.
- Choose any options that are necessary for each module.

Here are some suggestions to consider before you change the PAM configuration file:

- Use the `OTHER` entry for each module type so that every application does not have to be included.
- Make sure to consider the security implications of the `sufficient` and `optional` control flags.

- Review the man pages that are associated with the modules. These man pages can help you understand how each module functions, what options are available, and the interactions between stacked modules.

Caution – If the PAM configuration file is misconfigured or the file becomes corrupted, even superuser might be unable to log in. Since the `sulogin` command does not use PAM, superuser would then be required to boot the machine into single-user mode and fix the problem.

After you change the `/etc/pam.conf` file, review the file as much as possible while you are still logged in as superuser. Test all the commands that might have been affected by your changes. An example is adding a new module to the `telnet` service. In this example, you use the `telnet` command and verify that your changes make the service behave as expected.

▼ How to Add a PAM Module

1. **Become superuser or assume an equivalent role.**
2. **Determine which control flags and which other options should be used.**
   Refer to "PAM Modules" on page 177 information on the modules.
3. **Copy the new module to `/usr/lib/security/sparcv9`.**
   In the Solaris 8 release, the module should be copied to `/usr/lib/security`.
4. **Set the permissions so that the module file is owned by `root` and that permissions are `555`.**
5. **Edit the PAM configuration file, `/etc/pam.conf`, and add this module to the appropriate services.**

Verification

You must test *before* the system is rebooted in case the configuration file is misconfigured. Run `rlogin`, `su`, and `telnet` before you reboot the system. The service might be a daemon that is spawned only once when the system is booted. Then you must reboot the system before you can verify that the module has been added.

How to Prevent Unauthorized Access From Remote Systems With PAM

Remove the `rlogin` `auth` `rhosts_auth.so.1` entry from the PAM configuration file. This step prevents the reading of the `~/.rhosts` files during an `rlogin` session. Therefore, this step prevents unauthenticated access to the local system from remote systems. All `rlogin` access requires a password, regardless of the presence or contents of any `~/.rhosts` or `/etc/hosts.equiv` files.

Note – To prevent other unauthenticated access to the `~/.rhosts` files, remember to disable the `rsh` service. The best way to disable a service is to remove the service entry from the `/etc/inetd.conf` file. Changing the PAM configuration file does not prevent the service from being started.

▼ How to Initiate PAM Error Reporting

1. **Edit the `/etc/syslog.conf` file to add any of the following entries for PAM error reporting:**
   - `auth.alert` – Messages about conditions that should be fixed immediately
   - `auth.crit` – Critical messages
   - `auth.err` – Error messages
   - `auth.info` – Informational messages
   - `auth.debug` – Debugging messages
2. **Restart the `syslog` daemon, or send a `SIGHUP` signal to the daemon to activate the PAM error reporting.**

Example—Initiating PAM Error Reporting

In the following example, all alert messages are displayed on the console. Critical messages are mailed to root. Informational and debug messages are added to the `/var/log/pamlog` file.

```
auth.alert    /dev/console
auth.crit    'root'
auth.info;auth.debug    /var/log/pamlog
```

Each line in the log contains a time stamp, the name of the system that generated the message, and the message. The `pamlog` file is capable of logging a large amount of information.

PAM (Reference)

PAM uses run-time pluggable modules to provide authentication for system entry services. A stacking feature is provided to let you authenticate users through multiple services. Also provided is a password-mapping feature to not require that users remember multiple passwords.

PAM Modules

Every PAM module implements a specific mechanism. When you set up PAM authentication, you need to specify both the module and the module type, which defines what the module does. More than one module type, such as auth, account, session, or password, can be associated with each module.

The following table describes every PAM module, and includes the module name and the module file name. The path of each module is determined by the instruction set that is available in the Solaris release that is installed. The default path to the modules is `/usr/lib/security/$ISA`. The value for *$ISA* could be `sparc` or `i386`. See the `isalist`(5) man page for more information.

TABLE 10–1 PAM Modules

Module Name and Module File Name	Description
`authtok_check` `pam_authtok_check.so.1`	Provides support for password management. This module performs various checks on passwords. Those check are for the length of the password, for circular shift of the login name, for password complexity, and for the amount of variation between new passwords and old passwords. See `pam_authtok_check`(5) for more information.
`authtok_get` `pam_authtok_get.so.1`	Provides password prompting for authentication and password management. See `pam_authtok_get`(5) for more information.
`authtok_store` `pam_authtok_store.so.1`	Provides support for authentication only. This module updates the authentication token for the user. After the successful update, the module stores the token in the specified repository or default repository. See `pam_authtok_store`(5) for more information.
`dhkeys` `pam_dhkeys.so.1`	Provides support for Diffie-Hellman key management in authentication. This module supports Secure RPC authentication and Secure RPC authentication token management. See `pam_dhkeys`(5) for more information.

TABLE 10–1 PAM Modules *(Continued)*

Module Name and Module File Name	Description
`dial_auth` `pam_dial_auth.so.1`	Can only be used for authentication. This module uses data that is stored in the `/etc/dialups` and `/etc/d_passwd` files for authentication. This module is mainly used by the `login` command. See `pam_dial_auth`(5) for more information.
`krb5` `pam_krb5_auth.so.1`	Provides support for authentication, account management, session management, and password management. Kerberos credentials are used for authentication. See `pam_krb5`(5) for more information.
`ldap` `pam_ldap.so.1`	Provides support for authentication and password management. Data from an LDAP server are used for authentication. See `pam_ldap`(5) for more information.
`projects` `pam_projects.so.1`	Provides support for account management. See `pam_projects`(5) for more information.
`rhosts_auth` `pam_rhosts_auth.so.1`	Can only be used for authentication. This module uses data that is stored in the `~/.rhosts` and `/etc/host.equiv` files through the `ruserok()` routine. This module is mainly used by the `rlogin` and `rsh` commands. See `pam_rhosts_auth`(5) for more information.
`roles` `pam_roles.so.1`	Provides support for account management only. The RBAC `user_attr` database determines which roles a user can assume. See `pam_roles`(5) for more information.
`sample` `pam_sample.so.1`	Provides support for authentication, account management, session management, and password management. Used for testing. See `pam_sample`(5) for more information.
`smartcard` `pam_smartcard.so.1`	Provides support for authentication only. See `pam_smartcard`(5) for more information.
`unix` `pam_unix.so.1`	Provides support for authentication, account management, session management, and password management. Any of the four module type definitions can be used with this module. This module uses UNIX passwords for authentication. In the Solaris environment, the selection of appropriate name services to get password records is controlled through the `/etc/nsswitch.conf` file. See `pam_unix`(5) for more information.
`unix_account` `pam_unix_account.so.1`	Provides support for account management. This module retrieves password aging information from the repository that is specified in the `nsswitch.conf` file. Then the module verifies that the password and the user's account have not expired. See `pam_unix_account`(5) for more information.

TABLE 10–1 PAM Modules *(Continued)*

Module Name and Module File Name	Description
`unix_auth` `pam_unix_auth.so.1`	Provides support for authentication. This module verifies the password that is contained in the PAM handle. The module checks that the user's password matches the password in the specified repository or default repository. See `pam_unix_auth`(5) for more information.
`unix_session` `pam_unix_session.so.1`	Provides support for session management. This module initiates session management by updating the `/var/adm/lastlog` file. See `pam_unix_session`(5) for more information.

For security reasons, these module files must be owned by `root` and must not be writable through `group` or `other` permissions. If the file is not owned by `root`, PAM does not load the module.

PAM Module Types

You need to understand the PAM module types because the types define the interface to the module. Here are the four types of run-time PAM modules:

- The *authentication modules* provide authentication for the users. The modules also allow for credentials to be set, refreshed, or destroyed. In addition, the modules provide a valuable administration tool for user identification.
- The *account modules* check for password aging, account expiration, and access hour restrictions. After the user is identified through the authentication modules, the account modules determine if the user should be given access.
- The *session modules* manage the opening and the closing of an authentication session. These modules also can log activity or provide for cleanup after the session is over.
- The *password modules* allow for changes to the actual password.

PAM Configuration File

The PAM configuration file, `/etc/pam.conf`, determines the authentication services to be used, and the order in which the services are used. This file can be edited to select authentication mechanisms for each system entry application.

PAM Configuration File Syntax

The PAM configuration file consists of entries with the following syntax:

service_name module_type control_flag module_path module_options

service_name	Is the name of the service, for example, `ftp`, `login`, `telnet`.
module_type	Is the module type for the service. For more information, see "PAM Module Types" on page 179.
control_flag	Determines the continuation or failure behavior for the module.
module_path	Specifies the path to the library object that implements the service.
module_options	Specifies the options that are passed to the service modules.

You can add comments to the `pam.conf` file by starting the line with a `#` (pound sign). Use white spaces or tabs to delimit the fields.

Note – An entry in the PAM configuration file is ignored if one of the following conditions exists: the line has less than four fields, an invalid value is given for *module_type* or *control_flag*, or the named module does not exist.

Valid Service Names for PAM

The following table lists:

- The valid service names
- The module types that can be used with that service
- The daemon or command that is associated with the service name

Some module types are not appropriate for each service. For example, the `password` module type is appropriate for only the `passwd` command. Also, because the `passwd` command is not concerned with authentication, no `auth` module type is associated with the service.

TABLE 10–2 Valid Service Names for the `/etc/pam.conf` File

Service Name	Daemon or Command	Applicable Module Types
`cron`	`/usr/sbin/cron`	`auth, account`
`dtlogin`	`/usr/dt/bin/dtlogin`	`auth, account, session`
`dtsession`	`/usr/dt/bin/dtsession`	`auth`

TABLE 10–2 Valid Service Names for the `/etc/pam.conf` File *(Continued)*

Service Name	Daemon or Command	Applicable Module Types
`ftp`	`/usr/sbin/in.ftpd`	`auth, account, session`
`init`	`/usr/sbin/init`	`session`
`login`	`/usr/bin/login`	`auth, account, session`
`passwd`	`/usr/bin/passwd`	`password`
`ppp`	`/usr/bin/ppp`	`auth, account, session`
`rexd`	`/usr/sbin/rpc.rexd`	`account, session`
`rlogin`	`/usr/sbin/in.rlogind`	`auth, account, session`
`rsh`	`/usr/sbin/in.rshd`	`auth, account, session`
`sac`	`/usr/lib/saf/sac`	`session`
`ssh`	`/usr/bin/ssh`	`auth, account, session`
`su`	`/usr/bin/su`	`auth, account`
`telnet`	`/usr/sbin/in.telnetd`	`auth, account, session`
`ttymon`	`/usr/lib/saf/ttymon`	`session`
`uucp`	`/usr/sbin/in.uucpd`	`auth, account, session`

PAM Control Flags

To determine the continuation or failure behavior from a module, you must select a *control flag* for each entry in the PAM configuration file, `/etc/pam.conf`. Each module in a stack can determine the success or failure of the request.

Continuation behavior defines if any following modules are checked. Depending on the response from a particular module, you can decide to skip any additional modules.

Failure behavior defines how error messages are logged or reported. Failures are either optional or required. A required failure causes that request to fail, even if other modules succeed. An optional failure does not always cause the request to fail.

Even though these flags apply to all module types, the following explanation assumes that these flags are being used for authentication modules. The control flags are as follows:

- `binding` – With this control flag, if the module is successful and no preceding modules that are flagged as required have failed, then skip the remaining modules. If a failure is returned, record a required failure and continue processing the stack.

The `binding` control flag is like the `required` control flag, except that no additional module checking is done if the module is successful. A failure using this flag prevents the request from being successful, regardless of the response of any other modules. A success using this flag makes the request successful if no preceding required modules failed.

- `required` – With this control flag, if the module is successful, record a required success and continue checking any following modules. If the module fails, and if this is the first required failure, save the error message and continue checking the stack. If this is not the first failure, then continue checking the stack.

  The `required` control flag should be used when a particular module must succeed for the request to be successful. A failure when using this flag prevents the request from being successful, regardless of the response of any other modules. A success when using this flag does not mean that the request is successful. All of the responses from the other required modules in the stack must be successful for the request to succeed.

- `requisite` – With this control flag, if the module is successful, record a required success and continue checking any following modules. If the module fails, record a required failure, return the error message of the first required failure, and skip any additional checking.

  The `requisite` control flag is like the `required` control flag, except that no additional module checking is done if the module fails. A failure when using this flag prevents the request from being successful, regardless of the response of any other modules. A success when using this flag does not mean that the request is successful. All of the responses from the other required modules in the stack must be successful for the request to succeed.

- `optional` – With this control flag, if the module is successful, record an optional success and continue checking the stack. If the module fails, record an optional failure and continue checking the stack.

  The `optional` control flag should be used when successful authentication in the stack is enough for a user to be authenticated. This flag should only be used if this particular mechanism does not need to succeed. The success or failure of the request, is determined by any required failures or successes.

  If your users need to have permission associated with a specific mechanism to get their work done, then you should not label the module as `optional`.

- `sufficient` – With this control flag, if the module is successful, and no preceding modules that are flagged as required have failed, then skip the remaining modules. If the module fails, record an option failure and continue checking the stack.

  The `sufficient` control flag is like the `optional` control flag, except that no additional module checking is done if the module succeeds. A success when using this flag makes the request successful if no preceding required modules failed. A failure when using this flag causes the request to fail if no other modules succeeded.

More information about these control flags is provided in the following section, which describes the default `/etc/pam.conf` file.

Generic pam.conf File

The generic /etc/pam.conf file specifies the following actions:

1. When the login command is run, authentication must succeed for the pam_authtok_get, pam_dhkeys, pam_auth_unix, and the pam_dial_auth modules.
2. For the rlogin command, authentication through the pam_authtok_get, pam_dhkeys, and pam_auth_unix modules must succeed if authentication through pam_rhost_auth fails.
3. The sufficient control flag indicates that for the rlogin command, the successful authentication that is provided by the pam_rhost_auth module is sufficient. The next entry is ignored.
4. Most of the other commands that require authentication require successful authentication through the pam_authtok_get, pam_dhkeys, and pam_auth_unix modules.
5. For the rsh command, authentication through the pam_rhost_auth module is flagged as sufficient. No other authentication is required if authentication through the pam_rhost_auth module succeeds.

The OTHER service name allows a default to be set for any other commands that require authentication and are not included in the file. The OTHER option simplifies administration of the file, since many commands that are using the same module can be covered by using only one entry. Also, the OTHER service name, when used as a "catch-all," can ensure that each access is covered by one module. By convention, the OTHER entry is included at the bottom of the section for each module type.

Normally, the entry for the *module_path* is "root-relative." If the file name that you enter for *module_path* does not begin with a slash (/), the path /usr/lib/security/$ISA precedes the file name. A full path name must be used for modules that are located in other directories. The values for the *module_options* can be found in the man pages for the module. For example, the UNIX module is covered in the pam_unix(5) man page.

```
login   auth required          pam_authtok_get.so.1
login   auth required          pam_dhkeys.so.1
login   auth required          pam_unix_auth.so.1
login   auth required          pam_dial_auth.so.1
```

In this example, the login service specifies authentication through all four authentication modules. A login command fails if any one of the modules returns an error.

CHAPTER 11

Using Solaris Secure Shell (Tasks)

Solaris Secure Shell enables a user to securely access a remote host over an unsecured network. The shell provides commands for remote login and remote file transfer. The following is a list of the information in this chapter.

Introduction to Solaris Secure Shell

In Solaris Secure Shell, authentication is provided by the use of passwords, public keys, or both. All network traffic is encrypted. Thus, Solaris Secure Shell prevents a would-be intruder from being able to read an intercepted communication or from spoofing the system.

Solaris Secure Shell can also be used as an on-demand virtual private network, or VPN. A VPN can forward X Window system traffic or connect individual port numbers between the local machines and remote machines over an encrypted network link.

With Solaris Secure Shell, you can perform these actions:

- Log in to another host securely over an unsecured network
- Copy files securely between the two hosts
- Run commands securely on the remote host

Solaris Secure Shell supports two versions of the Secure Shell protocol. Version 1 is the original version of the protocol. Version 2 is more secure, and amends some of the basic security design flaws of Version 1. Version 1 is provided only to assist users who are migrating to Version 2. Users are strongly discouraged from using Version 1.

Note – Hereafter in this text, v1 is used to represent Version 1, and v2 is used to represent Version 2.

The requirements for Solaris Secure Shell authentication are as follows:

- **User authentication –** A user can be authenticated through either of the following:
  - **Passwords –** The user supplies the account password as in the login process.
  - **Public keys –** The user can create a public/private key pair that is stored on the local host. The remote hosts are provided with the public key, which is required to complete the authentication.

    The source host maintains the private key, and target hosts are provided with the public key that is needed to complete authentication. Public key authentication is a stronger authentication mechanism than password authentication, because the private key never travels over the network. The public/private key pair is stored in the user's home directory under the `.ssh` subdirectory. The following table shows the default names for the identity files, which store the public keys and private keys.

    TABLE 11–1 Naming Conventions for SSH Identity Files

Private Key, Public Key	Cipher and Protocol Version
`identity`, `identity.pub`	RSA v1
`id_rsa`, `id_rsa.pub`	RSA v2
`id_dsa`, `id_dsa.pub`	DSA v2

- **Host authentication –** Host authentication requires the remote host to have access to the local host's public key. A copy of the local host's public key is stored in `$HOME/.ssh/known_hosts` on the remote host.

The following table shows the authentication methods, the compatible protocol versions, the local host and remote host requirements, and the relative security. Note that the default method is password-based authentication.

TABLE 11–2 Authentication Methods for Solaris Secure Shell

Authentication Method (Protocol Version)	Local Host Requirements	Remote Host Requirements	Security Level
Password-based (v1 or v2)	User account	User account	Medium

TABLE 11–2 Authentication Methods for Solaris Secure Shell *(Continued)*

Authentication Method (Protocol Version)	Local Host Requirements	Remote Host Requirements	Security Level
RSA/DSA public key (v2)	User account Private key in `$HOME/.ssh/id_rsa` or `$HOME/.ssh/id_dsa` Public key in `$HOME/.ssh/id_rsa.pub` or `$HOME/.ssh/id_dsa.pub`	User account User's public key (`id_rsa.pub` or `id_dsa.pub`) in `$HOME/.ssh/authorized_keys`	Strong
RSA public key (v1)	User account Private key in `$HOME/.ssh/identity` Public key in `$HOME/.ssh/identity.pub`	User account User's public key (`identity.pub`) in `$HOME/.ssh/authorized_keys`	Strong
`.rhosts` with RSA (v1)	User account	User account Local host name in `/etc/hosts.equiv`, `/etc/shosts.equiv`, `$HOME/.rhosts`, or `$HOME/.shosts` Local host public key in `$HOME/.ssh/known_hosts` or `/etc/ssh/ssh_known_hosts`	Medium
`.rhosts` only (v1 or v2)	User account	User account Local host name in `/etc/hosts.equiv`, `/etc/shosts.equiv`, `$HOME/.rhosts`, or `$HOME/.shosts`	Weak

Using Solaris Secure Shell (Task Map)

Task	Description	For Instructions
Create a public/private key pair	The use of public/private key pairs is the preferred method for authenticating yourself and encrypting your communications.	"How to Create a Public/Private Key Pair" on page 188

Task	Description	For Instructions
Log in with Solaris Secure Shell	Encrypted Secure Shell communication is enabled by logging in remotely through a process similar to using `rsh`.	"How to Log In to Another Host With Solaris Secure Shell" on page 189
Log in without a password with Solaris Secure Shell	You can log in using Secure Shell without having to provide a password by using `ssh-agent`. The `ssh-agent` command can be run manually or from a startup script.	"How to Log In With No Password With the `ssh-agent` Command" on page 190 "How to Set Up the `ssh-agent` Command to Run Automatically" on page 191
Port forwarding in Solaris Secure Shell	You can specify a local port or a remote port to be used in a Secure Shell connection over TCP.	"How to Use Solaris Secure Shell Port Forwarding" on page 192
Copy files with Solaris Secure Shell	You can copy remote files securely.	"How to Copy Files With Solaris Secure Shell" on page 194
Transfer files with Solaris Secure Shell	You can log in to a remote host with Secure Shell by using transfer commands that are similar to `ftp`.	"How to Transfer Files Remotely With the `sftp` Command" on page 194
Connect from a host inside a firewall to a host on the outside	Secure shell provides commands that are compatible with HTTP or SOCKS5. The commands can be specified in a configuration file or on the command line.	"How to Set Up Default Connections to Hosts Outside a Firewall" on page 195 "Example—Connecting to Hosts Outside a Firewall From the Command Line" on page 196

Using Solaris Secure Shell

▼ How to Create a Public/Private Key Pair

The standard procedure for creating a Solaris Secure Shell public/private key pair follows. For additional options, see the `ssh-keygen`(1) man page.

1. **Start the key generation program.**

```
myLocalHost% ssh-keygen
Generating public/private rsa key pair.
...
```

2. **Enter the path to the file that will hold the key.**

By default, the file name `id_rsa`, which represents an RSA v2 key, appears in parentheses. You can select this file by pressing the Return key. Or, you can type an alternative filename.

```
Enter file in which to save the key (/home/johndoe/.ssh/id_rsa): <Return>
```

The public key name is created automatically. The string `.pub` is appended to the private key name.

3. **Enter a passphrase for using your key.**

   This passphrase is used for encrypting your private key. A good passphrase is 10-30 characters long, mixes alphabetic and numeric characters, and avoids simple English prose and English names. A null entry means no passphrase is used. A null entry is *strongly discouraged* for user accounts. Note that the passphrase is not displayed when you type it in.

   ```
   Enter passphrase (empty for no passphrase):     <Type the passphrase>
   ```

4. **Re-enter the passphrase to confirm it.**

   ```
   Enter same passphrase again:     <Type the passphrase>
   Your identification has been saved in /home/jdohnoe/.ssh/id_rsa.
   Your public key has been saved in /home/johndoe/.ssh/id_rsa.pub.
   The key fingerprint is:
   0e:fb:3d:57:71:73:bf:58:b8:eb:f3:a3:aa:df:e0:d1 johndoe@myLocalHost
   ```

5. **Check the results.**

   The key fingerprint, which is a colon-separated series of 2-digit hexadecimal values, is displayed. Check that the path to the key is correct. In the example, the path is `/home/johndoe/.ssh/id_rsa.pub`. At this point, you have created a public/private key pair.

6. **Set up the `authorized_keys` file on the destination host.**

   a. **Copy the `id_rsa.pub` file to the destination host. Type the command on one line with no backslash.**

      ```
      myLocalHost% cat $HOME/.ssh/id_rsa.pub | ssh myRemoteHost \
       'cat >> .ssh/authorized_keys && echo "Key uploaded successfully."'
      ```

   b. **When you are prompted, supply your login password.**

      When the file is copied, the phrase "Key uploaded successfully." is displayed.

▼ How to Log In to Another Host With Solaris Secure Shell

1. **Use the `ssh` command, specifying the name of the remote host.**

   ```
   myLocalHost% ssh myRemoteHost
   ```

   The first time that you run the `ssh` command, a prompt questions the authenticity of the remote host:

```
The authenticity of host 'myRemoteHost' can't be established.
RSA key fingerprint in md5 is: 04:9f:bd:fc:3d:3e:d2:e7:49:fd:6e:18:4f:9c:26
Are you sure you want to continue connecting(yes/no)?
```

This prompt is normal. You should type **yes** and continue. If you have used `ssh` in the past on this remote host, then the prompt is not normal. You should check for a breach in your security.

2. **Enter the Solaris Secure Shell passphrase and the account password when you are prompted for them.**

```
Enter passphrase for key '/home/johndoe/.ssh/id_rsa': <Return>
johndoe@myRemoteHost's password: <Return>
Last login: Fri Jul 20 14:24:10 2001 from myLocalHost
myRemoteHost%
```

Conduct transactions on the remote host. The commands that you send are encrypted. Any responses that you receive are encrypted.

Note – If you want to subsequently change your passphrase, use the `ssh-keygen` command with the `-p` option.

---

3. **When you are finished with your remote session, type exit or use your usual method for exiting your shell.**

```
myRemoteHost% exit
myRemoteHost% logout
Connection to myRemoteHost closed
myLocalHost%
```

▼ How to Log In With No Password With the `ssh-agent` Command

If you want to omit passphrase and password entry when you are using Solaris Secure Shell, you can use the agent daemon. Use the `ssh-agent` command at the beginning of the session. Then, store your private keys with the agent by using the `ssh-add` command. If you have different accounts on different hosts, add those keys that you intend to use in the session.

You can start the agent manually when needed as described in the following procedure. Or, you can set the agent to run automatically at the start of every session as described in "How to Set Up the `ssh-agent` Command to Run Automatically" on page 191.

1. **Start the agent daemon.**

The `ssh-agent` command starts the agent daemon and displays its process ID.

```
myLocalHost% eval `ssh-agent`
Agent pid 9892
```

```
myLocalHost%
```

2. **Add your private key to the agent daemon.**

   The `ssh-add` command adds your private key to the agent daemon so that subsequent Secure Shell activity does not prompt you for the passphrase.

```
myLocalHost% ssh-add
Enter passphrase for /home/johndoe/.ssh/id_rsa:
Identity added: /home/johndoe/.ssh/id_rsa(/home/johndoe/.ssh/id_rsa)
myLocalHost%
```

3. **Start a Solaris Secure Shell session.**

```
myLocalHost% ssh myRemoteHost
```

Example—Using `ssh-add` Options

You can use `ssh-add` to add other keys to the daemon as well. For example, you might concurrently have DSA v2, RSA v2, and RSA v1 keys. To list all keys that are stored in the daemon, use the `-l` option. To delete a single key from the daemon, use the `-d` option. To delete all keys, use the `-D` option.

```
myLocalHost% eval `ssh-agent`
Agent pid 3347
myLocalHost% ssh-add
Enter passphrase for /home/johndoe/.ssh/id_rsa:
Identity added: /home/johndoe/.ssh/id_rsa(/home/johndoe/.ssh/id_rsa)
myLocalHost% ssh-add /home/johndoe/.ssh/id_dsa
Enter passphrase for /home/johndoe/.ssh/id_dsa:     <type passphrase>
Identity added:
/home/johndoe/.ssh/id_dsa(/home/johndoe/.ssh/id_dsa)
myLocalHost% ssh-add -l
md5 1024 0e:fb:3d:53:71:77:bf:57:b8:eb:f7:a7:aa:df:e0:d1
/home/johndoe/.ssh/id_rsa(RSA)
md5 1024 c1:d3:21:5e:40:60:c5:73:d8:87:09:3a:fa:5f:32:53
/home/johndoe/.ssh/id_dsa(DSA)
myLocalHost% ssh-add -d
Identity removed:
/home/johndoe/.ssh/id_rsa(/home/johndoe/.ssh/id_rsa.pub)
/home/johndoe/.ssh/id_dsa(DSA)
```

▼ How to Set Up the `ssh-agent` Command to Run Automatically

You can avoid providing your passphrase and password whenever you use Secure Shell by starting an agent daemon, `ssh-agent`. You can start the agent daemon from the `.dtprofile` script.

1. **To start the agent daemon automatically, add the following lines to the end of the `$HOME/.dtprofile` script:**

```
if [ "$SSH_AUTH_SOCK" = "" -a -x /usr/bin/ssh-agent ]; then
                eval `/usr/bin/ssh-agent`
fi
```

2. **To terminate the Secure Shell agent daemon when you exit the CDE session, add the following to the `$HOME/.dt/sessions/sessionexit` script:**

```
if [ "$SSH_AGENT_PID" != "" -a -x /usr/bin/ssh-agent ]; then
                /usr/bin/ssh-agent -k
fi
```

   This entry ensures that no one can use the Secure Shell agent after the CDE session is terminated.

3. **Start a Solaris Secure Shell session.**

```
myLocalHost% ssh myRemoteHost
```

   There is no prompt for a passphrase.

▼ How to Use Solaris Secure Shell Port Forwarding

You can specify that a local port be forwarded to a remote host. Effectively, a socket is allocated to listen to the port on the local side. The connection from this port is made over a secure channel to the remote host. For example, you might specify port 143 to obtain email remotely with IMAP4. Similarly, a port can be specified on the remote side.

Note – Secure Shell port forwarding must use TCP connections. Secure Shell does not support UDP connections.

- **To set a local port to be forwarded, specify two ports. Specify the local port to listen to, and specify the remote host and port to forward to.**

```
myLocalHost% ssh -L localPort:remoteHost:remotePort
```

- **To set a remote port to receive a secure connection, specify two ports. Specify the remote port to listen to, and specify the local host and port to forward to.**

```
myLocalHost% ssh -R remotePort:localHost:localPort
```

Example—Using Local Port Forwarding to Receive Mail

The following example demonstrates how you can use local port forwarding to receive mail securely from a remote server.

```
myLocalHost% ssh -L 9143:myRemoteHost:143 myRemoteHost
```

This command forwards connections to port `9143` on `myLocalHost` to port `143`, which is the IMAP v2 server port on `myRemoteHost`. When the user launches a mail application, the user needs to specify the local port number. An example that uses the `dtmail` command is shown in Figure 11–1.

Note that the term *localhost* in this case and in "Example—Using Remote Port Forwarding to Communicate Outside of a Firewall" on page 193 refers to the keyword that designates the user's local host. The *localhost* keyword should not be confused with *myLocalHost*. The `myLocalHost` variable is the hypothetical host name that identifies a local host in the examples in this chapter.

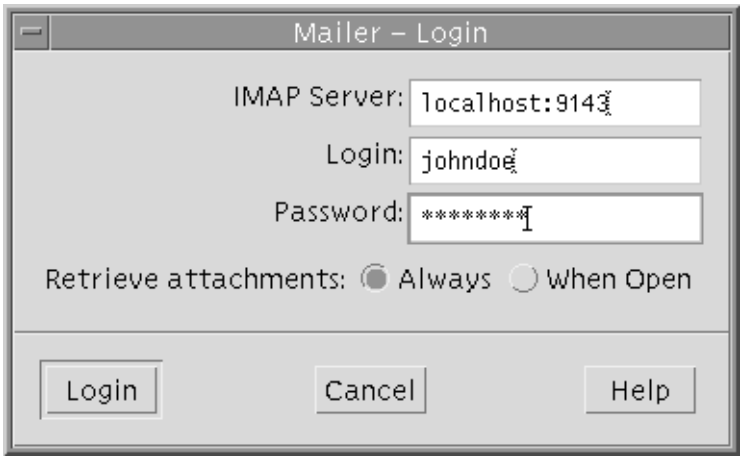


FIGURE 11–1 Specifying Port Forwarding for Email

Example—Using Remote Port Forwarding to Communicate Outside of a Firewall

This example demonstrates how a user in an enterprise environment can forward connections from a host on an external network to a host inside a corporate firewall.

```
myLocalHost% ssh -R 9022:myLocalHost:22 myOutsideHost
```

This command forwards connections to port `9022` on `myOutsideHost` to port `22`, the `sshd` server, on the local host.

```
myOutsideHost% ssh -p 9022 localhost
myLocalHost%
```

This command demonstrates how after the remote forwarding connection has been established, the user can use the `ssh` command to connect securely from the remote host.

▼ How to Copy Files With Solaris Secure Shell

Use the `scp` command to copy encrypted files between hosts. You can copy encrypted files between either a local and remote host, or between two remote hosts. The command operates similarly to the `rcp` command except that the `scp` command prompts for passwords. See `scp`(1) for more information.

1. **Start the secure copy program.**

   Specify the source file, user name at remote destination, and destination directory.

   ```
   myLocalHost% scp myfile.1 johndoe@myRemoteHost:~
   ```

2. **Type the Solaris Secure Shell passphrase when prompted.**

   ```
   Enter passphrase for key '/home/johndoe/.ssh/id_rsa': <Return>
   myfile.1       25% |*******                     |    640 KB  0:20 ETA
   myfile.1
   ```

   After you type the passphrase, the progress meter is displayed. See the second line in the preceding output. The progress meter displays:

   - The file name
   - The percentage of the file that has been transferred at this point
   - A series of asterisks that are analogous to the percentage transferred
   - The quantity of data transferred
   - The estimated time of arrival, or ETA, of the complete file (that is, the remaining amount of time)

How to Transfer Files Remotely With the `sftp` Command

The `sftp` command works similarly to `ftp`, but uses a different set of subcommands. The following table lists some representative subcommands.

TABLE 11–3 Interactive `sftp` Subcommands

Category	Subcommands	Description
Navigation	`cd` *path,*	Changes the remote directory to *path*
	`lcd` *path*	Changes the local directory to *path*
Ownership	`chgrp` *group file*	Changes the group for *file* to *group*, a numeric GID
	`chmod` *mode file*	Changes the permissions of *file*

TABLE 11–3 Interactive `sftp` Subcommands *(Continued)*

Category	Subcommands	Description
File copying	`get` *remote_file* [*local-path*]	Retrieves a remote file and stores the file on the local host
	`put` *local_file* [*remote_path*]	Stores a local file on the remote host
	`rename` *old_filenew_file*	Renames a local file
Directory listing	`ls` [*path*]	Lists the contents of the remote directory
Directory creation	`mkdir` *path*	Creates a remote directory
Miscellaneous	`exit`, `quit`	Quits the `sftp` command

▼ How to Set Up Default Connections to Hosts Outside a Firewall

You can use Solaris Secure Shell to make a connection from a host inside a firewall to a host on the other side of the firewall. This task is done by specifying a proxy command for `ssh` either in a configuration file or as an option on the command line. For more information, see "Example—Connecting to Hosts Outside a Firewall From the Command Line" on page 196.

In general, you can customize your `ssh` interactions through a configuration file, either your own personal file `$HOME/.ssh/config` or an administrative configuration file in `/etc/ssh/ssh_config`. See `ssh_config`(4). There are two types of proxy commands. One proxy command is for HTTP connections. The other proxy command is for SOCKS5 connections.

1. **Specify the proxy commands and hosts in a configuration file.**

   Use the following syntax to add as many lines as you need:

   ```
   [Host outside_host]
   ProxyCommand proxy_command [-h proxy_server] \
   [-p proxy_port] outside_host|%h outside_port|%p
   ```

   where

   `Host` *outside_host* — Limits the proxy command specification to instances when a remote host name is specified on the command line. If you use a wildcard for *outside_host*, you apply the specification to a set of hosts.

   proxy_command — Specifies the proxy command. The command can be either of the following:

   - `/usr/lib/ssh/ssh-http-proxy-connect` for HTTP connections

	■ `/usr/lib/ssh/ssh-socks5-proxy-connect` for SOCKS5 connections
`-h` *proxy_server* and `-p` *proxy_port*	These options specify a proxy server and a proxy port, respectively. If present, the proxies override any environment variables that specify proxy servers and proxy ports, such as `HTTPPROXY`, `HTTPPROXYPORT`, `SOCKS5_PORT`, `SOCKS5_SERVER`, and `http_proxy`. The `http_proxy` variable specifies a URL. If the options are not used, then the relevant environment variables must be set. See the `ssh-socks5-proxy-connect`(1) and `ssh-http-proxy-connect`(1) man pages.
outside_host	Designates a specific host to connect to. You can use `%h` to specify the host on the command line.
outside_port	Designates a specific port to connect to. You can use `%p` to specify the port on the command line. By specifying `%h` and `%p` without using the `Host` *outside_host* option, the proxy command is applied to the host argument whenever the `ssh` command is invoked.

2. **Run Solaris Secure Shell, specifying the outside host.**

   For example, type the following:

   ```
   myLocalHost% ssh myOutsideHost
   ```

   This command looks for a proxy command specification for `myOutsideHost` in your personal configuration file. If the specification is not found, then the command looks in the system-wide configuration file, `ssh_config`. The proxy command is substituted for `ssh`.

Example—Connecting to Hosts Outside a Firewall From the Command Line

The `-o` option to the `ssh` command lets you type any line that is permitted in an `ssh` configuration file. In this case, the proxy command specification from the previous task is used.

1. Specify the proxy commands and hosts in a configuration file.
2. Run the `ssh` command. Include a proxy command specification as an argument to the `-o` option. For example, type the following:

   ```
   % ssh -o'Proxycommand=/usr/lib/ssh/ssh-http-proxy-connect \
   -h myProxyServer -p 8080 myOutsideHost 22' myOutsideHost
   ```

This command substitutes the HTTP proxy command for `ssh`, uses port `8080` and `myProxyServer` as the proxy server, and connects to port `22` on `myOutsideHost`.

CHAPTER 12

Solaris Secure Shell Administration (Reference)

This chapter describes how Solaris Secure Shell works from the administrator's point of view and how it is configured. The following is a list of the reference information in this chapter.

A Typical Solaris Secure Shell Session

The Solaris Secure Shell daemon (`sshd`) is normally started at boot from the `/etc/init.d/sshd` script. The daemon listens for connections from clients. ASolaris Secure Shell session begins when the user runs the `ssh`, `scp`, or `sftp` command. A new `sshd` daemon is forked for each incoming connection. The forked daemons handle key exchange, encryption, authentication, command execution, and data exchange with the client. These session characteristics are determined by client-side configuration files and server configuration files, and potentially command-line parameters. The client and server must authenticate themselves to each other. After successful authentication, the user can execute commands remotely and copy data between hosts.

Session Characteristics

The server-side behavior of the `sshd` daemon is controlled by keyword settings in the `/etc/ssh/sshd_config` file and potentially the command-line options when `sshd` is started. For example, `sshd_config` controls which types of authentication are permitted for accessing the server.

The behavior on the client side is controlled by the Solaris Secure Shell parameters in this order of precedence:

- Command-line options
- User's configuration file (`$HOME/.ssh/config`)
- System-wide configuration file (`/etc/ssh/ssh_config`)

For example, a user can override a system-wide configuration `Cipher` that is set to `blowfish` by specifying `-c 3des` on the command line.

Authentication

The steps in the authentication process for Solaris Secure Shell are as follows:

1. The user runs the `ssh`, `scp`, or `sftp` command.
2. The client and server agree on a shared session key.

   In v1, the remote host sends its host (RSA) key and a server (RSA) key to the client. Note that the server key is typically generated every hour and stored in memory only. The client checks that the remote host key is stored in the `$HOME/.ssh/known_hosts` file on the local host. The client then generates a 256 bit random number and encrypts it with the remote host's host key and server key. The encrypted random number is used as a session key to encrypt all further communications in the session.

   In v2, the remote host uses DSA in its host key and does not generate a server key. Instead, the shared session key is derived through a Diffie-Hellman agreement.
3. The local and remote hosts authenticate each other.

   In v1, the client can use `.rhosts`, `.rhosts` with RSA, RSA challenge-response, or password-based authentication. In v2, only `.rhosts`, DSA, and password-based authentication are permitted.

Command Execution and Data Forwarding

After authentication is complete, the user can use Solaris Secure Shell, generally by requesting a shell or executing a command. Through the `ssh` options, the user can make requests, such as allocating a pseudo-tty, forwarding X11 connections or TCP/IP connections, or enabling an `ssh-agent` over a secure connection. The basic components of a user session are as follows:

1. The user requests a shell or the execution of a command, which begins the session mode.

   In this mode, data is sent or received through the terminal on the client side, and the shell or command on the server side.
2. The user program terminates.

3. All X11 forwarding and TCP/IP forwarding is stopped. Any X11 connections and TCP/IP connections that already exist remain open.
4. The server sends the command exit to the client, and both sides exit.

Configuring the Solaris Secure Shell

The characteristics of a Solaris Secure Shell session are controlled by configuration files, which can be overridden to a certain degree by options on the command line.

Solaris Secure Shell Client Configuration

In most cases, the client-side characteristics of a Solaris Secure Shell session are governed by the system-wide configuration file, `/etc/ssh/ssh_config`, which is set up by the administrator. The settings in the system-wide configuration file can be overridden by the user's configuration in `$HOME/.ssh/config`. In addition, the user can override both configuration files on the command line.

The command line options are client requests and are permitted or denied on the server side by the `/etc/ssh/sshd_config` file (see `ssh_config`(4)). The configuration file keywords and command options are introduced in the following sections and are described in detail in the `ssh`(1), `scp`(1), `sftp`(1), and `ssh_config`(4) man pages. Note that in the two user configuration files, the `Host` keyword indicates a host or wildcard expression to which all following keywords up to the next `Host` keyword apply.

Host-Specific Parameters

If it is useful to have different Solaris Secure Shell characteristics for different local hosts, the administrator can define separate sets of parameters in the `/etc/ssh/ssh_config` file to be applied according to host or regular expression. This task is done by grouping entries in the file by `Host` keyword. If the `Host` keyword is not used, the entries in the client configuration file apply to whichever local host a user is working on.

Client-Side Authentication Parameters

The authentication method is determined by setting one of the following keywords to "yes":

- `DSAAuthentication`

- PasswordAuthentication
- RhostsAuthentication
- RhostsRSAAuthentication

The keyword UseRsh specifies that the rlogin and rsh commands be used, probably due to no Secure Shell support.

The Protocol keyword sets the Solaris Secure Shell protocol version to v1 or v2. You can specify both versions separated by a comma. The first version is tried and upon failure, the second version is used.

The IdentityFile keyword specifies an alternate file name to hold the user's private key.

The keyword Cipher specifies the v1 encryption algorithm, which might be blowfish or 3des. The keyword Ciphers specifies an order of preference for the v2 encryption algorithms: 3des-cbc, blowfish-cbc, and aes128-cbc. The commands ssh and scp have a -c option for specifying the encryption algorithm on the command line.

Known Host File Parameters

The known host files (/etc/ssh/ssh_known_hosts and $HOME/.ssh/known_hosts) contain the public keys for all hosts with which the client can communicate by using Solaris Secure Shell. The GlobalKnownHostsFile keyword specifies an alternate file instead of /etc/ssh/ssh_known_hosts. The UserKnownHostsFile keyword specifies an alternate to $HOME/.ssh/known_hosts.

The StrictHostKeyChecking keyword requires new hosts to be added manually to the known hosts file, and refuses any host whose public key has changed or whose public key is not in the known hosts file. The keyword CheckHostIP enables the IP address for hosts in the known host files to be checked, in case a key has been changed due to DNS spoofing.

Client-Side X11 Forwarding and Port Forwarding Parameters

The LocalForward keyword specifies a local TCP/IP port to be forwarded over a secure channel to a specified port on a remote host. The GatewayPorts keyword enables remote hosts to connect to local forwarded ports.

The command ssh enables port forwarding through these options:

- -L, which specifies the local port to be forwarded to the specified port on the remote host
- -R, which specifies a remote port to be forwarded to the local host and specified port

The ForwardX11 keyword redirects X11 connections to the remote host with the DISPLAY environment variable set. The XAuthLocation keyword specifies the location of the xauth program.

Client-Side Connection and Other Parameters

The NumberOfPasswordPrompts keyword specifies how many times the user is prompted for a password before Solaris Secure Shell quits. The ConnectionAttempts keyword specifies how many tries (at one try per second) are made before Solaris Secure Shell either quits or falls back to rsh if the FallBackToRsh keyword is set.

The Compression keyword enables compression of transmitted data. The CompressionLevel keyword sets a level of 1 to 9, trading off between speed and amount of compression.

User specifies an alternate user name. Hostname specifies an alternate name for a remote host. ProxyCommand specifies an alternate command name for starting Solaris Secure Shell. Any command that can connect to your proxy server can be used. The command should read from its standard input and write to its standard output.

Batchmode disables password prompts, which is useful for scripts and other batch jobs.

KeepAlive enables messages to indicate network problems due to host crashes. LogLevel sets the verbosity level for ssh messages.

EscapeChar defines a single character that is used as a prefix for displaying special characters as plain text.

Solaris Secure Shell Server Configuration

The server-side characteristics of a Solaris Secure Shell session are governed by the /etc/ssh/sshd_config file, which is set up by the administrator.

Server-Side Authentication Parameters

Permitted authentication methods are indicated by theses keywords:

- DSAAuthentication
- PasswordAuthentication
- RhostsAuthentication
- RhostsRSAAuthentication
- RSAAuthentication

HostKey and HostDSAKey identify files that hold host public keys when the default file name is not used. KeyRegenerationInterval defines how often the server key is regenerated.

`Protocol` specifies the version. `Ciphers` specifies the encryption algorithms for v2. `ServerKeyBits` defines the number of bits in the server's key.

Ports and Forwarding Parameters

`AllowTCPForwarding` specifies whether TCP forwarding is permitted.

`GatewayPorts` allows remote hosts to connect to ports forwarded for the client. `Port` specifies the port number that `sshd` listens on. `ListenAddress` designates a specific local address that `sshd` listens to. If there is no `ListenAddress` specification, `sshd` listens to all addresses by default.

`X11Forwarding` allows X11 forwarding. `X11DisplayOffset` specifies the first display number that is available for forwarding. This keyword prevents `sshd` from interfering with real X11 servers. `XAuthLocation` specifies the location of the `xauth` program.

Session Control Parameters

`KeepAlive` displays messages regarding broken connections and host crashes. `LogLevel` sets the verbosity level of messages from `sshd`. `SyslogFacility` provides a facility code for messages that are logged from `sshd`.

Server Connection and Other Parameters

The `AllowGroups`, `AllowUsers`, `DenyGroups`, and `DenyUsers` keywords control which users can or cannot use `ssh`.

The `LoginGraceTime`, `MaxStartups`, `PermitRootLogin`, and `PermitEmptyPasswords` keywords set controls on users who are logging in. `StrictModes` causes `sshd` to check file modes and ownership of the user's files and home directory before login. `UseLogin` specifies whether `login` is used for interactive login sessions. Turning this keyword on should not be necessary and is not recommended for the Solaris environment.

`Subsystem` configures a file transfer daemon for using `sftp`.

Maintaining Known Hosts on a Site-Wide Basis

Each host that needs to talk to another host securely must have the server's public key stored in the local host's `/etc/ssh/ssh_known_hosts` file. Although it is most convenient to update the `/etc/ssh/ssh_known_hosts` files by a script, this practice is heavily discouraged because it opens a major security vulnerability.

The `/etc/ssh/ssh_known_hosts` file should only be distributed by a secure mechanism as follows:

- Over a secure connection such as Solaris Secure Shell, IPsec, or kerberized `ftp` from a known and trusted machine
- At system install time

To avoid the possibility of an intruder gaining access by inserting bogus public keys into a `known_hosts` file, you should use the jumpstart server as the known and trusted source of the `ssh_known_hosts` file. The `ssh_known_hosts` file can be distributed during installation and by regularly running scripts on the individual hosts that pull in the latest version by using `scp`. This approach is secure because each host already has the public key of the jumpstart server.

Solaris Secure Shell Files

The following table shows the important Solaris Secure Shell files and the suggested UNIX permissions.

TABLE 12–1 Solaris Secure Shell Files

File Name	Description	Suggested Permissions and Owner
`/etc/ssh/sshd_config`	Contains configuration data for `sshd`, the Secure Shell daemon.	`-rw-r--r-- root`
`/etc/ssh/ssh_host_key`	Contains the host private key.	`-rw------- root`
`/etc/ssh_host_key.pub`	Contains the host public key. Used to copy the host key to the local `known_hosts` file.	`-rw-r--r-- root`

TABLE 12–1 Solaris Secure Shell Files *(Continued)*

File Name	Description	Suggested Permissions and Owner
`/var/run/sshd.pid`	Contains the process ID of the Secure Shell daemon, `sshd`, which listens for connections (if there are multiple daemons, the file contains the last daemon that was started).	`rw-r--r-- root`
`$HOME/.ssh/authorized_keys`	Lists the RSA keys that can be used with v1 to log into the user's account, or the DSA and RSA keys that can be used with v2.	`-rw-rw-r-- johndoe`
`/etc/ssh/ssh_known_hosts`	Contains the host public keys for all hosts with which the client may communicate securely. The file should be prepared by the administrator.	`-rw-r--r-- root`
`$HOME/.ssh/known_hosts`	Contains the host public keys for all hosts with which the client may communicate securely. The file is maintained automatically. Whenever the user connects with an unknown host, the remote host key is added to the file.	`-rw-r--r-- johndoe`
`/etc/nologin`	If this file exists, `sshd` refuses to let anyone except `root` log in. The contents are displayed to users who are attempting to log in.	`-rw-r--r-- root`
`$HOME/.rhosts`	Contains the host-user name pairs that specifies the hosts to which the user can log in to without a password. The file is used Secure Shell, as well as by the `rlogind` and `rshd` daemons.	`-rw-r—r-- johndoe`
`$HOME/.shosts`	Contains the host-user name pairs that specifies the hosts to which the user can log in to without a password using Secure Shell only.	`-rw-r—r-- johndoe`
`/etc/hosts.equiv`	Contains the hosts that are used in `.rhosts` authentication and Secure Shell authentication.	`-rw-r--r-- root`
`/etc/ssh/shosts.equiv`	Contains the hosts that are used in Secure Shell authentication.	`-rw-r--r-- root`
`$HOME/.ssh/environment`	Used for initialization to make assignments at login.	`-rw------- johndoe`

TABLE 12–1 Solaris Secure Shell Files *(Continued)*

File Name	Description	Suggested Permissions and Owner
`$HOME/.ssh/rc`	Runs initialization routines before the user shell starts.	`-rw------- johndoe`
`/etc/ssh/sshrc`	Runs host-specific initialization routines that are specified by an administrator for all users.	`-rw-r--r-- root`

The following table summarizes the major Solaris Secure Shell commands.

TABLE 12–2 Solaris Secure Shell Commands

Command	Description
`ssh`	A program for logging in to a remote machine and for executing commands on a remote machine. The command is intended to replace `rlogin` and `rsh`, and provide secure encrypted communications between two untrusted hosts over an insecure network. X11 connections and arbitrary TCP/IP ports can also be forwarded over the secure channel.
`sshd`	The daemon for Secure listens. This daemon listens for connections from clients and provides secure encrypted communications between two untrusted hosts over an insecure network.
`ssh-keygen`	Generates and manages authentication keys for `ssh`.
`ssh-agent`	A program that holds private keys that are used for public key authentication. `ssh-agent` is started at the beginning of an X-session or a login session, and all other windows or programs are started as clients to the `ssh-agent` program. Through the use of environment variables, the agent can be located and automatically used for authentication when users log in to other machines while using `ssh`.
ssh-add	Adds RSA or DSA identities (keys) to the authentication agent, `ssh-agent`.
scp	Securely copies files between hosts on a network by using `ssh` for data transfer. Unlike `rcp`, `scp` asks for passwords or passphrases (if they are needed for authentication).
sftp	An interactive file transfer program, similar to `ftp`, that performs all operations over an encrypted `ssh` transport. `sftp` connects and logs into the specified host name and then enters an interactive command mode.

CHAPTER 13

Introduction to SEAM

This chapter provides an introduction to Sun Enterprise Authentication Mechanism (SEAM).

What Is SEAM?

SEAM is a client/server architecture that provides secure transactions over networks. SEAM offers strong user authentication, as well as data integrity and data privacy. *Authentication* guarantees that the identities of both the sender and the recipient of a network transaction are true. SEAM can also verify the validity of data being passed back and forth (*integrity*) and encrypt the data during transmission (*privacy*). Using SEAM, you can log on to other machines, execute commands, exchange data, and transfer files securely. Additionally, SEAM provides *authorization* services, which allows administrators to restrict access to services and machines. Moreover, as a SEAM user, you can regulate other people's access to your account.

SEAM is a *single-sign-on* system, which means that you only need to authenticate yourself to SEAM once per session, and all subsequent transactions during the session are automatically secured. After SEAM has authenticated you, you do not need to authenticate yourself every time you use a SEAM-based command such as `ftp` or `rsh`, or access data on an NFS file system. Thus, you do not have to send your password over the network, where it can be intercepted, each time you use these services.

SEAM is based on the Kerberos V5 network authentication protocol that was developed at the Massachusetts Institute of Technology (MIT). People who have used Kerberos V5 should therefore find SEAM very familiar. Since Kerberos V5 is a *de facto* industry standard for network security, SEAM promotes interoperability with other systems. In other words, because SEAM works with systems that use Kerberos V5, it allows for secure transactions even over heterogeneous networks. Moreover, SEAM provides authentication and security both between domains and within a single domain.

Note – Because SEAM is based on, and designed to interoperate with, Kerberos V5, this manual often uses the terms "Kerberos" and "SEAM" more or less interchangeably, for example, "Kerberos realm" or "SEAM-based utility." Moreover, "Kerberos" and "Kerberos V5" are used interchangeably. The manual draws distinctions when necessary.

SEAM allows for flexibility in running Solaris applications. You can configure SEAM to allow both SEAM-based and non-SEAM-based requests for network services such as the NFS service, `telnet`, and `ftp`. As a result, current Solaris applications still work even if they are running on systems on which SEAM is not installed. Of course, you can also configure SEAM to allow only SEAM-based network requests.

Additionally, applications do not have to remain committed to SEAM if other security mechanisms are developed. Because SEAM is designed to integrate modularly into the Generic Security Service (GSS) API, applications that make use of the GSS-API can utilize whichever security mechanism best suits its needs.

How SEAM Works

The following is an overview of the SEAM authentication system. For a more detailed description, see "How the Authentication System Works" on page 327.

From the user's standpoint, SEAM is mostly invisible after the SEAM session has been started. Commands such as `rsh` or `ftp` work pretty much in their usual fashion. Initializing a SEAM session often involves no more than logging in and providing a Kerberos password.

The SEAM system revolves around the concept of a *ticket*. A ticket is a set of electronic information that serves as identification for a user or a service such as the NFS service. Just as your driver's license identifies you and indicates what driving permissions you have, so a ticket identifies you and your network access privileges. When you perform a SEAM-based transaction (for example, if you `rlogin` in to another machine), you transparently send a request for a ticket to a *Key Distribution Center*, or KDC. The KDC

accesses a database to authenticate your identity and returns a ticket that grants you permission to access the other machine. "Transparently" means that you do not need to explicitly request a ticket; it happens as part of the `rlogin` command. Because only the authenticated client can get a ticket for a specific service, another client cannot use `rlogin` under an assumed identity.

Tickets have certain attributes associated with them. For example, a ticket can be *forwardable* (which means that it can be used on another machine without a new authentication process), or *postdated* (not valid until a specified time). How tickets are used (for example, which users are allowed to obtain which types of ticket) is set by *policies* that are determined when SEAM is installed or administered.

Note – You will frequently see the terms *credential* and *ticket*. In the greater Kerberos world, they are often used interchangeably. Technically, however, a credential is a ticket plus the *session key* for that session. This difference is explained in more detail in "Gaining Access to a Service Using SEAM" on page 328.

The following sections further explain the SEAM authentication process.

Initial Authentication: the Ticket-Granting Ticket

Kerberos authentication has two phases: an initial authentication that allows for all subsequent authentications, and the subsequent authentications themselves.

The following figure shows how the initial authentication takes place.

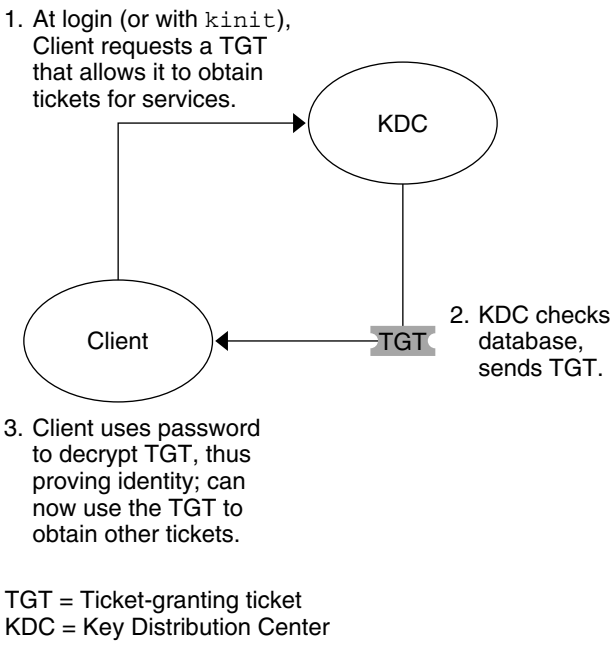


FIGURE 13–1 Initial Authentication for SEAM Session

1. A client (a user, or a service such as NFS) begins a SEAM session by requesting a *ticket-granting ticket* (TGT) from the Key Distribution Center (KDC). This request is often done automatically at login.

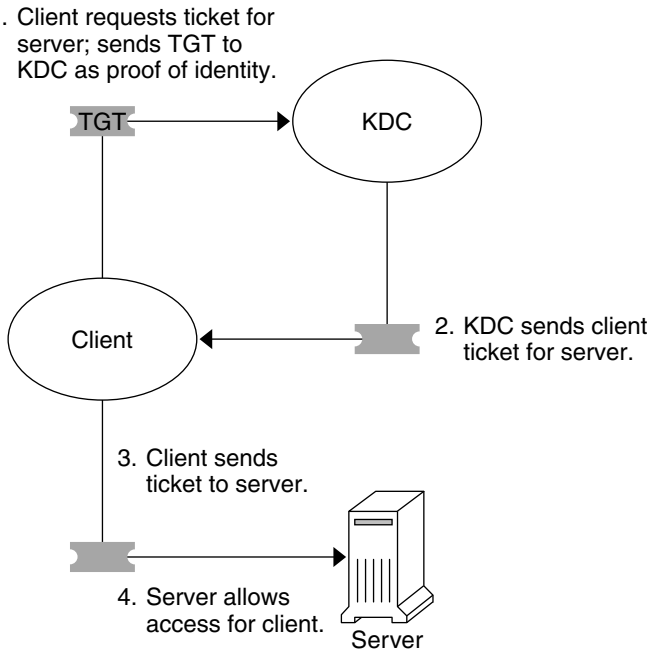
   A ticket-granting ticket is needed to obtain other tickets for specific services. Think of the ticket-granting ticket as similar to a passport. Like a passport, the ticket-granting ticket identifies you and allows you to obtain numerous "visas," where the "visas" (tickets) are not for foreign countries but for remote machines or network services. Like passports and visas, the ticket-granting ticket and the other various tickets have limited lifetimes. The difference is that "Kerberized" commands notice that you have a passport and obtain the visas for you. You don't have to perform the transactions yourself.

   Another analogy for the ticket-granting ticket is that of a three-day ski pass which is good at four different ski resorts. You show the pass at whichever resort you decide to go to (until it expires) and you receive a lift ticket for that resort. Once you have the lift ticket, you can ski all you want at that resort. If you go to another resort the next day, you once again show your pass, and you get an additional lift ticket for the new resort. The difference is that the SEAM-based commands notice that you have the weekend ski pass, and get the lift ticket for you, so you don't have to perform the transactions yourself.
2. The KDC creates a ticket–granting ticket and sends it back, in encrypted form, to the client. The client decrypts the ticket-granting ticket by using the client's password.

3. Now in possession of a valid ticket-granting ticket, the client can request tickets for all sorts of network operations, such as `rlogin` or `telnet`, for as long as the ticket-granting ticket lasts. This ticket usually lasts for a few hours. Each time the client performs a unique network operation, it requests a ticket for that operation from the KDC.

Subsequent Authentications

After the client has received the initial authentication, each individual authentication follows the pattern that is shown in the following figure.



TGT = Ticket-granting ticket
KDC = Key Distribution Center

FIGURE 13–2 Obtaining Access to a Service

1. The client requests a ticket for a particular service (say, to `rlogin` into another machine) from the KDC by sending the KDC its ticket-granting ticket as proof of identity.
2. The KDC sends the ticket for the specific service to the client.

   For example, suppose user `joe` wants to access an NFS file system that has been shared with krb5 authentication required. Since he is already authenticated (that is, he already has a ticket-granting ticket), as he attempts to access the files, the NFS

client system automatically and transparently obtains a ticket from the KDC for the NFS service.

For example, suppose the user `joe` uses `rlogin` on the server `boston`. Since he is already authenticated (that is, he already has a ticket-granting ticket), he automatically and transparently obtains a ticket as part of the `rlogin` command. This ticket allows him to `rlogin` into `boston` as often as he wants until it expires. If `joe` wants to `rlogin` into the machine `denver`, he obtains another ticket, as in Step 1.

3. The client sends the ticket to the server.

   When using the NFS service, the NFS client automatically and transparently sends the ticket for the NFS service to the NFS server.

4. The server allows the client access.

These steps make it appear that the server doesn't ever communicate with the KDC. The server does, though; it registers itself with the KDC, just as the first client does. For simplicity's sake, we have left that part out.

The SEAM Remote Applications

What are the SEAM-based (or "Kerberized") commands that a user such as `joe` can use? They are:

- `ftp`
- `rcp`
- `rlogin`
- `rsh`
- `telnet`

These applications are the same as the Solaris applications of the same name, except that they use Kerberos principals to authenticate transactions, thereby providing Kerberos-based security. (See "Principals" on page 214 for information on principals.)

Principals

A client in SEAM is identified by its *principal*. A principal is a unique identity to which the KDC can assign tickets. A principal can be a user, such as `joe`, or a service, such as `nfs` or `telnet`.

By convention, a principal name is divided into three parts: the *primary*, the *instance*, and the *realm*. A typical SEAM principal would be, for example, `joe/admin@ENG.EXAMPLE.COM`, where:

- `joe` is the primary. The primary can be a user name, as shown here, or a service, such as `nfs`. The primary can also be the word `host`, which signifies that this principal is a service principal that is set up to provide various network services (`ftp`, `rcp`, `rlogin`, and so on).

- `admin` is the instance. An instance is optional in the case of user principals, but it is required for service principals. For example: if the user `joe` sometimes acts as a system administrator, he can use `joe/admin` to distinguish himself from his usual user identity. Likewise, if `joe` has accounts on two different hosts, he can use two principal names with different instances (for example, `joe/denver.example.com` and `joe/boston.example.com`). Notice that SEAM treats `joe` and `joe/admin` as two completely different principals.

  In the case of a service principal, the instance is the fully qualified host name. `bigmachine.eng.example.com` is an example of such an instance so that the primary/instance might be, for example, `ftp/bigmachine.eng.example.com` or `host/bigmachine.eng.example.com`.
- `ENG.EXAMPLE.COM` is the SEAM realm. Realms are discussed in "Realms" on page 215.

The following are all valid principal names:

- `joe`
- `joe/admin`
- `joe/admin@ENG.EXAMPLE.COM`
- `ftp/host.eng.example.com@ENG.EXAMPLE.COM`
- `host/eng.example.com@ENG.EXAMPLE.COM`

Realms

A realm is a logical network, similar to a domain, which defines a group of systems under the same *master KDC*. Figure 13–3 shows how realms can relate to one another. Some realms are hierarchical (one realm being a superset of the other realm). Otherwise, the realms are non-hierarchical (or "direct") and the mapping between the two realms must be defined. A feature of SEAM is that it permits authentication across realms. Each realm only needs to have a principal entry for the other realm in its KDC. The feature is called cross-realm authentication.

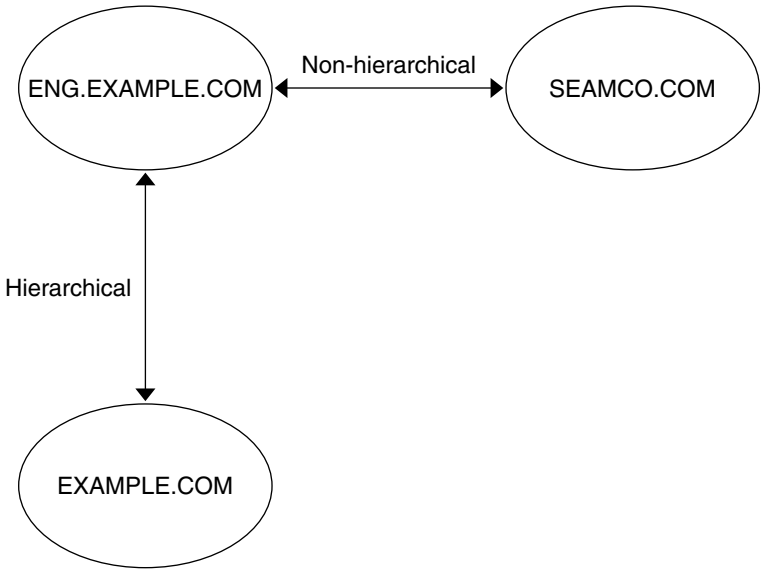


FIGURE 13–3 Realms

Realms and Servers

Each realm must include a server that maintains the master copy of the principal database. This server is called the *master KDC server*. Additionally, each realm should contain at least one *slave KDC server*, which contains duplicate copies of the principal database. Both the master KDC server and the slave KDC server create tickets that are used to establish authentication.

The realm can also include two additional types of SEAM servers. A SEAM network *application server* is a server that provides access to Kerberized applications (such as `ftp`, `telnet` and `rsh`). Realms can also include *NFS servers*, which provide NFS services by using Kerberos authentication. If you have installed SEAM 1.0 or 1.0.1, the realm might include a SEAM network *application server*, which provides access to Kerberized applications (such as `ftp`, `telnet`, and `rsh`).

The following figure shows what a hypothetical realm might contain.

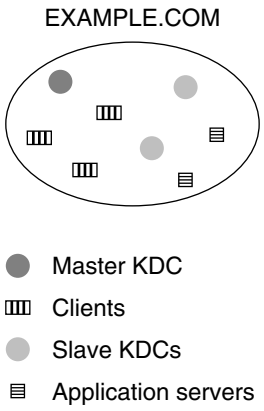


FIGURE 13–4 A Typical Realm

SEAM Security Services

In addition to providing secure authentication of users, SEAM provides two security services:

- **Integrity –** Just as authentication ensures that clients on a network are who they claim to be, integrity ensures that the data they send is valid and has not been tampered with during transit. Integrity is done through cryptographic checksumming of the data. Integrity also includes user authentication.
- **Privacy –** Privacy takes security a step further. Privacy not only includes verifying the integrity of transmitted data, but it encrypts the data before transmission, protecting it from eavesdroppers. Privacy authenticates users, as well.

Currently, of the various Kerberized applications which are part of SEAM, only the `ftp` command allows users to change security service at runtime ("on the fly"). Developers can design their RPC-based applications to choose a security service by using the RPCSEC_GSS programming interface.

SEAM Releases

Components of the SEAM product have been included in four releases. The following table describes which components are included in each release. All components are described in the following sections.

TABLE 13–1 SEAM Release Contents

Release Name	Contents
SEAM 1.0 in Solaris Easy Access Server (SEAS) 3.0	Full release of SEAM for the Solaris 2.6 and 7 releases
SEAM in the Solaris 8 release	SEAM client software only
SEAM 1.0.1 in the Solaris 8 Admin Pack	SEAM KDC and remote applications for the Solaris 8 release
SEAM in the Solaris 9 release	SEAM KDC and client software only
SEAM 1.0.2	SEAM remote applications for the Solaris 9 release

SEAM 1.0 Components

Similar to the MIT distribution of Kerberos V5, SEAM includes the following:

- Key Distribution Center (KDC) (master):
  - Kerberos database administration daemon – `kadmind`
  - Kerberos ticket processing daemon – `krb5kdc`
- Slave KDCs
- Database administration programs – `kadmin` and `kadmin.local`
- Database propagation software – `kprop`
- User programs for obtaining, viewing, and destroying tickets – `kinit`, `klist`, `kdestroy` – and for changing your SEAM password – `kpasswd`
- Applications – `ftp`, `rcp`, `rlogin`, `rsh`, and `telnet` – and daemons for these applications – `ftpd`, `rlogind`, `rshd` and `telnetd`
- Administration utilities – `ktutil`, `kdb5_util`
- Several libraries

In addition, SEAM includes the following:

- SEAM Administration Tool (`gkadmin`) – Allows you to administer the KDC. This Java™ technology-based GUI allows an administrator to perform the tasks that are usually performed through the `kadmin` command.
- The Pluggable Authentication Module (PAM) – Allows applications to use various authentication mechanisms. PAM can be used to make login and logouts transparent to the user.
- A utility (`gsscred`) and a daemon (`gssd`) – These programs help map UNIX user IDs (UIDs) to principal names. These programs are needed because SEAM NFS servers use UNIX UIDs to identify users and not principal names, which are stored in a different format.

- The Generic Security Service Application Programming Interface (GSS-API) – Allows applications to use multiple security mechanisms without having to recompile the application every time a new mechanism is added. Because GSS-API is machine-independent, it is appropriate for applications on the Internet. GSS-API provides applications with the ability to include the integrity and privacy security services, as well as authentication.
- The RPCSEC_GSS Application Programming Interface (API) – Allows NFS services to use Kerberos authentication. RPCSEC_GSS is a new security flavor that provides security services that are independent of the mechanisms being used. RPCSEC_GSS sits "on top" of the GSS-API layer. Any pluggable GSS_API-based security mechanism can be used by applications that use RPCSEC_GSS.
- A preconfiguration procedure – Allows you to set the parameters for installing and configuring SEAM, which make SEAM installation automatic. This procedure is especially useful for multiple installations.
- Kernel modifications – Allows for faster performance.

SEAM Components in the Solaris 8 Release

The Solaris 8 release included only the client-side portions of SEAM, so many components are not included. This product enables systems that run the Solaris 8 release to become SEAM clients without having to install SEAM separately. To use these capabilities, you must install a KDC that uses either SEAS 3.0 or the Solaris 8 Admin Pack, the MIT distribution, or Windows2000. The client-side components are not useful without a configured KDC to distribute tickets. The following components were included in this release:

- User programs for obtaining, viewing, and destroying tickets – `kinit`, `klist`, `kdestroy`.
- User program for changing your SEAM password – `kpasswd`.
- Key table administration utility – `ktutil`.
- Additions to the Pluggable Authentication Module (PAM) – Allows applications to use various authentication mechanisms. PAM can be used to make login and logouts transparent to the user.
- GSS_API plug–ins – Provides Kerberos protocol and cryptographic support.
- NFS client and server support.

SEAM 1.0.1 Components

The SEAM 1.0.1 release includes all components of the SEAM 1.0 release that are not already included in the Solaris 8 release. The components are as follows:

- Key Distribution Center (KDC) (master):

    - Kerberos database administration daemon – `kadmind`.
    - Kerberos ticket processing daemon – `krb5kdc`.
- Slave KDCs.
- Database administration programs – `kadmin` and `kadmin.local`.
- Database propagation software – `kprop`
- Applications – `ftp`, `rcp`, `rlogin`, `rsh`, and `telnet` – and daemons for these applications – `ftpd`, `rlogind`, `rshd` and `telnetd`.
- Administration utility – `kdb5_util`.
- SEAM Administration Tool (`gkadmin`) – Allows you to administer the KDC. This Java technology-based GUI allows an administrator to perform the tasks that are usually performed through the `kadmin` command.
- A preconfiguration procedure – Allows you to set the parameters for installing and configuring SEAM, which makes SEAM installation automatic. This procedure is especially useful for multiple installations.
- Several libraries.

SEAM Components in the Solaris 9 Release

The Solaris 9 release includes all components of the SEAM 1.0 release, except for the remote applications and the preconfiguration procedure.

SEAM 1.0.2 Components

The SEAM 1.0.2 release includes the remote applications. These applications are the only part of SEAM 1.0 that have not been incorporated into the Solaris 9 release. The components for the remote applications are as follows:

- Client applications – `ftp`, `rcp`, `rlogin`, `rsh`, and `telnet`.
- Server Daemons – `ftpd`, `rlogind`, `rshd` and `telnetd`.

CHAPTER 14

Planning for SEAM

This chapter should be studied by administrators who are involved in the installation and maintenance of SEAM. The chapter discusses several installation and configuration issues that administrators must resolve before they install or configure SEAM.

This is a list of the issues that a system administrator or other knowledgeable support staff should resolve:

Why Plan for SEAM?

Before you install SEAM, you must resolve several configuration issues. Although changing the configuration after the initial install is not impossible, it becomes more difficult with each new client that is added to the system. In addition, some changes require a full re-installation, so it is better to consider long-term goals when you planning your SEAM configuration.

Realms

A realm is logical network, similar to a domain, which defines a group of systems that are under the same master KDC. As with establishing a DNS domain name, issues such as the realm name, the number and size of each realm, and the relationship of a realm to other realms for cross-realm authentication should be resolved before you configure SEAM.

Realm Names

Realm names can consist of any ASCII string. Usually, the realm name is the same as your DNS domain name, in uppercase. This convention helps differentiate problems with SEAM from problems with the DNS namespace, while using a name that is familiar. If you do not use DNS or you choose to use a different string, then you can use any string. However, the configuration process requires more work. The use of realm names that follow the standard Internet naming structure is wise.

Number of Realms

The number of realms that your installation requires depends on several factors:

- The number of clients to be supported. Too many clients in one realm makes administration more difficult and eventually requires that you split the realm. The primary factors that determine the number of clients that can be supported are as follows:
  - The amount of SEAM traffic that each client generates
  - The bandwidth of the physical network
  - The speed of the hosts

  Since each installation will have different limitations, there is no rule for determining the maximum number of clients.
- How far apart the clients are. It might make sense to set up several small realms if the clients are in different geographic regions.
- The number of hosts that are available to be installed as KDCs. Each realm should have at least two KDC servers (master and slave).

Realm Hierarchy

When you are configuring multiple realms for cross-realm authentication, you need to decide how to tie the realms together. You can establish a hierarchical relationship between the realms that provides automatic paths to the related domains. Of course,

all realms in the hierarchical chain must be configured properly. The automatic paths can ease the administration burden. However, if there are many levels of domains, you might not want to use the default path because it requires too many transactions.

You can also choose to establish the connection directly. A direct connection is most useful when too many levels exist between two hierarchical domains or when there is no hierarchal relationship. The connection must be defined in the `/etc/krb5/krb5.conf` file on all hosts that use the connection. So, some additional work is required. For an introduction, see "Realms" on page 215 and for the configuration procedures for multiple realms, see "Configuring Cross-Realm Authentication" on page 236.

Mapping Host Names Onto Realms

The mapping of host names onto realm names is defined in the `domain_realm` section of the `krb5.conf` file. These mappings can be defined for a whole domain and for individual hosts, depending on the requirements. See the `krb5.conf(4)` man page for more information.

Client and Service Principal Names

When you are using SEAM, it is strongly recommended that DNS services already be configured and running on all hosts. If DNS is used, it must be enabled on all systems or on none of them. If DNS is available, then the principal should contain the Fully Qualified Domain Name (FQDN) of each host. For example, if the host name is `boston`, the DNS domain name is `example.com`, and the realm name is `EXAMPLE.COM`, then the principal name for the host should be `host/boston.example.com@EXAMPLE.COM`. The examples in this book use the FQDN for each host.

For the principal names that include the FQDN of an host, it is important to match the string that describes the DNS domain name in the `/etc/resolv.conf` file. SEAM requires that the DNS domain name be in lowercase letters when you are entering the FQDN for a principal. The DNS domain name can include uppercase and lowercase letters, but only use lowercase letters when you are creating a host principal. For example, it doesn't matter if the DNS domain name is `example.com`, `Example.COM`, or any other variation. The principal name for the host would still be `host/boston.example.com@EXAMPLE.COM`.

SEAM can run without DNS services, but some key capabilities, such as the ability to communicate with other realms, will not work. If DNS is not configured, then a simple host name can be used as the instance name. In this case, the principal would be `host/boston@EXAMPLE.COM`. If DNS is enabled later, all host principals must be deleted and replaced in the KDC database.

Ports for the KDC and Admin Services

By default, port `88` and port `750` are used for the KDC, and port `749` is used for the KDC administration daemon. Different port numbers can be used. However, if you change the port numbers, then the `/etc/services` and `/etc/krb5/krb5.conf` files must be changed on every client. In addition, the `/etc/krb5/kdc.conf` file on each KDC must be updated.

Slave KDCs

Slave KDCs generate credentials for clients just as the master KDC does. The slave KDCs provide backup if the master becomes unavailable. Each realm should have at least one slave KDC. Additional slave KDCs might required, depending on these factors:

- The number of physical segments in the realm. Normally, the network should be set up so that each segment can function, at least minimally, without the rest of the realm. To do so, a KDC must be accessible from each segment. The KDC in this instance could be either a master or a slave.
- The number of clients in the realm. By adding more slave KDC servers, you can reduce the load on the current servers.

It is possible to add too many slave KDCs. Remember that the KDC database must be propagated to each server, so the more KDC servers that are installed, the longer it can take to get the data updated throughout the realm. Also, since each slave retains a copy of the KDC database, more slaves increase the risk of a security breach.

In addition, one or more slave KDCs can easily be configured to be swapped with the master KDC. The advantage to following this procedure on at least one slave KDC is that if the master KDC fails for any reason, you will have a system preconfigured that will be easy to swap as the master KDC. For instructions on how to configure a swappable slave KDC, see “Swapping a Master KDC and a Slave KDC” on page 249.

Database Propagation

The database that is stored on the master KDC must be regularly propagated to the slave KDCs. One of the first issues to resolve is how often to update the slave KDCs. The desire to have up-to-date information that is available to all clients needs to be weighed against the amount of time it takes to complete the update. For more information about database propagation, see "Administering the Kerberos Database" on page 252.

In large installations with many KDCs in one realm, it is possible for one or more slaves to propagate the data so that the process is done in parallel. This strategy reduces the amount of time that the update takes, but it also increases the level of complexity in administering the realm.

Clock Synchronization

All hosts that participate in the Kerberos authentication system must have their internal clocks synchronized within a specified maximum amount of time. Known as *clock skew*, this feature provides another Kerberos security check. If the clock skew is exceeded between any of the participating hosts, requests are rejected.

One way to synchronize all the clocks is to use the Network Time Protocol (NTP) software. See "Synchronizing Clocks between KDCs and SEAM Clients" on page 247 for more information. Other ways of synchronizing the clocks are available, so the use of NTP is not required. However, some form of synchronization should be used to prevent access failures because of clock skew.

Online Help URL

The online help URL is used by the SEAM Administration Tool, so the URL should be defined properly to enable the "Help Contents" menu to work. The HTML version of this manual can be installed on any appropriate server. Alternately, you can decide to use the collections at `http://docs.sun.com`.

The URL should point to the section titled "SEAM Administration Tool" in the "Administering Principals and Policies" chapter in this book. You can choose another HTML page, if another location is more appropriate.

CHAPTER 15

Configuring SEAM (Tasks)

This chapter provides configuration and installation procedures network application servers.

Configuring SEAM (Task Map)

Parts of the configuration process depend on other parts and must be done in a specific order. These procedures often establish services that are required to use SEAM. Other procedures are not dependent on any order, and can be done when appropriate. The following task map shows a suggested order for a SEAM installation.

TABLE 15–1 First Steps: SEAM Configuration Order

Task	Description	For Instructions
1. Plan for your SEAM installation	Lets you resolve configuration issues before you start the software configuration process. Planning ahead saves you time and other resources in the long run.	Chapter 14

TABLE 15–1 First Steps: SEAM Configuration Order *(Continued)*

Task	Description	For Instructions
2. (Optional) Install NTP	Configures the Network Time Protocol (NTP) software, or another clock synchronization protocol. In order for SEAM to work properly, the clocks on all systems in the realm must be synchronized.	"Synchronizing Clocks between KDCs and SEAM Clients" on page 247
3. Configure the master KDC server	Configures and builds the master KDC server and database for a realm.	"How to Configure a Master KDC" on page 229
4. (Optional) Configure a slave KDC server	Configures and builds a slave KDC server for a realm.	"How to Configure a Slave KDC" on page 233
5. (Optional) Increase security on the KDC servers	Prevents security breaches on the KDC servers.	"How to Restrict Access to KDC Servers" on page 258
6. (Optional) Configure swappable KDC servers	Makes the task of swapping the master KDC and a slave KDC easier.	"How to Configure a Swappable Slave KDC" on page 249

Once the required steps have been completed, the following procedures can be used when required.

TABLE 15–2 Next Steps: Additional SEAM Tasks

Task	Description	For Instructions
Configure cross-realm authentication	Enables communications from one realm to another realm.	"Configuring Cross-Realm Authentication" on page 236
Configure SEAM clients	Enables a client to use SEAM services.	"Configuring SEAM Clients" on page 244
Configure SEAM NFS server	Enables a server to share a file system that requires Kerberos authentication.	"Configuring SEAM NFS Servers" on page 239

Configuring KDC Servers

After you install the SEAM software, you must configure the KDC servers. Configuring a master KDC and at least one slave KDC provides the service that issues credentials. These credentials are the basis for SEAM, so the KDCs must be installed before you attempt other tasks.

The most significant difference between a master KDC and a slave KDC is that only the master KDC can handle database administration requests. For instance, changing a password or adding a new principal must be done on the master KDC. These changes can then be propagated to the slave KDCs. Both the slave KDC and master KDC generate credentials. This feature provides redundancy in case the master KDC cannot respond.

▼ How to Configure a Master KDC

In this procedure, the following configuration parameters are used:

- Realm name = `EXAMPLE.COM`
- DNS domain name = `example.com`
- Master KDC = `kdc1.example.com`
- Slave KDC = `kdc2.example.com`
- admin principal = `kws/admin`
- Online help URL = `http://denver:8888/ab2/coll.384.1/SEAM/@AB2PageView/6956`

Note – Adjust the URL to point to the "SEAM Administration Tool" section, as described in "Online Help URL" on page 225.

---

1. **Complete the prerequisites for configuring a master KDC.**

   This procedure requires that DNS must be running. For specific naming instructions if this master is to be swappable, see "Swapping a Master KDC and a Slave KDC" on page 249.

2. **Become superuser on the master KDC.**

3. **Edit the Kerberos configuration file (`krb5.conf`).**

   You need to change the realm names and the names of the servers. See the `krb5.conf`(4) man page for a full description of this file.

```
kdc1 # cat /etc/krb5/krb5.conf
[libdefaults]
        default_realm = EXAMPLE.COM

[realms]
                EXAMPLE.COM = {
                kdc = kdc1.example.com
                kdc = kdc2.example.com
                admin_server = kdc1.example.com
        }
```

```
[domain_realm]
        .example.com = EXAMPLE.COM
#
# if the domain name and realm name are equivalent,
# this entry is not needed
#
[logging]
        default = FILE:/var/krb5/kdc.log
        kdc = FILE:/var/krb5/kdc.log

[appdefaults]
    gkadmin = {
        help_url = http://denver:8888/ab2/coll.384.1/SEAM/@AB2PageView/6956
        }
```

In this example, the lines for `domain_realm`, `kdc`, `admin_server`, and all `domain_realm` entries were changed. In addition, the line that defines the `help_url` was edited.

4. **Edit the KDC configuration file (`kdc.conf`).**

   You need to change the realm name. See the `kdc.conf`(4) man page for a full description of this file.

```
kdc1 # cat /etc/krb5/kdc.conf
[kdcdefaults]
        kdc_ports = 88,750

[realms]
        EXAMPLE.COM= {
                profile = /etc/krb5/krb5.conf
                database_name = /var/krb5/principal
                admin_keytab = /etc/krb5/kadm5.keytab
                acl_file = /etc/krb5/kadm5.acl
                kadmind_port = 749
                max_life = 8h 0m 0s
                max_renewable_life = 7d 0h 0m 0s
        }
```

   In this example, the realm name definition in the `realms` section was changed.

5. **Create the KDC database by using the `kdb5_util` command.**

   The `kdb5_util` command creates the KDC database. Also, when used with the `-s` option, this command creates a stash file that is used to authenticate the KDC to itself before the `kadmind` and `krb5kdc` daemons are started.

```
kdc1 # /usr/sbin/kdb5_util create -r EXAMPLE.COM -s
Initializing database '/var/krb5/principal' for realm 'EXAMPLE.COM'
master key name 'K/M@EXAMPLE.COM'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key:      <type the key>
Re-enter KDC database master key to verify:      <type it again>
```

   The `-r` option followed by the realm name is not required if the realm name is equivalent to the domain name in the server's name space.

6. **Edit the Kerberos access control list file (`kadm5.acl`).**

   Once populated, the `/etc/krb5/kadm5.acl` file should contain all principal names that are allowed to administer the KDC. The first entry that is added might look similar to the following:

   ```
   kws/admin@EXAMPLE.COM    *
   ```

   This entry gives the `kws/admin` principal in the `EXAMPLE.COM` realm the ability to modify principals or policies in the KDC. The default installation includes an asterisk (*) to match all `admin` principals. This default could be a security risk, so it is more secure to include a list of all of the `admin` principals. See the `kadm5.acl`(4) man page for more information.

7. **Start the `kadmin.local` command.**

   The next sub-steps create principals that are used by SEAM.

   ```
   kdc1 # /usr/sbin/kadmin.local
   kadmin.local:
   ```

   a. **Add administration principals to the database.**

      You can add as many `admin` principals as you need. You must add at least one `admin` principal to complete the KDC configuration process. For this example, a `kws/admin` principal is added. You can substitute an appropriate principal name instead of "`kws`."

      ```
      kadmin.local: addprinc kws/admin
      Enter password for principal kws/admin@EXAMPLE.COM:      <type the password>
      Re-enter password for principal kws/admin@EXAMPLE.COM:      <type it again>
      Principal "kws/admin@EXAMPLE.COM" created.
      kadmin.local:
      ```

   b. **Create a keytab file for the `kadmind` service.**

      This command sequence creates a special keytab file with principal entries for `kadmin` and `changepw`. These principals are needed for the `kadmind` service. Note that when the principal instance is a host name, the FQDN must be entered in lowercase letters, regardless of the case of the domainname in the `/etc/resolv.conf` file.

```
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab kadmin/kdc1.example.com
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type DES-CBC-CRC
          added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab changepw/kdc1.example.com
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type DES-CBC-CRC
          added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local:
```

   c. **Quit `kadmin.local`.**

      You have added all of the required principals for the next steps.

      ```
      kadmin.local: quit
      ```

8. **Start the Kerberos daemons.**

```
kdc1 # /etc/init.d/kdc start
kdc1 # /etc/init.d/kdc.master start
```

9. **Start `kadmin`.**

   At this point, you can add principals by using the SEAM Administration Tool. To do so, you must log on with one of the `admin` principal names that you created earlier in this procedure. However, the following command-line example is shown for simplicity.

   ```
   kdc1 # /usr/sbin/kadmin -p kws/admin
   Enter password:      <Type kws/admin password>
   kadmin:
   ```

   a. **Create the master KDC host principal.**

      The host principal is used by Kerberized applications (such as `klist` and `kprop`) . Note that when the principal instance is a host name, the FQDN must be entered in lowercase letters, regardless of the case of the domainname in the `/etc/resolv.conf` file.

      ```
      kadmin: addprinc -randkey host/kdc1.example.com
      Principal "host/kdc1.example.com@EXAMPLE.COM" created.
      kadmin:
      ```

   b. **(Optional) Create the master KDC `root` principal.**

      This principal is used for authenticated NFS-mounting. So, the principal might not be necessary on a master KDC. Note that when the principal instance is a host name, the FQDN must be entered in lowercase letters, regardless of the case of the domainname in the `/etc/resolv.conf` file.

```
kadmin: addprinc root/kdc1.example.com
Enter password for principal root/kdc1.example.com@EXAMPLE.COM:      <type the password>
Re-enter password for principal root/kdc1.example.com@EXAMPLE.COM:      <type it again>
Principal "root/kdc1.example.com@EXAMPLE.COM" created.
kadmin:
```

   c. **Add the master KDC's host principal to the master KDC's keytab file.**

      Adding the host principal to the keytab file allows this principal to be used automatically.

      ```
      kadmin: ktadd host/kdc1.example.com
      kadmin: Entry for principal host/kdc1.example.com with
        kvno 3, encryption type DES-CBC-CRC added to keytab
        WRFILE:/etc/krb5/krb5.keytab
      kadmin:
      ```

   d. **Quit `kadmin`.**

      ```
      kadmin: quit
      ```

10. **Add an entry for each KDC into the propagation configuration file (`kpropd.acl`).**

    See the `kprop`(1M) man page for a full description of this file.

```
kdc1 # cat /etc/krb5/kpropd.acl
host/kdc1.example.com@EXAMPLE.COM
host/kdc2.example.com@EXAMPLE.COM
```

11. **(Optional) Synchronize the master KDCs clock by using NTP or another clock synchronization mechanism.**

    It is not required to install and use the Network Time Protocol (NTP). However, every clock must be within the default time that is defined in the `libdefaults` section of the `krb5.conf` file in order for authentication to succeed. See "Synchronizing Clocks between KDCs and SEAM Clients" on page 247 for information about NTP.

▼ How to Configure a Slave KDC

In this procedure, a new slave KDC named `kdc3` is configured. This procedure uses the following configuration parameters:

- Realm name = `EXAMPLE.COM`
- DNS domain name = `example.com`
- Master KDC = `kdc1.example.com`
- Slave KDC = `kdc2.example.com` and `kdc3.example.com`
- admin principal = `kws/admin`
- Online help URL = `http://denver:8888/ab2/coll.384.1/SEAM/@AB2PageView/6956`

Note – Adjust the URL to point to the "SEAM Administration Tool" section, as described in the "Online Help URL" on page 225.

---

1. **Complete the prerequisites for configuring a slave KDC.**

   The master KDC must be configured. For specific instructions if this slave is to be swappable, see "Swapping a Master KDC and a Slave KDC" on page 249.

2. **On the master KDC, become superuser.**

3. **On the master KDC, start `kadmin`.**

   You must log on with one of the `admin` principal names that you created when you configure the master KDC.

   ```
   kdc1 # /usr/sbin/kadmin -p kws/admin
   Enter password:      <Enter kws/admin password>
   kadmin:
   ```

   a. **On the master KDC, add slave host principals to the database, if not already done.**

In order for the slave to function, it must have a host principal. Note that when the principal instance is a host name, the FQDN must be entered in lowercase letters, regardless of the case of the domainname in the `/etc/resolv.conf` file.

```
kadmin: addprinc -randkey host/kdc3.example.com
Principal "host/kdc3@EXAMPLE.COM" created.
kadmin:
```

b. (Optional) On the master KDC, create the slave KDC `root` principal.

This principal is only needed if the slave will be NFS-mounting an authenticated file system. Note that when the principal instance is a host name, the FQDN must be entered in lowercase letters, regardless of the case of the domainname in the `/etc/resolv.conf` file.

```
kadmin: addprinc root/kdc3.example.com
Enter password for principal root/kdc3.example.com@EXAMPLE.COM:      <type the password>
Re-enter password for principal root/kdc3.example.com@EXAMPLE.COM:      <type it again>
Principal "root/kdc3.example.com@EXAMPLE.COM" created.
kadmin:
```

c. Quit `kadmin`.

```
kadmin: quit
```

4. On the master KDC, edit the Kerberos configuration file (`krb5.conf`).

You need to add an entry for each slave. See the `krb5.conf`(4) man page for a full description of this file.

```
kdc1 # cat /etc/krb5/krb5.conf
[libdefaults]
        default_realm = EXAMPLE.COM

[realms]
                EXAMPLE.COM = {
                kdc = kdc1.example.com
                kdc = kdc2.example.com
                kdc = kdc3.example.com
                admin_server = kdc1.example.com
        }

[domain_realm]
        .example.com = EXAMPLE.COM
#
# if the domain name and realm name are equivalent,
# this entry is not needed
#
[logging]
        default = FILE:/var/krb5/kdc.log
        kdc = FILE:/var/krb5/kdc.log

[appdefaults]
    gkadmin = {
        help_url = http://denver:8888/ab2/coll.384.1/SEAM/@AB2PageView/6956
```

5. **On the master KDC, add an entry for each slave KDC into the database propagation configuration file (`kpropd.acl`).**

   See the `kprop`(1M) man page for a full description of this file.

   ```
   kdc1 # cat /etc/krb5/kpropd.acl
   host/kdc1.example.com@EXAMPLE.COM
   host/kdc2.example.com@EXAMPLE.COM
   host/kdc3.example.com@EXAMPLE.COM
   ```

6. **On all slave KDCs, copy the KDC administration files from the master KDC server.**

   This step needs to be followed on all slave KDCs, since the master KDC server has updated information that each KDC server needs. You can use `ftp` or a similar transfer mechanism to grab copies of the following files from the master KDC:

   - `/etc/krb5/krb5.conf`
   - `/etc/krb5/kdc.conf`
   - `/etc/krb5/kpropd.acl`

7. **On the new slave, add the slave's host principal to the slave's keytab file by using `kadmin`.**

   You must log on with one of the `admin` principal names that you created when you configure the master KDC. This entry allows `kprop` and other Kerberized applications to function. Note that when the principal instance is a host name, the FQDN must be entered in lowercase letters, regardless of the case of the domainname in the `/etc/resolv.conf` file.

   ```
   kdc3 # /usr/sbin/kadmin -p kws/admin
   Enter password:      <Type kws/admin password>
   kadmin: ktadd host/kdc3.example.com
   kadmin: Entry for principal host/kdc3.example.com with
     kvno 3, encryption type DES-CBC-CRC added to keytab
     WRFILE:/etc/krb5/krb5.keytab
   kadmin: quit
   ```

8. **On the master KDC, add slave KDC names to the `cron` job, which automatically runs the backups, by running `crontab -e`.**

   Add the name of each slave KDC server at the end of the `kprop_script` line.

   ```
   10 3 * * * /usr/lib/krb5/kprop_script kdc2.example.com kdc3.example.com
   ```

   You might also want to change the time of the backups. This configuration starts the backup process every day at 3:10 AM.

9. **On the master KDC, back up and propagate the database by using `kprop_script`.**

   If a backup copy of the database is already available, it is not necessary to complete another backup. See "How to Manually Propagate the Kerberos Database to the Slave KDCs" on page 255 for further instructions.

   ```
   kdc1 # /usr/lib/krb5/kprop_script kdc3.example.com
   Database propagation to kdc3.example.com: SUCCEEDED
   ```

10. **On the new slave, create a stash file by using `kdb5_util`.**

```
kdc3 # /usr/sbin/kdb5_util stash
kdb5_util: Cannot find/read stored master key while reading master key
kdb5_util: Warning: proceeding without master key

Enter KDC database master key:     <type the key>
```

11. **(Optional) On the new slave KDC, synchronize the master KDCs clock by using NTP or another clock synchronization mechanism.**

    It is not required to install and use the Network Time Protocol (NTP). However, every clock must be within the default time that is defined in the `libdefaults` section of the `krb5.conf` file in order for authentication to succeed. See "Synchronizing Clocks between KDCs and SEAM Clients" on page 247 for information about NTP.

12. **On the new slave, start the KDC daemon (`krb5kdc`).**

    ```
    kdc3 # /etc/init.d/kdc start
    ```

Configuring Cross-Realm Authentication

You have several ways of linking realms together so that users in one realm can be authenticated in another realm. Normally, this cross-realm authentication is accomplished by establishing a secret key that is shared between the two realms. The relationship of the realms can be either hierarchal or directional (see "Realm Hierarchy" on page 222).

▼ How to Establish Hierarchical Cross-Realm Authentication

The example in this procedure uses two realms, `ENG.EAST.EXAMPLE.COM` and `EAST.EXAMPLE.COM`. Cross-realm authentication will be established in both directions. This procedure must be completed on the master KDC in both realms.

1. **Complete the prerequisites for establishing hierarchical cross-realm authentication.**

   The master KDC for each realm must be configured. To fully test the authentication process, several clients or slave KDCs must be installed.

2. **Become superuser on the first master KDC.**

3. **Create ticket-granting ticket service principals for the two realms.**

   You must log on with one of the `admin` principal names that was created when you configured the master KDC.

```
# /usr/sbin/kadmin -p kws/admin
Enter password:     <Type kws/admin password>
```

```
kadmin: addprinc krbtgt/ENG.EAST.EXAMPLE.COM@EAST.EXAMPLE.COM
Enter password for principal krgtgt/ENG.EAST.EXAMPLE.COM@EAST.EXAMPLE.COM:      <type the
                                                                                 password>
kadmin: addprinc krbtgt/EAST.EXAMPLE.COM@ENG.EAST.EXAMPLE.COM
Enter password for principal krgtgt/EAST.EXAMPLE.COM@ENG.EAST.EXAMPLE.COM:      <type the
                                                                                 password>
kadmin: quit
```

Note – The password that is entered for each service principal must be identical in both KDCs. Thus, the password for the service principal `krbtgt/ENG.EAST.EXAMPLE.COM@EAST.EXAMPLE.COM` must be the same in both realms.

---

4. **Add entries to the Kerberos configuration file to define domain names for every realm (`krb5.conf`).**

   ```
   # cat /etc/krb5/krb5.conf
   [libdefaults]
    .
    .
   [domain_realm]
           .eng.east.example.com = ENG.EAST.EXAMPLE.COM
           .east.example.com = EAST.EXAMPLE.COM
   ```

   In this example, domain names for the `ENG.EAST.EXAMPLE.COM` and `EAST.EXAMPLE.COM` realms are defined. It is important to include the subdomain first, since the file is searched top down.

5. **Copy the Kerberos configuration file to all clients in this realm.**

   In order for cross-realm authentication to work, all systems (including slave KDCs and other servers) must have the new version of the Kerberos configuration file (`/etc/krb5/krb5.conf`) installed.

6. **Repeat these steps in the second realm.**

▼ How to Establish Direct Cross-Realm Authentication

The example in this procedure uses two realms: `ENG.EAST.EXAMPLE.COM` and `SALES.WEST.EXAMPLE.COM`. Cross-realm authentication will be established in both directions. This procedure must be completed on the master KDC in both realms.

1. **Complete the prerequisites for establishing direct cross-realm authentication.**

   The master KDC for each realm must be configured. To fully test the authentication process, several clients or slave KDCs must be installed.

2. **Become superuser on one of the master KDC servers.**

3. **Create ticket-granting ticket service principals for the two realms.**

   You must log on with one of the `admin` principal names that was created when you configured the master KDC.

```
# /usr/sbin/kadmin -p kws/admin
Enter password:      <Type kws/admin password>
kadmin: addprinc krbtgt/ENG.EAST.EXAMPLE.COM@SALES.WEST.EXAMPLE.COM
Enter password for principal
  krgtgt/ENG.EAST.EXAMPLE.COM@SALES.WEST.EXAMPLE.COM:      <type the password>
kadmin: addprinc krbtgt/SALES.WEST.EXAMPLE.COM@ENG.EAST.EXAMPLE.COM
Enter password for principal
  krgtgt/SALES.WEST.EXAMPLE.COM@ENG.EAST.EXAMPLE.COM:      <type the password>
kadmin: quit
```

Note – The password that is entered for each service principal must be identical in both KDCs. Thus, the password for the service principal `krbtgt/ENG.EAST.EXAMPLE.COM@SALES.WEST.EXAMPLE.COM` must be the same in both realms.

---

4. **Add entries in the Kerberos configuration file to define the direct path to the remote realm (`krb5.conf`).**

   This example shows the clients in the `ENG.EAST.EXAMPLE.COM` realm. You would need to swap the realm names to get the appropriate definitions in the `SALES.WEST.EXAMPLE.COM` realm.

```
# cat /etc/krb5/krb5.conf
[libdefaults]
 .
 .
[capaths]
    ENG.EAST.EXAMPLE.COM = {
        SALES.WEST.EXAMPLE.COM = .
    }

    SALES.WEST.EXAMPLE.COM = {
         ENG.EAST.EXAMPLE.COM = .
    }
```

5. **Copy the Kerberos configuration file to all clients in the current realm.**

   In order for cross-realm authentication to work, all systems (including slave KDCs and other servers) must have the new version of the Kerberos configuration file (`krb5.conf`) installed.

6. **Repeat these steps for the second realm.**

Configuring SEAM NFS Servers

NFS services use UNIX user IDs (UIDs) to identify a user and cannot directly use principals. To translate the principal to a UID, a credential table that maps user principals to UNIX UIDs must be created. The procedures in this section focus on the tasks that are necessary to configure a SEAM NFS server, to administer the credential table, and to initiate Kerberos security modes for NFS-mounted file systems. The following task map describes the tasks that are covered in this section.

TABLE 15–3 Configuring SEAM NFS Servers (Task Map)

Task	Description	For Instructions
Configure a SEAM NFS server	Enables a server to share a file system that requires Kerberos authentication.	"How to Configure SEAM NFS Servers" on page 239
Create a credential table	Generates a credential table.	"How to Create a Credential Table" on page 241
Change the credential table that maps user principles to UNIX UIDs	Updates information in the credential table.	"How to Add a Single Entry to the Credential Table" on page 241
Share a file system with Kerberos authentication	Shares a file system with security modes so that Kerberos authentication is required.	"How to Set Up a Secure NFS Environment With Multiple Kerberos Security Modes" on page 242

▼ How to Configure SEAM NFS Servers

In this procedure, the following configuration parameters are used:

- Realm name = `EXAMPLE.COM`
- DNS domain name = `example.com`
- NFS server = `denver.example.com`
- admin principal = `kws/admin`

1. **Complete the prerequisites for configuring a SEAM NFS server.**

   The master KDC must be configured. To fully test the process, you need several clients.

2. **(Optional) Install the NTP client or other clock synchronization mechanism.**

   It is not required to install and use the Network Time Protocol (NTP). However, every clock must be within the default time that is defined in the `libdefaults` section of the `krb5.conf` file in order for authentication to succeed. See "Synchronizing Clocks between KDCs and SEAM Clients" on page 247 for information about NTP.

3. **Start `kadmin`.**

   You can use the SEAM Administration Tool to add a principal, as explained in "How to Create a New Principal" on page 284. To do so, you must log on with one of the `admin` principal names that you created when you configured the master KDC. However, the following example shows how to add the required principals by using the command line.

```
denver # /usr/sbin/kadmin -p kws/admin
Enter password:      <Type kws/admin password>
kadmin:
```

   a. **Create the server's NFS service principal.**

      Note that when the principal instance is a host name, the FQDN must be entered in lowercase letters, regardless of the case of the domainname in the `/etc/resolv.conf` file.

```
kadmin: addprinc -randkey nfs/denver.example.com
Principal "nfs/denver.example.com" created.
kadmin:
```

   b. **(Optional) Create a `root` principal for the NFS server.**

```
kadmin: addprinc root/denver.example.com
Enter password for principal root/denver.example.com@EXAMPLE.COM:      <type the password>
Re-enter password for principal root/denver.example.com@EXAMPLE.COM:      <type it again>
Principal "root/denver.example.com@EXAMPLE.COM" created.
kadmin:
```

   c. **Add the server's NFS service principal to the server's keytab file.**

```
kadmin: ktadd nfs/denver.example.com
kadmin: Entry for principal nfs/denver.example.com with
  kvno 3, encryption type DES-CBC-CRC added to keytab
  WRFILE:/etc/krb5/krb5.keytab
kadmin:
```

   d. **Quit `kadmin`.**

```
kadmin: quit
```

4. **Create the `gsscred` table.**

   See "How to Create a Credential Table" on page 241 for more information.

5. **Share the NFS file system with Kerberos security modes.**

   See "How to Set Up a Secure NFS Environment With Multiple Kerberos Security Modes" on page 242 for more information.

6. **On each client, authenticate both the user principal and the `root` principal.**

▼ How to Create a Credential Table

The `gsscred` credential table is used by an NFS server to map SEAM principals to a UID. In order for NFS clients to mount file systems from an NFS server with Kerberos authentication, this table must be created or made available.

1. **Edit `/etc/gss/gsscred.conf` and change the mechanism.**

   Change the mechanism to `files`.

2. **Create the credential table by using `gsscred`.**

   ```
   # gsscred -m kerberos_v5 -a
   ```

   The `gsscred` command gathers information from all sources that are listed with the `passwd` entry in the `/etc/nsswitch.conf` file. You might need to temporarily remove the `files` entry, if you do not want the local password entries included in the credential table. See the `gsscred`(1M) man page for more information.

▼ How to Add a Single Entry to the Credential Table

This procedure requires that the `gsscred` table has already been created on the NFS server.

1. **Become superuser on a NFS server.**

2. **Add an entry to the table by using `gsscred`.**

   ```
   # gsscred -m mech [ -n name [ -u uid ]] -a
   ```

mech	Defines the security mechanism to be used.
name	Defines the principal name for the user, as defined in the KDC.
uid	Defines the UID for the user, as defined in the password database.
`-a`	Adds the UID to principal name mapping.

Example—Adding a Single Entry to the Credential Table

In the following example, an entry is added for the user named `sandy`, which is mapped to UID `3736`. The UID is pulled from the password file if it is not included on the command line.

```
# gsscred -m kerberos_v5 -n sandy -u 3736 -a
```

▼ How to Set Up a Secure NFS Environment With Multiple Kerberos Security Modes

1. **Become superuser on the NFS server.**

2. **Verify that there is a NFS service principal in the keytab file.**

   The `klist` command reports if there is a keytab file and displays the principals. If the results show that there is no keytab file or that there is no NFS service principal, you need to verify the completion of all of the steps in "How to Configure SEAM NFS Servers" on page 239.

   ```
   # klist -k
   Keytab name: FILE:/etc/krb5/krb5.keytab
   KVNO Principal
   ---- --------------------------------------------------------
      3 nfs/denver.example.com@EXAMPLE.COM
   ```

3. **Enable Kerberos security modes in the `/etc/nfssec.conf` file.**

   Edit the `/etc/nfssec.conf` file and remove the "#" from in front of the Kerberos security modes.

   ```
   # cat /etc/nfssec.conf
    .
    .
   #
   # Uncomment the following lines to use Kerberos V5 with NFS
   #
   krb5            390003  kerberos_v5     default -               # RPCSEC_GSS
   krb5i           390004  kerberos_v5     default integrity       # RPCSEC_GSS
   krb5p           390005  kerberos_v5     default privacy         # RPCSEC_GSS
   ```

4. **Edit the `/etc/dfs/dfstab` file and add the `sec=` option with the required security modes to the appropriate entries.**

   `share -F nfs -o sec=`*mode file-system*

mode	Specifies the security modes to be used when sharing. When using multiple security modes, the first mode in the list is used as the default by the automounter.
file-system	Defines the path to the file system to be shared.

   All clients that attempt to access files from the named file system require Kerberos authentication. To access files, both the user principal and the `root` principal on the NFS client should be authenticated.

5. **Make sure that the NFS service is running on the server.**

If this command is the first `share` command or set of `share` commands that you have initiated, it is likely that the NFS daemons are not running. The following commands kill the daemons and restart them.

```
# /etc/init.d/nfs.server stop
# /etc/init.d/nfs.server start
```

6. **(Optional) If the automounter is being used, edit the auto_master database to select a security mode other than the default.**

   You need not follow this procedure if you are not using the automounter to access the file system or if the default selection for the security mode is acceptable.

   file-system `auto_home -nosuid,sec=`*mode*

7. **(Optional) Manually issue the `mount` command to access the file system by using a non-default mode.**

   Alternatively, you could use the `mount` command to specify the security mode, but this alternative does not take advantage of the automounter:

   ```
   # mount -F nfs -o sec=mode file-system
   ```

Example—Sharing a File System With One Kerberos Security Mode

In this example, the `dfstab` file line means that Kerberos authentication must succeed before any files can be accessed through the NFS service.

```
# grep krb /etc/dfs/dfstab
share -F nfs -o sec=krb5 /export/home
```

Example—Sharing a File System With Multiple Kerberos Security Modes

In this example, all three Kerberos security modes have been selected. If no security mode is specified when a mount request is made, the first mode that is listed is used on all NFS V3 clients (in this case, `krb5`). See the `nfssec.conf`(4) man page for more information.

```
# grep krb /etc/dfs/dfstab
share -F nfs -o sec=krb5:krb5i:krb5p /export/home
```

Configuring SEAM Clients

SEAM clients include any host, not a KDC server, on the network that needs to use SEAM services. This section provides a procedure for installing a SEAM client, as well as specific information about using `root` authentication to mount NFS file systems.

▼ How to Configure a SEAM Client

In this procedure, the following configuration parameters are used:

- Realm name = `EXAMPLE.COM`
- DNS domain name = `example.com`
- Master KDC = `kdc1.example.com`
- Slave KDC = `kdc2.example.com`
- Client = `client.example.com`
- admin principal = `kws/admin`
- User principal = `mre`
- Online help URL = `http://denver:8888/ab2/coll.384.1/SEAM/@AB2PageView/6956`

Note – Adjust the URL to point to the "SEAM Administration Tool" section, as described in the "Online Help URL" on page 225.

---

1. **Become superuser.**
2. **Edit the Kerberos configuration file (`krb5.conf`).**

   To change the file from the SEAM default version, you need to change the realm names and the names of the servers. You also need to identify the path to the help files for `gkadmin`.

```
kdc1 # cat /etc/krb5/krb5.conf
[libdefaults]
        default_realm = EXAMPLE.COM

[realms]
                EXAMPLE.COM = {
                kdc = kdc1.example.com
                kdc = kdc2.example.com
                admin_server = kdc1.example.com
        }
```

```
[domain_realm]
        .example.com = EXAMPLE.COM
#
# if the domain name and realm name are equivalent,
# this entry is not needed
#
[logging]
        default = FILE:/var/krb5/kdc.log
        kdc = FILE:/var/krb5/kdc.log

[appdefaults]
    gkadmin = {
        help_url = http://denver:8888/ab2/coll.384.1/SEAM/@AB2PageView/6956
```

3. **(Optional) Synchronize the client's clock with the master KDC's clock by using NTP or another clock synchronization mechanism.**

   It is not required to install and use the Network Time Protocol (NTP). However, every clock must be within the default time that is defined in the `libdefaults` section of the `krb5.conf` file in order for authentication to succeed. See "Synchronizing Clocks between KDCs and SEAM Clients" on page 247 for information about NTP.

4. **(Optional) Create a user principal if a user principal does not already exist.**

   You need to create a user principal only if the user associated with this host does not have a principal assigned already. See "How to Create a New Principal" on page 284 for instructions on using the SEAM Administration Tool. The following is a command-line example.

```
client1 # /usr/sbin/kadmin -p kws/admin
Enter password:     <Type kws/admin password>
kadmin: addprinc mre
Enter password for principal mre@EXAMPLE.COM:     <type the password>
Re-enter password for principal mre@EXAMPLE.COM:     <type it again>
kadmin:
```

5. **Create a `root` principal.**

   Note that when the principal instance is a host name, the FQDN must be entered in lowercase letters, regardless of the case of the domainname in the `/etc/resolv.conf` file.

```
kadmin: addprinc root/client1.example.com
Enter password for principal root/client1.example.com@EXAMPLE.COM:     <type the password>
Re-enter password for principal root/client1.example.com@EXAMPLE.COM:     <type it again>
kadmin: quit
```

6. **(Optional) To use Kerberos with NFS, enable Kerberos security modes in the `/etc/nfssec.conf` file.**

   Edit the `/etc/nfssec.conf` file and remove the "#" from in front of the Kerberos security modes.

```
# cat /etc/nfssec.conf
 .
 .
```

```
#
# Uncomment the following lines to use Kerberos V5 with NFS
#
krb5            390003  kerberos_v5     default -               # RPCSEC_GSS
krb5i           390004  kerberos_v5     default integrity       # RPCSEC_GSS
krb5p           390005  kerberos_v5     default privacy         # RPCSEC_GSS
```

7. **(Optional) If you want a user on the SEAM client to automatically mount Kerberized NFS file systems that use Kerberos authentication, you must authenticate the `root` user.**

   This process is done most securely by using the `kinit` command. However, users will need to use `kinit` as `root` every time they need to mount a file system that is secured by Kerberos. You can choose to use a keytab file instead. For detailed information about the keytab file requirement, see "Setting Up `Root` Authentication to Mount NFS File Systems" on page 247.

   ```
   client1 # /usr/bin/kinit root/client1.example.com
   Password for root/client1.example.com@EXAMPLE.COM:      <Type password>
   ```

   To use the keytab file option, add the `root` principal to the client's keytab by using `kadmin`:

   ```
   client1 # /usr/sbin/kadmin -p kws/admin
   Enter password:      <Type kws/admin password>
   kadmin: ktadd root/client1.example.com
   kadmin: Entry for principal root/client.example.com with
     kvno 3, encryption type DES-CBC-CRC added to keytab
     WRFILE:/etc/krb5/krb5.keytab
   kadmin: quit
   ```

8. **If you want the client to warn users about Kerberos ticket expiration, create an entry in the `/etc/krb5/warn.conf` file.**

   See the `warn.conf`(4) man page for more information.

Example—Setting Up a SEAM Client Using a Non-SEAM KDC

It is possible to set up a SEAM client to work with a non-SEAM KDC. In this case, a line must be included in the `/etc/krb5/krb5.conf` file in the `realms` section. This line changes the protocol that is used when the client is communicating with the Kerberos password-changing server. The format of this line follows.

```
[realms]
                EXAMPLE.COM = {
                kdc = kdc1.example.com
                kdc = kdc2.example.com
                admin_server = kdc1.example.com
                kpasswd_protocol = SET_CHANGE
        }
```

Setting Up `Root` Authentication to Mount NFS File Systems

If users want to access a non-Kerberized NFS file system, either the NFS file system can be mounted as `root`, or the file system can be accessed automatically through the automounter whenever users access it (without requiring `root` permissions).

Mounting a Kerberized NFS file system is very much the same, but it does incur an additional obstacle. To mount a Kerberized NFS file system, users must use the `kinit` command as `root` to obtain credentials for the client's `root` principal, because a client's `root` principal is typically not in the client's keytab. This step is required even when the automounter is set up. This step also forces all users to know their system's `root` password and the `root` principal's password.

To bypass this step, you can add a client's `root` principal to the client's keytab file, which automatically provides credentials for `root`. Although this solution enables users to mount NFS file systems without running the `kinit` command and enhances ease-of-use, it is a security risk. For example, if someone gains access to a system with the `root` principal in its keytab, this person can obtain credentials for `root`. So make sure that you take the appropriate security precautions. See "Administering Keytab Files" on page 303 for more information.

Synchronizing Clocks between KDCs and SEAM Clients

All hosts that participate in the Kerberos authentication system must have their internal clocks synchronized within a specified maximum amount of time (known as *clock skew*). This requirement provides another Kerberos security check. If the clock skew is exceeded between any of the participating hosts, client requests are rejected.

The clock skew also determines how long application servers must keep track of all Kerberos protocol messages, in order to recognize and reject replayed requests. So, the longer the clock skew value, the more information that application servers have to collect.

The default value for the maximum clock skew is 300 seconds (five minutes). You can change this default in the `libdefaults` section of the `krb5.conf` file.

Note – For security reasons, do not increase the clock skew beyond 300 seconds.

Since it is important to maintain synchronized clocks between the KDCs and SEAM clients, you should use the Network Time Protocol (NTP) software to synchronize them. NTP public domain software from the University of Delaware is included in the Solaris software, starting with the Solaris 2.6 release.

Note – Another way to synchronize clocks is to use the `rdate` command and `cron` jobs, a process that can be less involved than using NTP. However, this section will continue to focus on using NTP. And, if you use the network to synchronize the clocks, the clock synchronization protocol must itself be secure.

NTP enables you to manage precise time or network clock synchronization, or both, in a network environment. NTP is basically a server/client implementation. You pick one system to be the master clock (the NTP server). Then, you set up all your other systems (the NTP clients) to synchronize their clocks with the master clock.

To synchronize the clocks, NTP uses the `xntpd` daemon, which sets and maintains a UNIX system time-of-day in agreement with Internet standard time servers. The following shows an example of this server/client NTP implementation.

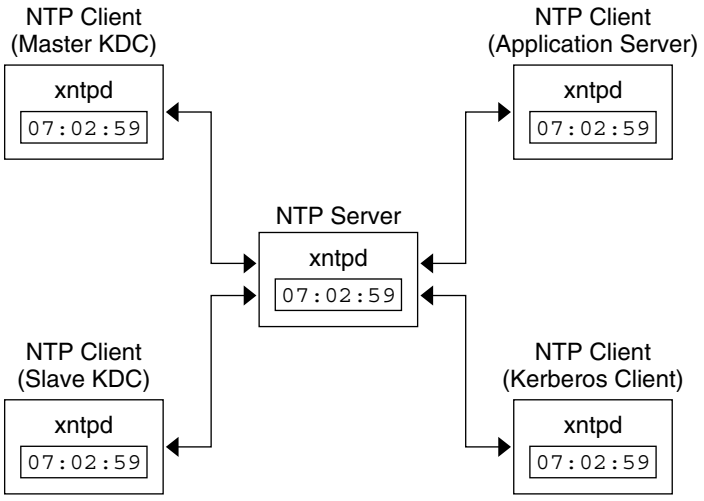


FIGURE 15–1 Synchronizing Clocks by Using NTP

To ensure that the KDCs and SEAM clients maintain synchronized clocks, implement the following steps:

1. Set up an NTP server on your network (this server can be any system, except the master KDC). See "Managing Network Time Protocol (Tasks)" in *System Administration Guide: Resource Management and Network Services* to find the NTP server task.

2. As you configure the KDCs and SEAM clients on the network, set them up to be NTP clients of the NTP server. See "Managing Network Time Protocol (Tasks)" in *System Administration Guide: Resource Management and Network Services* to find the NTP client task.

Swapping a Master KDC and a Slave KDC

You should use the procedures in this section to make the swap of a master KDC with a slave KDC easier. You should swap the master KDC with a slave KDC only if the master KDC server fails for some reason, or if the master KDC needs to be re-installed (for example, because new hardware is installed).

▼ How to Configure a Swappable Slave KDC

Perform this procedure on the slave KDC server that you want to have available to become the master KDC.

1. **Use alias names for the master KDC and the swappable slave KDC during the KDC installation.**

   When you define the host names for the KDCs, make sure that each system has an alias included in DNS. Also, use the alias names when you define the hosts in the `/etc/krb5/krb5.conf` file.

2. **Follow the steps to install a slave KDC.**

   Prior to any swap, this server should function as any other slave KDC in the realm. See "How to Configure a Slave KDC" on page 233 for instructions.

3. **Move the master KDC commands.**

   To prevent the master KDC commands from being run from this slave KDC, move the `kprop`, `kadmind` and `kadmin.local` commands to a reserved place.

```
kdc4 # mv /usr/lib/krb5/kprop /usr/lib/krb5/kprop.save
kdc4 # mv /usr/lib/krb5/kadmind /usr/lib/krb5/kadmind.save
kdc4 # mv /usr/sbin/kadmin.local /usr/sbin/kadmin.local.save
```

4. **Comment out the `kprop` line in the `root` crontab file.**

   This step prevents the slave KDC from propagating its copy of the KDC database.

```
kdc4 # crontab -e
#ident  "@(#)root       1.20    01/11/06 SMI"
#
```

```
# The root crontab should be used to perform accounting data collection.
#
# The rtc command is run to adjust the real time clock if and when
# daylight savings time changes.
#
10 3 * * * /usr/sbin/logadm
15 3 * * 0 /usr/lib/fs/nfs/nfsfind
1 2 * * * [ -x /usr/sbin/rtc ] && /usr/sbin/rtc -c > /dev/null 2>&1
30 3 * * * [ -x /usr/lib/gss/gsscred_clean ] && /usr/lib/gss/gsscred_clean
#10 3 * * * /usr/lib/krb5kprop_script kdc1.example.sun.com #SUNWkr5ma
```

▼ How to Swap a Master KDC and a Slave KDC

This procedure requires that the slave KDC server has been set up as a swappable slave (see "How to Configure a Swappable Slave KDC" on page 249). In this procedure, the master KDC server that is being swapped out is named `kdc1`. The slave KDC that will become the new master KDC is named `kdc4`.

1. **On the old master KDC, kill the `kadmind` process.**

   ```
   kdc1 # /etc/init.d/kdc.master stop
   ```

   When you kill the `kadmind` process, you prevent any changes from being made to the KDC database.

2. **On the old master KDC, comment out the `kprop` line in the `root` crontab file.**

   ```
   kdc1 # crontab -e
   #ident  "@(#)root       1.20    01/11/06 SMI"
   #
   # The root crontab should be used to perform accounting data collection.
   #
   # The rtc command is run to adjust the real time clock if and when
   # daylight savings time changes.
   #
   10 3 * * * /usr/sbin/logadm
   15 3 * * 0 /usr/lib/fs/nfs/nfsfind
   1 2 * * * [ -x /usr/sbin/rtc ] && /usr/sbin/rtc -c > /dev/null 2>&1
   30 3 * * * [ -x /usr/lib/gss/gsscred_clean ] && /usr/lib/gss/gsscred_clean
   #10 3 * * * /usr/lib/krb5/kprop_script kdc2.example.sun.com #SUNWkr5ma
   ```

   This step prevents the old master from propagating its copy of the KDC database.

3. **On the old master KDC, run `kprop_script` to back up and propagate the database.**

   ```
   kdc1 # /usr/lib/krb5/kprop_script kdc4.example.com
   Database propagation to kdc4.example.com: SUCCEEDED
   ```

4. **On the old master KDC, move the master KDC commands.**

   To prevent the master KDC commands from being run, move the `kprop`, `kadmind` and `kadmin.local` commands to a reserved place.

```
kdc4 # mv /usr/lib/krb5/kprop /usr/lib/krb5/kprop.save
kdc4 # mv /usr/lib/krb5/kadmind /usr/lib/krb5/kadmind.save
kdc4 # mv /usr/sbin/kadmin.local /usr/sbin/kadmin.local.save
kdc4 # mv /etc/krb5/kadm5.acl /etc/krb5/kadm5.acl.save
```

5. **On the DNS server, change the alias names for the master KDC.**

   To change the servers, edit the `example.com` zone file and change the entry for `masterkdc`.

   ```
   masterkdc IN CNAME kdc4
   ```

6. **On the DNS server, restart the Internet domain name server.**

   Run the following command on both servers to get the new alias information:

   ```
   # pkill -1 in.named
   ```

7. **On the new master KDC, move the master KDC commands.**

   ```
   kdc4 # mv /usr/lib/krb5/kprop.save /usr/lib/krb5/kprop
   kdc4 # mv /usr/lib/krb5/kadmind.save /usr/lib/krb5/kadmind
   kdc4 # mv /usr/sbin/kadmin.local.save /usr/sbin/kadmin.local
   ```

8. **On the new master KDC, edit the Kerberos access control list file (`kadm5.acl`).**

   Once populated, the `/etc/krb5/kadm5.acl` file should contain all principal names that are allowed to administer the KDC. The first entry that is added might look similar to the following:

   ```
   kws/admin@EXAMPLE.COM   *
   ```

   This entry gives the `kws/admin` principal in the `EXAMPLE.COM` realm the ability to modify principals or policies in the KDC. The default installation includes an asterisk (*) to match all `admin` principals. This default could be a security risk, so it is more secure to include a list of all of the `admin` principals. See the `kadm5.acl`(4) man page for more information.

9. **On the new master KDC, create a keytab file for `kadmin` by using `kadmin.local`.**

   This command sequence creates a special keytab file with principal entries for `admin` and `changepw`. These principals are needed for the `kadmind` service.

```
kdc4 # /usr/sbin/kadmin.local
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab kadmin/kdc4.example.com
Entry for principal kadmin/kdc4.example.com with kvno 3, encryption type DES-CBC-CRC
          added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab changepw/kdc4.example.com
Entry for principal changepw/kdc4.example.com with kvno 3, encryption type DES-CBC-CRC
          added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local: quit
```

10. **On the new master KDC, start `kadmind`.**

    ```
    kdc4 # /etc/init.d/kdc.master start
    ```

11. **Enable the `kprop` line in the `root` crontab file.**

```
kdc4 # crontab -e
#ident  "@(#)root       1.19    98/07/06 SMI"
#
# The root crontab should be used to perform accounting data collection.
#
# The rtc command is run to adjust the real time clock if and when
# daylight savings time changes.
#
10 3 * * * /usr/sbin/logadm
15 3 * * 0 /usr/lib/fs/nfs/nfsfind
1 2 * * * [ -x /usr/sbin/rtc ] && /usr/sbin/rtc -c > /dev/null 2>&1
30 3 * * * [ -x /usr/lib/gss/gsscred_clean ] && /usr/lib/gss/gsscred_clean
10 3 * * * /usr/lib/krb5/kprop_script kdc1.example.sun.com #SUNWkr5ma
```

Administering the Kerberos Database

The Kerberos database is the backbone of Kerberos and must be maintained properly. This section provides some procedures on how to administer the Kerberos database, such as backing up and restoring the database, setting up parallel propagation, and administering the stash file. The steps to initially set up the database are in "How to Configure a Master KDC" on page 229.

Backing Up and Propagating the Kerberos Database

Propagating the Kerberos database from the master KDC to the slave KDCs is one of the most important configuration tasks. If propagation doesn't happen often enough, the master KDC and the slave KDCs will lose synchronization. So, if the master KDC goes down, the slave KDCs will not have the most recent database information. Also, if a slave KDC has been configured as a master KDC for purposes of load balancing, the clients that use that slave KDC as a master KDC will not have the latest information. Therefore, you must make sure that propagation occurs often enough, based on how often you change the Kerberos database.

When you configure the master KDC, you set up the `kprop_script` command in a `cron` job to automatically back up the Kerberos database to the `/var/krb5/slave_datatrans` dump file and propagate it to the slave KDCs. But, as with any file, the Kerberos database can become corrupted. If data corruption occurs on a slave KDC, you might never notice, since the next automatic propagation of the database installs a fresh copy. However, if corruption occurs on the master KDC, the corrupted database is propagated to all of the slave KDCs during the next propagation. And, the corrupted backup overwrites the previous uncorrupted backup file on the master KDC.

Because there is no “safe” backup copy in this scenario, you should also set up a `cron` job to periodically copy the `slave_datatrans` dump file to another location or to create another separate backup copy by using the `dump` command of `kdb5_util`. Then, if your database becomes corrupted, you can restore the most recent backup on the master KDC by using the `load` command of `kdb5_util`.

Another important note: Because the database dump file contains principal keys, you need to protect the file from being accessed by unauthorized users. By default, the database dump file has read and write permissions only as `root`. To protect against unauthorized access, use only the `kprop` command to propagate the database dump file, which encrypts the data that is being transferred. Also, `kprop` propagates the data only to the slave KDCs, which minimizes the chance of accidentally sending the database dump to unauthorized hosts.

Caution – If the Kerberos database is updated after it has been propagated and if the database subsequently is corrupted before the next propagation, the KDC slaves will not contain the updates. The updates will be lost. For this reason, if you add significant updates to the Kerberos database before a regularly scheduled propagation, you should manually propagate the database to avoid data loss.

The `kpropd.acl` File

The `kpropd.acl` file on a KDC provides a list of host principal names, one per line, that specifies the systems from which the KDC can receive an updated database through propagation. If the master KDC is used to propagate all the slave KDCs, the `kpropd.acl` file on each slave needs to contain only the host principal name of the master KDC.

However, the SEAM installation and subsequent configuration steps in this book instruct you to add the same `kpropd.acl` file to the master KDC and the slave KDCs. This file contains all the KDC host principal names. This configuration allows you to propagate from any KDC, in case the propagating KDCs become temporarily unavailable. And, by keeping an identical copy on all KDCs, you make the configuration easy to maintain.

The `kprop_script` Command

The `kprop_script` command uses the `kprop` command to propagate the Kerberos database to other KDCs. If the `kprop_script` command is run on a slave KDC, it propagates the slave KDC’s copy of the Kerberos database to other KDCs. The `kprop_script` accepts a list of host names for arguments, separated by spaces, which denote the KDCs to propagate.

When the `kprop_script` is run, it creates a backup of the Kerberos database to the `/var/krb5/slave_datatrans` file and copies the file to the specified KDCs. The Kerberos database is locked until the propagation is finished.

▼ How to Back Up the Kerberos Database

1. **Become superuser on the master KDC.**
2. **Back up the Kerberos database by using the `dump` command of the `kdb5_util` command.**

```
# /usr/sbin/kdb5_util dump [-verbose] [-d dbname] [filename [principals...]]
```

`-verbose`	Prints the name of each principal and policy that is being backed up.
dbname	Defines the name of the database to back up. Note that ".db" is appended to whatever database name is specified, and you can specify an absolute path for the file. If the `-d` option is not specified, the default database name is `/var/krb5/principal`, which actually becomes `/var/krb5/principal.db`.
filename	Defines the file that is used to back up the database. You can specify an absolute path for the file. If you don't specify a file, the database is dumped to standard output.
principal	Defines a list of one or more principals (separated by a space) to back up. You must use fully-qualified principal names. If you don't specify any principals, the entire database is backed up.

Example—Backing Up the Kerberos Database

In the following example, the Kerberos database is backed up to a file called `dumpfile`. Because the `-verbose` option is specified, each principal is printed as it is backed up.

```
# kdb5_util dump -verbose dumpfile
kadmin/kdc1.eng.example.com@ENG.EXAMPLE.COM
krbtgt/eng.example.com@ENG.EXAMPLE.COM
kadmin/history@ENG.EXAMPLE.COM
pak/admin@ENG.EXAMPLE.COM
pak@ENG.EXAMPLE.COM
changepw/kdc1.eng.example.com@ENG.EXAMPLE.COM
```

In the following example, the `pak` and `pak/admin` principals are backed up from the Kerberos database.

```
# kdb5_util dump -verbose dumpfile pak/admin@ENG.EXAMPLE.COM pak@ENG.EXAMPLE.COM
pak/admin@ENG.EXAMPLE.COM
pak@ENG.EXAMPLE.COM
```

▼ How to Restore the Kerberos Database

1. **Become superuser on the master KDC.**
2. **Restore the Kerberos database by using the `load` command of `kdb_util`.**

```
# /usr/sbin/kdb5_util load [-verbose] [-d dbname] [-update] [filename]
```

`-verbose`	Prints the name of each principal and policy that is being restored.
dbname	Defines the name of the database to restore. Note that ".db" is appended to whatever database name is specified, and you can specify an absolute path for the file. If the `-d` option is not specified, the default database name is `/var/krb5/principal`, which actually becomes `/var/krb5/principal.db`.
`-update`	Updates the existing database. Otherwise, a new database is created or the existing database is overwritten.
filename	Defines the file from which to restore the database. You can specify an absolute path for the file.

Example—Restoring the Kerberos Database

In the following example, the database called `database1.db` is restored into the current directory from the `dumpfile` file. Since the `-update` option isn't specified, a new database is created by the restore.

```
# kdb5_util load -d database1 dumpfile
```

▼ How to Manually Propagate the Kerberos Database to the Slave KDCs

This procedure shows you how to propagate the Kerberos database by using the `kprop` command. Use this procedure if you need to synchronize a slave KDC with the master KDC outside the periodic `cron` job. And, unlike the `kprop_script`, you can use `kprop` to propagate just the current database backup without first making a new backup of the Kerberos database.

1. **Become superuser on the master KDC.**
2. **(Optional) Back up the database by using the `kdb5_util` command.**

```
# /usr/sbin/kdb5_util dump /var/krb5/slave_datatrans
```

3. **Propagate the database to a slave KDC by using the `kprop` command.**

```
# /usr/lib/krb5/kprop -f /var/krb5/slave_datatrans slave_KDC
```

If you want to back up the database and propagate it to a slave KDC outside the periodic `cron` job, you can also use the `kprop_script` command as follows:

```
# /usr/lib/krb5/kprop_script slave_KDC
```

Setting Up Parallel Propagation

In most cases, the master KDC is used exclusively to propagate its Kerberos database to the slave KDCs. However, if your site has many slave KDCs, you might consider load-sharing the propagation process, known as *parallel propagation*.

Parallel propagation allows specific slave KDCs to share the propagation duties with the master KDC. This sharing of duties enables the propagation to be done faster and to lighten the work for the master KDC.

For example, say your site has one master KDC and six slave KDCs (shown in Figure 15–2), where `slave-1` through `slave-3` consist of one logical grouping and `slave-4` through `slave-6` consist of another logical grouping. To set up parallel propagation, you could have the master KDC propagate the database to `slave-1` and `slave-4`, and those KDC slaves could in turn propagate the database to the KDC slaves in their group.

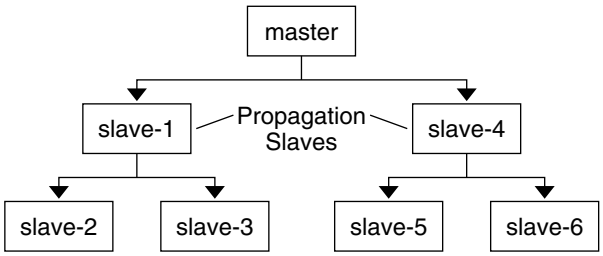


FIGURE 15–2 Example of Parallel Propagation Configuration

How to Set Up Parallel Propagation

The following is not a detailed step-by-step procedure, but a high-level list of configuration steps to enable parallel propagation.

1. On the master KDC, change the `kprop_script` entry in its `cron` job to include arguments for only the KDC slaves that will perform the succeeding propagation (*propagation slaves*).
2. On each propagation slave, add a `kprop_script` entry to its `cron` job, which must include arguments for the slaves to propagate. To successfully propagate in parallel, the cron job should be set up to run after the propagation slave is itself propagated with the new Kerberos database.

Note – How long it will take for a propagation slave to be propagated depends on factors such as network bandwidth and the size of the database.

3. On each slave KDC, set up the appropriate permissions to be propagated. This step is done by adding the host principal name of its propagating KDC to its `kpropd.acl` file.

Example—Setting Up Parallel Propagation

Using the example in Figure 15–2, the master KDC's `kprop_script` entry would look similar to the following:

```
0 3 * * * /usr/lib/krb5/kprop_script slave-1.example.com slave-4.example.com
```

The `slave-1`'s `kprop_script` entry would look similar to the following:

```
0 4 * * * /usr/lib/krb5/kprop_script slave-2.example.com slave-3.example.com
```

Note that the propagation on the slave starts an hour after it is propagated by the master.

The `kpropd.acl` file on the propagation slaves would contain the following entry:

```
host/master.example.com@EXAMPLE.COM
```

The `kpropd.acl` file on the KDC slaves being propagated by `slave-1` would contain the following entry:

```
host/slave-1.example.com@EXAMPLE.COM
```

Administering the Stash File

The stash file contains the master key for the Kerberos database, which is automatically created when you create a Kerberos database. If the stash file gets corrupted, you can use the `stash` command of the `kdb5_util` utility to replace the corrupted file. The only time you should need to remove a stash file is after removing the Kerberos database with the `destroy` command of `kdb5_util`. Because the stash file isn't automatically removed with the database, you have to remove it to finish the cleanup.

▼ How to Remove a Stash File

1. **Become superuser on the KDC that contains the stash file.**
2. **Remove the stash file.**

```
# rm stash-file
```

In this example, *stash-file* is the path to the stash file. By default, the stash file is located at `/var/krb5/.k5.realm`.

If you need to re-create the stash file, you can use the `-f` option of the `kdb5_util` command.

Increasing Security

Follow these steps to increase security on SEAM application servers and on KDC servers.

▼ How to Restrict Access to KDC Servers

Both master KDC servers and slave KDC servers have copies of the KDC database stored locally. Restricting access to these servers so that the databases are secure is important to the overall security of the SEAM installation.

1. **Disable remote services in the `/etc/inetd.conf` file.**

   To provide a secure KDC server, all nonessential network services should be disabled by commenting out the entry that starts the service in the `/etc/inetd.conf` file. In most circumstances, the only services that would need to run would be `time` and `krdb5_kprop`. In addition, any services that use loopback tli (`ticlts`, `ticotsord`, and `ticots`) can be left enabled. After you edit the file, it should look similar to the following (to shorten the example many comments have been removed):

```
kdc1 # cat /etc/inetd.conf
#
#ident  "@(#)inetd.conf 1.33    98/06/02 SMI"   /* SVr4.0 1.5   */
  .
  .
#name     dgram   udp     wait    root    /usr/sbin/in.tnamed     in.tnamed
#
#shell    stream  tcp     nowait  root    /usr/sbin/in.rshd       in.rshd
#login    stream  tcp     nowait  root    /usr/sbin/in.rlogind    in.rlogind
#exec     stream  tcp     nowait  root    /usr/sbin/in.rexecd     in.rexecd
#comsat   dgram   udp     wait    root    /usr/sbin/in.comsat     in.comsat
#talk     dgram   udp     wait    root    /usr/sbin/in.talkd      in.talkd
#
#uucp     stream  tcp     nowait  root    /usr/sbin/in.uucpd      in.uucpd
#
#finger   stream  tcp     nowait  nobody  /usr/sbin/in.fingerd    in.fingerd
#
# Time service is used for clock synchronization.
```

```
#
time      stream  tcp     nowait  root    internal
time      dgram   udp     wait    root    internal
#
  .
  .
#
100234/1  tli rpc/ticotsord wait  root    /usr/lib/gss/gssd     gssd
#dtspc    stream  tcp     nowait  root    /usr/dt/bin/dtspcd      /usr/dt/bin/dtspcd
#100068/2-5 dgram rpc/udp wait    root    /usr/dt/bin/rpc.cmsd    rpc.cmsd
100134/1 tli rpc/ticotsord wait   root    /usr/lib/ktkt_warnd kwarnd
krb5_prop stream  tcp     nowait  root    /usr/lib/krb5/kpropd  kpropd
```

Reboot the KDC server after the changes are made.

2. **Restrict access to the hardware that supports the KDC.**

   In order to restrict physical access, make sure that the KDC server and its monitor are located in a secure facility. Users should not be able to access this server in any way.

3. **Store KDC database backups on local disks or on the KDC slaves.**

   Make tape backups of your KDC only if the tapes are stored securely. Follow the same practice for copies of keytab files. It would be best to store these files on a local file system that is not shared to other systems. The storage file system can be on either the master KDC server or any of the slave KDCs.

CHAPTER 16

SEAM Error Messages and Troubleshooting

This chapter provides resolutions for error messages that you might receive when you use SEAM, as well as some troubleshooting tips for various problems. This is a list of the error message and troubleshooting information in this chapter.

SEAM Error Messages

This section provides information about SEAM error messages, including why each error occurs and a way to fix it.

SEAM Administration Tool Error Messages

`Unable to view the list of principals or policies; use the Name field.`

Cause: The `admin` principal that you logged in with does not have the list privilege (`l`) in the Kerberos ACL file (`kadm5.acl`), so you cannot view the principal list or policy list.

Solution: You must enter the principal and policy names in the Name field to work on them, or you need to log on with a principal that has the appropriate privileges.

`JNI: Java array creation failed`

`JNI: Java class lookup failed`

`JNI: Java field lookup failed`

`JNI: Java method lookup failed`

`JNI: Java object lookup failed`

`JNI: Java object field lookup failed`

`JNI: Java string access failed`

`JNI: Java string creation failed`

Cause: A serious problem exists with the Java Native Interface that is used by the SEAM Administration Tool (`gkadmin`).

Solution: Exit `gkadmin` and restart it. If the problem persists, please report a bug.

Common SEAM Error Messages (A-M)

This section provides an alphabetical list (A-M) of common error messages for the SEAM commands, SEAM daemons, PAM framework, GSS interface, the NFS service, and the Kerberos library.

major_error minor_error `gssapi error importing name`

Cause: An error occurred while a service name was being imported.

Solution: Make sure that the service principal is in the host's keytab file.

`Bad krb5 admin server hostname while initializing kadmin interface`

Cause: An invalid host name is configured for `admin_server` in the `krb5.conf` file.

Solution: Make sure that the correct host name for the master KDC is specified on the `admin_server` line in the `krb5.conf` file.

`Cannot contact any KDC for requested realm`

Cause: No KDC responded in the requested realm.

Solution: Make sure that at least one KDC (either the master or slave) is reachable or that the `krb5kdc` daemon is running on the KDCs. Check the `/etc/krb5/krb5.conf` file for the list of configured KDCs (`kdc` = *kdc_name*).

`Cannot determine realm for host`

Cause: Kerberos cannot determine the realm name for the host.

Solution: Make sure that there is a default realm name, or that the domain name mappings are set up in the Kerberos configuration file (`krb5.conf`).

`Cannot find KDC for requested realm`
Cause: No KDC was found in the requested realm.

Solution: Make sure that the Kerberos configuration file (`krb5.conf`) specifies a KDC in the `realm` section.

`cannot initialize realm` *realm_name*
Cause: The KDC might not have a stash file.

Solution: Make sure that the KDC has a stash file. If not, create a stash file by using the `kdb5_util` command, and try running the `krb5kdc`command again. The easiest way to start `krb5kdc` is to run the `/etc/init.d/kdc` script.

`Cannot resolve KDC for requested realm`
Cause: Kerberos cannot determine any KDC for the realm.

Solution: Make sure that the Kerberos configuration file (`krb5.conf`) specifies a KDC in the `realm` section.

`Cannot reuse password`
Cause: The password that you entered has been used before by this principal.

Solution: Choose a password that has not been chosen before, at least not within the number of passwords that are kept in the KDC database for each principal (this policy is enforced by the principal's policy).

`Can't get forwarded credentials`
Cause: Credential forwarding could not be established.

Solution: Make sure that the principal has forwardable credentials.

`Can't open/find Kerberos configuration file`
Cause: The Kerberos configuration file (`krb5.conf`) was unavailable.

Solution: Make sure that the `krb5.conf` file is available in the correct location and has the correct permissions. This file should be writable by `root` and readable by everyone else.

`Client/server realm mismatch in initial ticket request`
Cause: A realm mismatch between the client and server occurred in the initial ticket request.

Solution: Make sure that the server you are communicating with is in the same realm as the client, or that the realm configurations are correct.

`Client or server has a null key`
Cause: The principal has a null key.

Solution: Modify the principal to have a non-null key by using the `cpw` command of `kadmin`.

`Communication failure with server while initializing kadmin interface`

Cause: The host that was entered for the admin server, also called the master KDC, did not have the `kadmind` daemon running.

Solution: Make sure that you specified the correct host name for the master KDC. If you specified the correct host name, make sure that `kadmind` is running on the master KDC that you specified.

`Credentials cache file permissions incorrect`

Cause: You do not have the appropriate read or write permissions on the credentials cache (`/tmp/krb5cc_`*uid*).

Solution: Make sure that you have read and write permissions on the credentials cache.

`Credentials cache I/O operation failed` *XXX*

Cause: Kerberos had a problem writing to the system's credentials cache (`/tmp/krb5cc_`*uid*).

Solution: Make sure that the credentials cache has not been removed, and that there is space left on the device by using the `df` command.

`Decrypt integrity check failed`

Cause: You might have an invalid ticket.

Solution:

1. Make sure that your credentials are valid. Destroy your tickets with `kdestroy` and create new tickets with `kinit`.
2. Make sure that the target host has a keytab file with the correct version of the service key. Use `kadmin` to view the key version number of the service principal (for example, `host/`*FQDN_hostname*) in the Kerberos database. Also, use `klist -k` on the target host to make sure that it has the same key version number.

`df: cannot statvfs` *filesystem*`: Invalid argument`

Cause: The `df` command cannot access the Kerberized NFS file system, which is currently mounted, to generate its report, because you no longer have the appropriate `root` credentials. Destroying your credentials for a mounted Kerberized file system does not automatically unmount the file system.

Solution: You must create new `root` credentials to access the Kerberized file system. If you no longer require access to the Kerberized file system, unmount the file system.

`failed to obtain credentials cache`

Cause: During `kadmin` initialization, a failure occurred when `kadmin` tried to obtain credentials for the `admin` principal.

Solution: Make sure that you used the correct principal and password when you executed `kadmin`.

`Field is too long for this implementation`

Cause: The message size that was being sent by a Kerberized application was too long. The maximum message size that can be handled by Kerberos is 65535 bytes. In addition, there are limits on individual fields within a protocol message that is sent by Kerberos.

Solution: Make sure that your Kerberized applications are sending valid message sizes.

`GSS-API (or Kerberos) error`

Cause: This message is a generic GSS-API or Kerberos error message and can be caused by several different problems.

Solution: Check the `/etc/krb5/kdc.log` file to find the more specific GSS-API error message that was logged when this error occurred.

`Hostname cannot be canonicalized`

Cause: Kerberos cannot make the host name fully qualified.

Solution: Make sure that the host name is defined in DNS and that the host-name-to-address and address-to-host-name mappings are consistent.

`Illegal cross-realm ticket`

Cause: The ticket sent did not have the correct cross-realms. The realms might not have the correct trust relationships set up.

Solution: Make sure that the realms you are using have the correct trust relationships.

`Improper format of Kerberos configuration file`

Cause: The Kerberos configuration file (`krb5.conf`) has invalid entries.

Solution: Make sure that all the relations in the `krb5.conf` file are followed by the "=" sign and a value. Also, verify that the brackets are present in pairs for each subsection.

`Inappropriate type of checksum in message`

Cause: The message contained an invalid checksum type.

Solution: Check which valid checksum types are specified in the `krb5.conf` and `kdc.conf` files.

`Incorrect net address`
Cause: There was a mismatch in the network address. The network address in the ticket that was being forwarded was different from the network address where the ticket was processed. This message might occur when tickets are being forwarded.

Solution: Make sure that the network addresses are correct. Destroy your tickets with `kdestroy`, and create new tickets with `kinit`.

`Invalid flag for file lock mode`
Cause: An internal Kerberos error occurred.

Solution: Please report a bug.

`Invalid message type specified for encoding`
Cause: Kerberos could not recognize the message type that was sent by the Kerberized application.

Solution: If you are using a Kerberized application that was developed by your site or a vendor, make sure that it is using Kerberos correctly.

`Invalid number of character classes`
Cause: The password that you entered for the principal does not contain enough password classes, as enforced by the principal's policy.

Solution: Make sure that you enter a password with the minimum number of password classes that the policy requires.

`KADM err: Memory allocation failure`
Cause: There is not enough memory to run `kadmin`.

Solution: Free up memory and try running `kadmin` again.

`KDC can't fulfill requested option`
Cause: The KDC did not allow the requested option. A possible problem might be that postdating or forwardable options were being requested, and the KDC did not allow it. Another problem might be that you requested the renewal of a TGT, but you didn't have a renewable TGT.

Solution: Determine if you are requesting an option that either the KDC does not allow or if you are requesting a type of ticket that is not available.

`KDC policy rejects request`
Cause: The KDC policy did not allow the request. For example, the request to the KDC did not have an IP address in its request, or forwarding was requested, but the KDC did not allow it.

Solution: Make sure that you are using `kinit` with the correct options. If necessary, modify the policy that is associated with the principal or change the principal's attributes to allow the request. You can modify the policy or principal by using `kadmin`.

`KDC reply did not match expectations`

Cause: The KDC reply did not contain the expected principal name, or other values in the response were incorrect.

Solution: Make sure that the KDC you are communicating with complies with RFC1510, or that the request you are sending is a Kerberos V5 request, or that the KDC is available.

`Key table entry not found`

Cause: There is no entry for the service principal in the network application server's keytab file.

Solution: Add the appropriate service principal to the server's keytab file so that it can provide the Kerberized service.

`Key version number for principal in key table is incorrect`

Cause: A principal's key version is different in the keytab file and in the Kerberos database. Either a service's key has been changed, or you might be using an old service ticket.

Solution: If a service's key has been changed (for example, by using `kadmin`), you need to extract the new key and store it in the host's keytab file where the service is running.

Alternately, you might be using an old service ticket that has an older key. You might want to run the `kdestroy` command and then the `kinit` command again.

`login: load_modules: can not open module /usr/lib/security/pam_krb5.so.1`

Cause: Either the Kerberos PAM module is missing or it is not a valid executable binary.

Solution: Make sure that the Kerberos PAM module is in the `/usr/lib/security` directory and that it is a valid executable binary. Also, make sure that the `/etc/pam.conf` file contains the correct path to `pam_krb5.so.1`.

`Looping detected inside krb5_get_in_tkt`

Cause: Kerberos made several attempts to get the initial tickets but failed.

Solution: Make sure that at least one KDC is responding to authentication requests.

`Master key does not match database`
Cause: The loaded database dump was not created from a database that contains the master key, which is located in `/var/krb5/.k5.`*REALM*.

Solution: Make sure that the master key in the loaded database dump matches the master key that is located in `/var/krb5/.k5.`*REALM*.

`Matching credential not found`
Cause: The matching credential for your request was not found. Your request requires credentials that are unavailable in the credentials cache.

Solution: Destroy your tickets with `kdestroy` and create new tickets with `kinit`.

`Message out of order`
Cause: Messages that were sent using sequential-order privacy arrived out of order. Some messages might have been lost in transit.

Solution: You should reinitialize the Kerberos session.

`Message stream modified`
Cause: There was a mismatch between the computed checksum and the message checksum. The message might have been modified while in transit, which can indicate a security leak.

Solution: Make sure that the messages are being sent across the network correctly. Since this message can also indicate the possible tampering of messages while they are being sent, destroy your tickets using `kdestroy` and reinitialize the Kerberos services that you are using.

Common SEAM Error Messages (N-Z)

This section provides an alphabetical list (N-Z) of common error messages for the SEAM commands, SEAM daemons, PAM framework, GSS interface, the NFS service, and the Kerberos library.

`No credentials cache file found`
Cause: Kerberos could not find the credentials cache (`/tmp/krb5cc_`*uid*).

Solution: Make sure that the credential file exists and is readable. If it isn't, try performing the `kinit` again.

`Operation requires "`*privilege*`" privilege`
Cause: The `admin` principal that was being used does not have the appropriate privilege configured in the `kadm5.acl` file.

Solution: Use a principal that has the appropriate privileges. Or, configure the principal that was being used to have the appropriate privileges by modifying the `kadm5.acl` file. Usually, a principal with "/admin" as part of its name has the appropriate privileges.

`PAM-KRB5: Kerberos V5 authentication failed: password incorrect`

Cause: Your UNIX password and Kerberos passwords are different. Most non-Kerberized commands, such as `login`, are set up through PAM to automatically authenticate with Kerberos by using the same password that you specified for your UNIX password. If your passwords are different, the Kerberos authentication fails.

Solution: You must enter your Kerberos password when prompted.

`Password is in the password dictionary`

Cause: The password that you entered is in a password dictionary that is being used. Your password is not a good choice for a password.

Solution: Choose a password that has a mix of password classes.

`Permission denied in replay cache code`

Cause: The system's replay cache could not be opened. The server might have been first run under a user ID different than your current user ID.

Solution: Make sure that the replay cache has the appropriate permissions. The replay cache is stored on the host where the Kerberized server application is running (`/usr/tmp/rc_`*service_name*). Instead of changing the permissions on the current replay cache, you can also remove the replay cache before you run the Kerberized server under a different user ID.

`Protocol version mismatch`

Cause: Most likely, a Kerberos V4 request was sent to the KDC. SEAM supports only the Kerberos V5 protocol.

Solution: Make sure that your applications are using the Kerberos V5 protocol.

`Request is a replay`

Cause: The request has already been sent to this server and processed. The tickets might have been stolen, and someone else is trying to reuse the tickets.

Solution: Wait for a few minutes and reissue the request.

`Requested principal and ticket don't match`

Cause: The service principal that you are connecting to and the service ticket that you have do not match.

Solution: Make sure that DNS is functioning properly. If you are using another vendor's software, make sure that the software is using principal names correctly.

`Requested protocol version not supported`

Cause: Most likely, a Kerberos V4 request was sent to the KDC. SEAM supports only the Kerberos V5 protocol.

Solution: Make sure that your applications are using the Kerberos V5 protocol.

`Required parameters in krb5.conf missing while initializing kadmin interface`

Cause: There is a missing parameter (such as the `admin_server` parameter) in the `krb5.conf` file.

Solution: Determine which parameter is missing and add it to the `krb5.conf` file.

`Server rejected authentication (during sendauth exchange)`

Cause: The server that you are trying to communicate with rejected the authentication. Most often this error occurs during Kerberos database propagation. Some common causes might be problems with the `kpropd.acl` file, DNS, or the keytab file.

Solution: If you get this error when you are running applications other than `kprop`, investigate whether the server's keytab file is correct.

`Set gss service nfs@<host> failed. Check nfs service credential.`

Cause: This message is generated by `syslog` after a `share` command has failed with an "invalid argument" message. The most likely cause of this message is that either there is no keytab file or that there is no NFS service principle in the keytab file.

Solution: To isolate the problem, run `klist -k` to see if the keytab file exists and if there is an NFS service principal for the host in the keytab file.

`The ticket isn't for us`

`Ticket/authenticator don't match`

Cause: There was a mismatch between the ticket and authenticator. The principal name in the request might not have matched the service principal's name, because the ticket was being sent with an FQDN name of the principal while the service expected non-FQDN, or vice versa.

Solution: If you get this error when you are running applications other than `kprop`, investigate whether the server's keytab file is correct.

`Ticket expired`

Cause: Your ticket times have expired.

Solution: Destroy your tickets with `kdestroy` and create new tickets with `kinit`.

`Ticket is ineligible for postdating`
Cause: The principal does not allow its tickets to be postdated.

Solution: Modify the principal with `kadmin` to allow postdating.

`Ticket not yet valid`
Cause: The postdated ticket is not valid yet.

Solution: Create new tickets with the correct date, or wait until the current tickets are valid.

`Truncated input file detected`
Cause: The database dump file that was being used in the operation is not a complete dump file.

Solution: Create the dump file again, or use a different database dump file.

`Wrong principal in request`
Cause: There was an invalid principal name in the ticket. This error might indicate a DNS or FQDN problem.

Solution: Make sure that the principal of the service matches the principal in the ticket.

SEAM Troubleshooting

This section provides troubleshooting information for the SEAM software.

Problems Mounting a Kerberized NFS File System

- If mounting a Kerberized NFS file system fails, make sure that the `/var/tmp/rc_nfs` file exists on the NFS server. If the file system is not owned by `root`, remove it and try the mount again.
- If you have a problem accessing a Kerberized NFS file system, make sure that there is an entry for `gssd` in the `inetd.conf` file on your system and the NFS server.
- If you see either the `invalid argument` or `bad directory` error message when you are trying to access a Kerberized NFS file system, the problem might be that you are not using a fully-qualified DNS name when you are trying to mount the NFS file system. The host that is being mounted is not the same as the host name part of the service principal in the server's keytab file.

This problem might also occur if your server has multiple Ethernet interfaces, and you have set up DNS to use a "name per interface" scheme instead of a "multiple address records per host" scheme. For SEAM, you should set up multiple address records per host as follows[1] :

```
my.host.name.    A       1.2.3.4
                 A       1.2.4.4
                 A       1.2.5.4

my-en0.host.name.        A       1.2.3.4
my-en1.host.name.        A       1.2.4.4
my-en2.host.name.        A       1.2.5.4

4.3.2.1          PTR     my.host.name.
4.4.2.1          PTR     my.host.name.
4.5.2.1          PTR     my.host.name.
```

In this example, the setup allows one reference to the different interfaces and allows a single service principal instead of three service principals in the server's keytab file.

Problems Authenticating as `root`

If authentication fails when you try to become superuser on your system and you have already added the `root` principal to your host's keytab file, there are two potential problems to check. First, make sure that the `root` principal in the keytab file has a fully-qualified name as its instance. If it does, check the `/etc/resolv.conf` file to make sure that the system is correctly set up as a DNS client.

1 Ken Hornstein, "Kerberos FAQ," [http://www.nrl.navy.mil/CCS/people./kenh/kerberos-faq.html], accessed 11 December 1998.

CHAPTER 17

Administering Principals and Policies (Tasks)

This chapter provides procedures for managing principals and the policies that are associated with them. This chapter also shows how to manage a host's keytab file.

This chapter should be used by anyone who needs to administer principals and policies. Before you use this chapter, you should be familiar with principals and policies, including any planning considerations. Refer to Chapter 13 and Chapter 14, respectively.

This is a list of the information in this chapter.

Ways to Administer Principals and Policies

The Kerberos database on the master KDC contains all of your realm's Kerberos principals, their passwords, policies, and other administrative information. To create and delete principals, and to modify their attributes, you can use the `kadmin` or `gkadmin` commands.

The `kadmin` command provides an interactive command-line interface that enables you to maintain Kerberos principals, policies, and keytab files. There are two versions of the `kadmin` command:

- `kadmin`, which uses Kerberos authentication to operate securely from anywhere on the network
- `kadmin.local`, which must be run directly on the master KDC

Other than `kadmin` using Kerberos to authenticate the user, the capabilities of the two versions are identical. The local version is necessary to enable you to set up enough of the database so that you can use the remote version.

Also, SEAM provides the SEAM Administration Tool, `gkadmin`, which is an interactive graphical user interface (GUI) that provides essentially the same capabilities as the `kadmin` command. See "SEAM Administration Tool" on page 274 for more information.

SEAM Administration Tool

The SEAM Administration Tool is an interactive graphical user interface (GUI) that enables you to maintain Kerberos principals and policies. This tool provides much the same capabilities as the `kadmin` command. However, this tool does not support the management of keytab files. You must use the `kadmin` command to administer keytab files, which is described in "Administering Keytab Files" on page 303.

Similar to the `kadmin` command, the SEAM Tool uses Kerberos authentication and encrypted RPC to operate securely from anywhere on the network. The SEAM Tool enables you to do the following:

- Create new principals that are based on default values or existing principals
- Create new policies that are based on existing policies
- Add comments for principals
- Set up default values for creating new principals
- Log in as another principal without exiting the tool
- Print or save principal lists and policy lists
- View and search principal lists and policy lists

The SEAM Tool also provides context-sensitive help and general online help.

The following task maps provide pointers to the various tasks that you can do with the SEAM Tool:

- "Administering Principals (Task Map)" on page 279
- "Administering Policies (Task Map)" on page 291

Also, go to "SEAM Tool Panel Descriptions" on page 299 for descriptions of all the principal attributes and policy attributes that you can either specify or view in the SEAM Tool.

Command-Line Equivalents of the SEAM Tool

This section lists the `kadmin` commands that provide the same capabilities as the SEAM Tool. These commands can be used without running an X Window system. Even though most procedures in this chapter use the SEAM Tool, many procedures also provide corresponding examples that use the command-line equivalents.

TABLE 17–1 Command-Line Equivalents of the SEAM Tool

SEAM Tool Procedure	Equivalent `kadmin` Command
View the list of principals	`list_principals` or `get_principals`
View a principal's attributes	`get_principal`
Create a new principal	`add_principal`
Duplicate a principal	No command-line equivalent
Modify a principal	`modify_principal` or `change_password`
Delete a principal	`delete_principal`
Set up defaults for creating new principals	No command-line equivalent
View the list of policies	`list_policies` or `get_policies`
View a policy's attributes	`get_policy`
Create a new policy	`add_policy`
Duplicate a policy	No command-line equivalent
Modify a policy	`modify_policy`
Delete a policy	`delete_policy`

Files Modified by the SEAM Tool

The only file that the SEAM Tool modifies is the `$HOME/.gkadmin` file. This file contains the default values for creating new principals. You can update this file by choosing Properties from the Edit menu.

Print and Online Help Features of the SEAM Tool

The SEAM Tool provides both print features and online help features. From the Print menu, you can send the following to a printer or a file:

- List of available principals on the specified master KDC
- List of available policies on the specified master KDC
- The currently selected principal or the loaded principal

- The currently selected policy or the loaded policy

From the Help menu, you can access context-sensitive help and general help. When you choose Context-Sensitive Help from the Help menu, the Context-Sensitive Help window is displayed and the tool is switched to help mode. In help mode, when you click on any fields, labels, or buttons on the window, help on that item is displayed in the Help window. To switch back to the tool's normal mode, click Dismiss in the Help window.

You can also choose Help Contents, which opens an HTML browser that provides pointers to the general overview and task information that is provided in this chapter.

Working With Large Lists in the SEAM Tool

As your site starts to accumulate a large number of principals and policies, the time it takes the SEAM Tool to load and display the principal and policy lists will become increasingly longer. Thus, your our productivity with the tool will increase. There are several ways to work around this problem.

First, you can completely eliminate the time to load the lists by not having the SEAM Tool load the lists. You can set this option by choosing Properties from the Edit menu, and unchecking the Show Lists field. Of course, when the tool doesn't load the lists, it can't display the lists, and you can no longer use the list panels to select principals or policies. Instead, you must type a principal or policy name in the new Name field that is provided, then select the operation that you want to perform on it. In effect, typing a name is equivalent to selecting an item from the list.

Another way to work with large lists is to cache them. In fact, caching the lists for a limited time is set as the default behavior for the SEAM Tool. The SEAM Tool must still initially load the lists into the cache, but after that, the tool can use the cache rather than retrieve the lists again. This option eliminates the need to keep loading the lists from the server, which is what takes so long.

You can set list caching by choosing Properties from the Edit menu. There are two cache settings. You can choose to cache the list forever, or you can specify a time limit when the tool must reload the lists from the server into the cache.

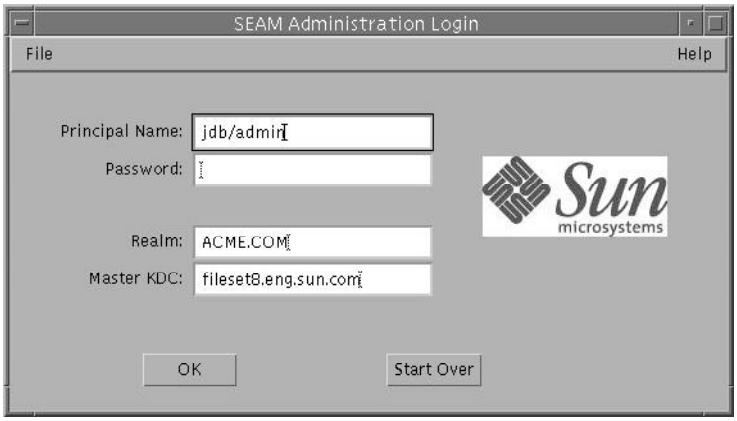
Caching the lists still enables you to use the list panels to select principals and policies, so it doesn't affect how you use the SEAM Tool as the first option does. Also, even though caching doesn't enable you to see the changes of others, you can still see the latest list information based on your changes, since your changes update the lists both on the server and in the cache. And, if you want to update the cache to see other changes and get the lastest copy of the lists, you can use the Refresh menu whenever you want to refresh the cache from the server.

▼ How to Start the SEAM Tool

1. **Start the SEAM Tool by using the `gkadmin` command.**

   ```
   $ /usr/sbin/gkadmin
   ```

   The SEAM Administration Login window is displayed.



2. **If you don't want to use the default values, specify new default values.**

   The window automatically fills in with default values. The default principal name is determined by taking your current identity from the `USER` environment variable and appending `/admin` to it (*username*`/admin`). The default Realm and Master KDC fields are selected from the `/etc/krb5/krb5.conf` file. If you ever want to retrieve the default values, click Start Over.

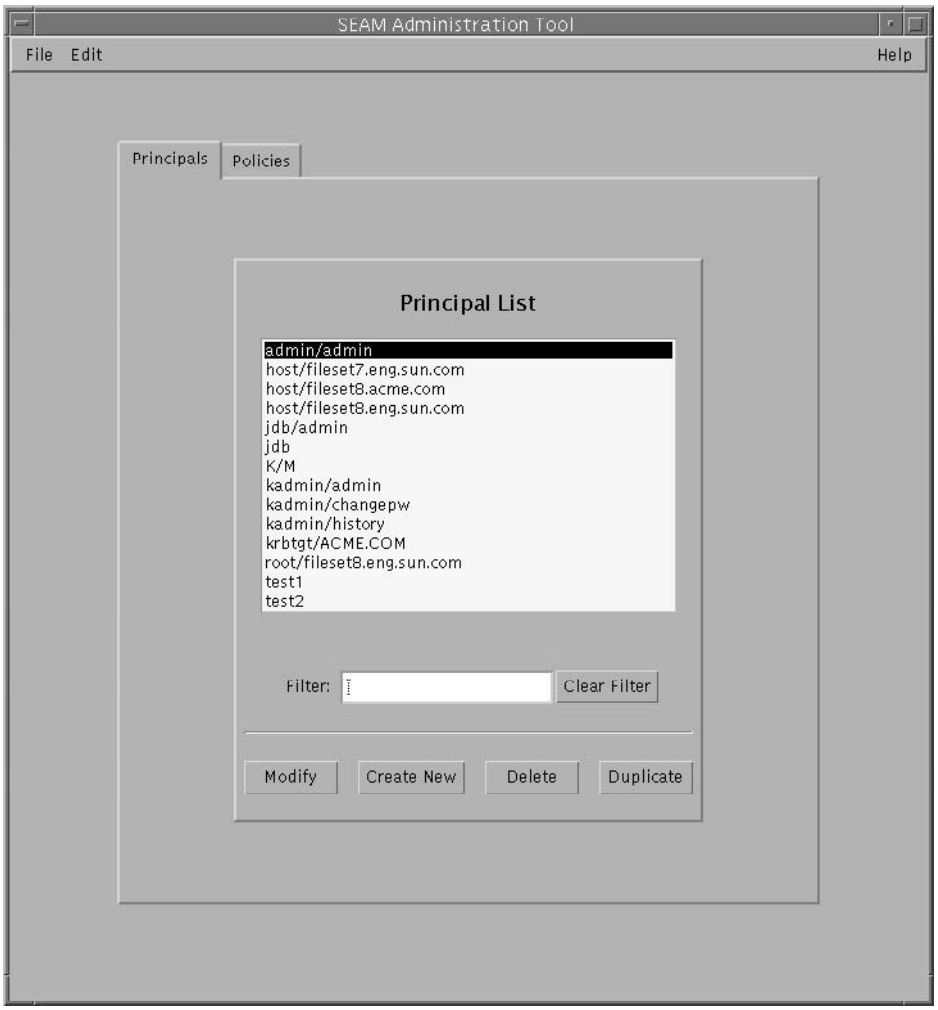
Note – The administration operations that each Principal Name can perform are dictated by the Kerberos ACL file, `/etc/krb5/kadm5.acl`. For information about limited privileges, see "Using the SEAM Tool With Limited Kerberos Administration Privileges" on page 302.

---

3. **Enter a password for the specified principal name.**

4. **Click OK.**

   The following window is displayed.



Administering Principals

This section provides the step-by-step instructions to administer principals with the SEAM Tool. This section also provides examples of equivalent command lines, when available.

Administering Principals (Task Map)

Task	Description	For Instructions
View the list of principals	View the list of principals by clicking the Principals tab.	"How to View the List of Principals" on page 280
View a principal's attributes	View a principal's attributes by selecting the Principal in the Principal List, then clicking the Modify button.	"How to View a Principal's Attributes" on page 282
Create a new principal	Create a new principal by clicking the Create New button in the Principal List panel.	"How to Create a New Principal" on page 284
Duplicate a principal	Duplicate a principal by selecting the principal to duplicate in the Principal List, then clicking the Duplicate button.	"How to Duplicate a Principal" on page 286
Modify a principal	Modify a principal by selecting the principal to modify in the Principal List, then clicking the Modify button. Note that you cannot modify a principal's name. To rename a principal, you must duplicate the principal, specify a new name for it, save it, and then delete the old principal.	"How to Modify a Principal" on page 286
Delete a principal	Delete a principal by selecting the principal to delete in the Principal List, then clicking the Delete button.	"How to Delete a Principal" on page 287
Set up defaults for creating new principals	Set up defaults for creating new principals by choosing Properties from the Edit menu.	"How to Set Up Defaults for Creating New Principals" on page 288
Modify the Kerberos administration privileges (`kadm5.acl` File)	*Command-line only.* The Kerberos administration privileges determine what operations a principal can perform on the Kerberos database, such as add and modify. You need to edit the `/etc/krb5/kadm5.acl` file to modify the Kerberos administration privileges for each principal.	"How to Modify the Kerberos Administration Privileges" on page 289

Automating the Creation of New Principals

Even though the SEAM Tool provides ease-of-use, it doesn't provide a way to automate the creation of new principals. Automation is especially useful if you need to add 10 or even 100 new principals in a short time. However, by using the `kadmin.local` command in a Bourne shell script, you can do just that.

The following shell script line is an example of how automate the creation of new principals:

```
sed -e 's/^\(.*\)$/ank +needchange -pw \1 \1/' < princnames |
        time /usr/sbin/kadmin.local> /dev/null
```

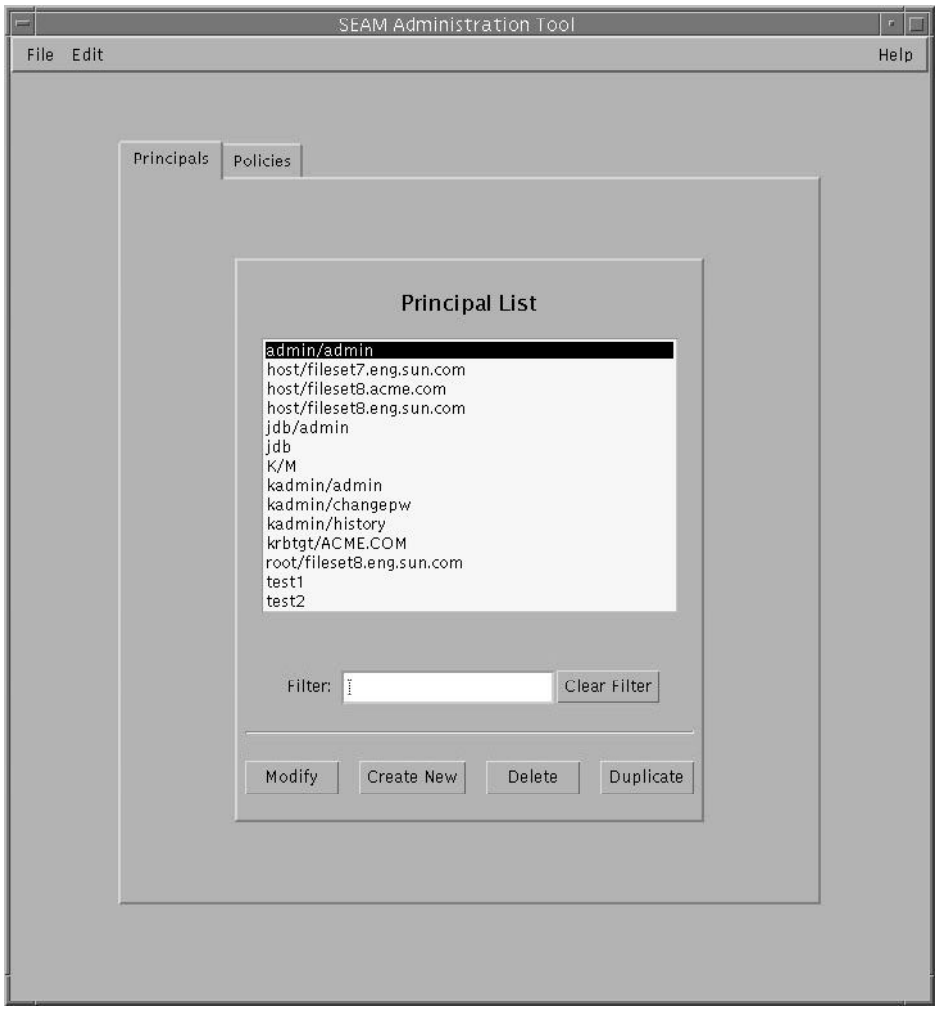
This example is split over two lines readability. The script reads in a file called `princnames` that contains principal names and their passwords, and adds them to the Kerberos database. You would have to create the `princnames` file, which contains a principal name and its password on each line, separated by one or more spaces. The `+needchange` option configures the principal so that the user is prompted for a new password during login with the principal for the first time. This practice helps to ensure that the passwords in the `princnames` file are not a security risk.

You can build more elaborate scripts. For example, your script could use the information in the name service to obtain the list of user names for the principal names. What you do and how you do it is determined by your site needs and your scripting expertise.

▼ How to View the List of Principals

An example of the command-line equivalent follows this procedure.

1. **If necessary, start the SEAM Tool.**
   See "How to Start the SEAM Tool" on page 277 for details.
2. **Click the Principals tab.**
   The list of principals is displayed.



3. **Display a specific principal or a sublist of principals.**

   Type a filter string in the Filter field, and press Return. If the filter succeeds, the list of principals that match the filter is displayed.

   The filter string must consist of one or more characters. Because the filter mechanism is case sensitive, you need to use the appropriate uppercase and lowercase letters for the filter. For example, if you type the filter string `ge`, the filter mechanism displays only the principals with the `ge` string in them (for example, `george` or `edge`).

   If you want to display the entire list of principals, click Clear Filter.

Example—Viewing the List of Principals (Command Line)

In the following example, the `list_principals` command of `kadmin` is used to list all the principals that match `test*`. Wildcards can be used with the `list_principals` command.

```
kadmin: list_principals test*
test1@EXAMPLE.COM
test2@EXAMPLE.COM
kadmin: quit
```

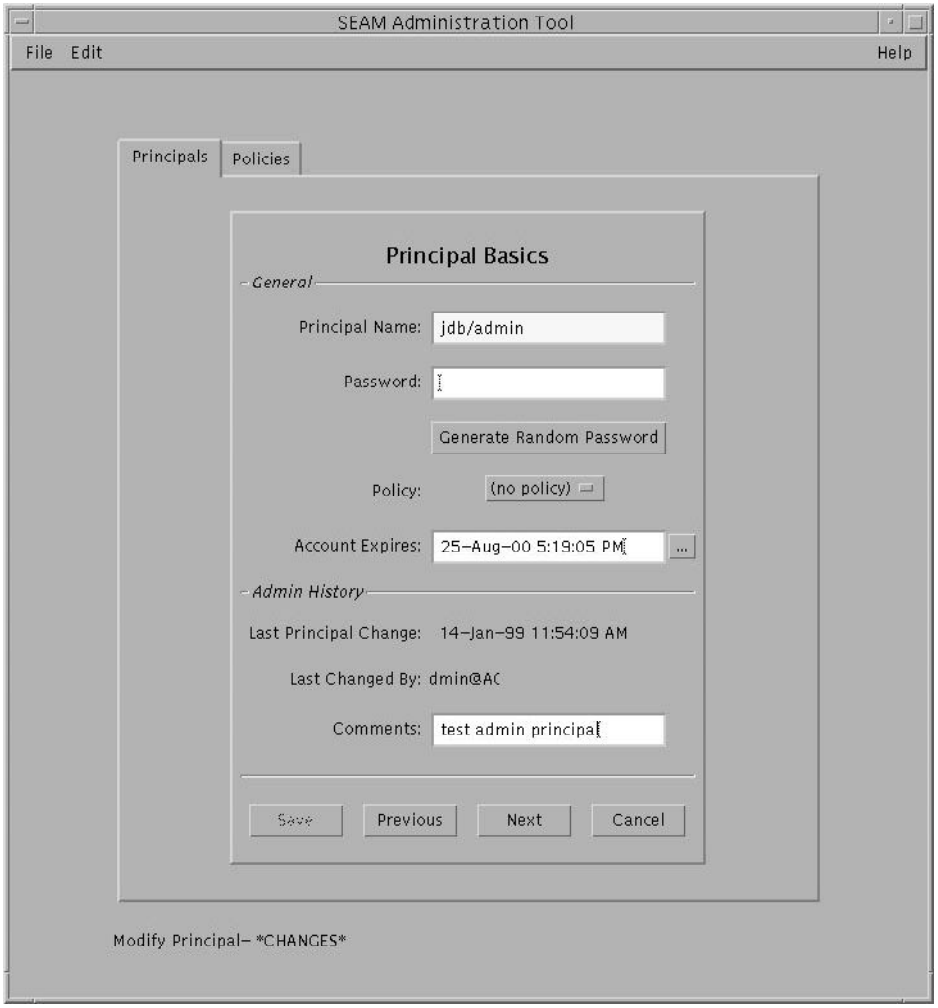
▼ How to View a Principal's Attributes

An example of the command-line equivalent follows this procedure.

1. **If necessary, start the SEAM Tool.**
   See "How to Start the SEAM Tool" on page 277 for details.
2. **Click the Principals tab.**
3. **Select the principal in the list that you want to view, then click Modify.**
   The Principal Basics panel that contains some of the principal's attributes is displayed.
4. **Continue to click Next to view all the principal's attributes.**
   Three windows contain attribute information. Choose Context-Sensitive Help from the Help menu to get information about the various attributes in each window. Or, for all the principal attribute descriptions, go to "SEAM Tool Panel Descriptions" on page 299.
5. **When you are finished viewing, click Cancel.**

Example—Viewing a Principal's Attributes

The following example shows the first window when you are viewing the `jdb/admin` principal.



Example—Viewing a Principal's Attributes (Command Line)

In the following example, the `get_principal` command of `kadmin` is used to view the attributes of the `jdb/admin` principal.

```
kadmin: getprinc jdb/admin
Principal: jdb/admin@EXAMPLE.COM
Expiration date: Fri Aug 25 17:19:05 PDT 2000
Last password change: [never]
Password expiration date: Wed Apr 14 11:53:10 PDT 1999
Maximum ticket life: 1 day 16:00:00
```

```
Maximum renewable life: 1 day 16:00:00
Last modified: Thu Jan 14 11:54:09 PST 1999 (admin/admin@EXAMPLE.COM)
Last successful authentication: [never]
Last failed authentication: [never]
Failed password attempts: 0
Number of keys: 1
Key: vno 1, DES cbc mode with CRC-32, no salt
Attributes: REQUIRES_HW_AUTH
Policy: [none]
kadmin: quit
```

▼ How to Create a New Principal

An example of the command-line equivalent follows this procedure.

1. **If necessary, start the SEAM Tool.**
   See "How to Start the SEAM Tool" on page 277 for details.

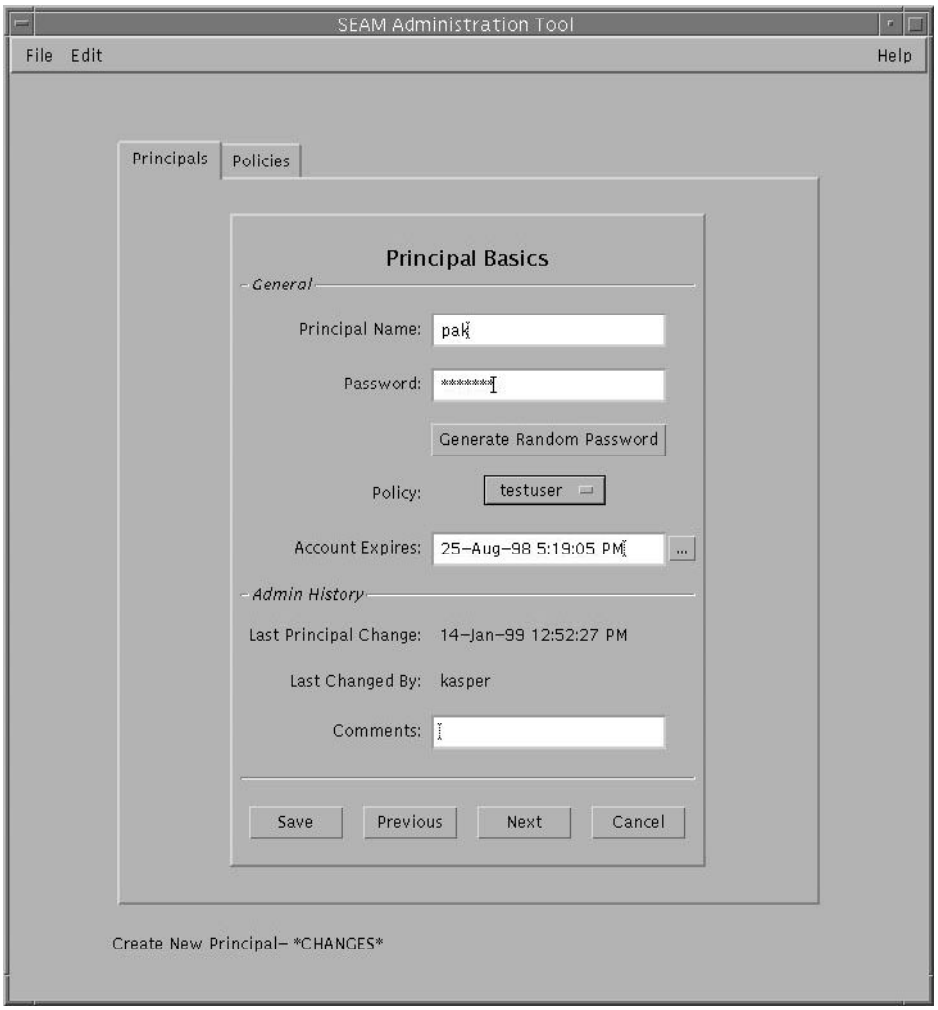
Note – If you are creating a new principal that might need a new policy, you should create the new policy before you create the new principal. Go to "How to Create a New Policy" on page 295.

---

2. **Click the Principals tab.**
3. **Click New.**
   The Principal Basics panel that contains some attributes for a principal is displayed.
4. **Specify a principal name and a password.**
   Both the principal name and password are mandatory.
5. **Specify values for the principal's attributes, and continue to click Next to specify more attributes.**
   Three windows contain attribute information. Choose Context-Sensitive Help from the Help menu to get information about the various attributes in each window. Or, for all the principal attribute descriptions, go to "SEAM Tool Panel Descriptions" on page 299.
6. **Click Save to save the principal, or click Done on the last panel.**
7. **If needed, set up Kerberos administration privileges for the new principal in the `/etc/krb5/kadm5.acl` file.**
   See "How to Modify the Kerberos Administration Privileges" on page 289 for more details.

Example—Creating a New Principal

The following example shows the Principal Basics panel when a new principal called `pak` is created. The policy is set to `testuser`.



Example—Creating a New Principal (Command Line)

In the following example, the `add_principal` command of `kadmin` is used to create a new principal called `pak`. The principal's policy is set to `testuser`.

```
kadmin: add_principal -policy testuser pak
Enter password for principal "pak@EXAMPLE.COM": <type the password>
```

```
Re-enter password for principal "pak@EXAMPLE.COM": <type the password again>
Principal "pak@EXAMPLE.COM" created.
kadmin: quit
```

▼ How to Duplicate a Principal

This procedure explains how to use all or some of the attributes of an existing principal to create a new principal. No command-line equivalent exists for this procedure.

1. **If necessary, start the SEAM Tool.**
   See "How to Start the SEAM Tool" on page 277 for details.
2. **Click the Principals tab.**
3. **Select the principal in the list that you want to duplicate, then click Duplicate.**
   The Principal Basics panel is displayed. All the attributes of the selected principal are duplicated except for the Principal Name and Password fields, which are empty.
4. **Specify a principal name and a password.**
   Both the principal name and the password are mandatory. To make an exact duplicate of the principal you selected, click Save and skip to Step 7.
5. **Specify different values for the principal's attributes, and continue to click Next to specify more attributes.**
   Three windows contain attribute information. Choose Context-Sensitive Help from the Help menu to get information about the various attributes in each window. Or, for all the principal attribute descriptions, go to "SEAM Tool Panel Descriptions" on page 299.
6. **Click Save to save the principal, or click Done on the last panel.**
7. **If needed, set up Kerberos administration privileges for the principal in `/etc/krb5/kadm5.acl` file.**
   See "How to Modify the Kerberos Administration Privileges" on page 289 for more details.

▼ How to Modify a Principal

An example of the command-line equivalent follows this procedure.

1. **If necessary, start the SEAM Tool.**
   See "How to Start the SEAM Tool" on page 277 for details.
2. **Click the Principals tab.**

3. **Select the principal in the list that you want to modify, then click Modify.**

   The Principal Basics panel that contains some of the attributes for the principal is displayed.

4. **Modify the principal's attributes, and continue to click Next to modify more attributes.**

   Three windows contain attribute information. Choose Context-Sensitive Help from the Help menu to get information about the various attributes in each window. Or, for all the principal attribute descriptions, go to "SEAM Tool Panel Descriptions" on page 299.

Note – You cannot modify a principal's name. To rename a principal, you must duplicate the principal, specify a new name for it, save it, and then delete the old principal.

---

5. **Click Save to save the principal, or click Done on the last panel.**

6. **Modify the Kerberos administration privileges for the principal in the `/etc/krb5/kadm5.acl` file.**

   See "How to Modify the Kerberos Administration Privileges" on page 289 for more details.

Example—Modifying a Principal's Password (Command Line)

In the following example, the `change_password` command of `kadmin` is used to modify the password for the `jdb` principal. The `change_password` command does not let you change the password to a password that is in the principal's password history.

```
kadmin: change_password jdb
Enter password for principal "jdb": <type the new password>
Re-enter password for principal "jdb": <type the password again>
Password for "jdb@EXAMPLE.COM" changed.
kadmin: quit
```

To modify other attributes for a principal, you must use the `modify_principal` command of `kadmin`.

▼ How to Delete a Principal

An example of the command-line equivalent follows this procedure.

1. **If necessary, start the SEAM Tool.**

See "How to Start the SEAM Tool" on page 277 for details.

2. **Click the Principals tab.**
3. **Select the principal in the list that you want to delete, then click Delete.**
   After you confirm the deletion, the principal is deleted.
4. **Remove the principal from the Kerberos access control list (ACL) file, `/etc/krb5/kadm5.acl`.**
   See "How to Modify the Kerberos Administration Privileges" on page 289 for more details.

Example—Deleting a Principal (Command Line)

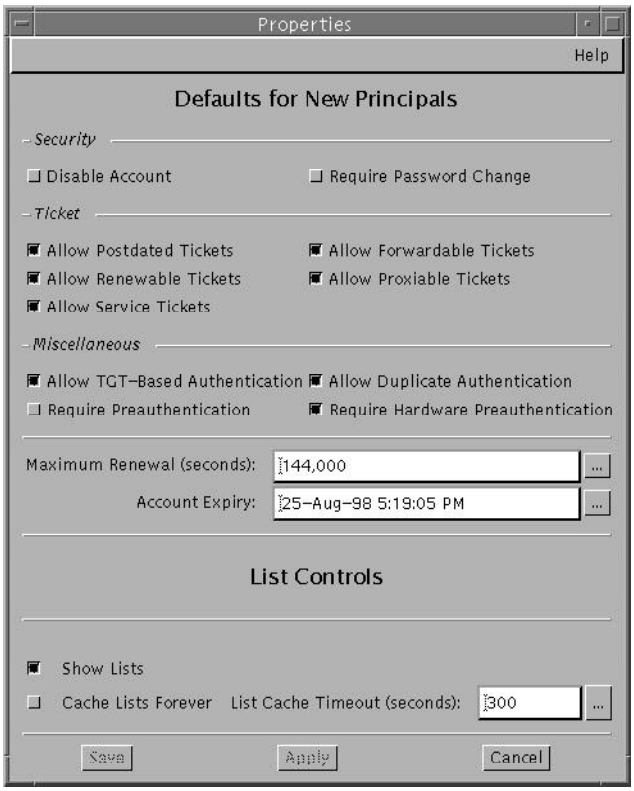
In the following example, the `delete_principal` command of `kadmin` is used to delete the `jdb` principal.

```
kadmin: delete_principal pak
Are you sure you want to delete the principal "pak@EXAMPLE.COM"? (yes/no): yes
Principal "pak@EXAMPLE.COM" deleted.
Make sure that you have removed this principal from all ACLs before reusing.
kadmin: quit
```

▼ How to Set Up Defaults for Creating New Principals

No command-line equivalent exists for this procedure.

1. **If necessary, start the SEAM Tool.**
   See "How to Start the SEAM Tool" on page 277 for details.
2. **Choose Properties from the Edit Menu.**
   The Properties window is displayed.



3. **Select the defaults that you want when you create new principals.**

   Choose Context-Sensitive Help from the Help menu for information about the various attributes in each window.

4. **Click Save.**

▼ How to Modify the Kerberos Administration Privileges

Even though your site probably has many user principals, you usually want only a few users to be able to administer the Kerberos database. Privileges to administer the Kerberos database are determined by the Kerberos access control list (ACL) file, `kadm5.acl`. The `kadm5.acl` file enables you to allow or disallow privileges for individual principals. Or, you can use the '*' wildcard in the principal name to specify privileges for groups of principals.

1. **Become superuser on the master KDC.**

2. **Edit the `/etc/krb5/kadm5.acl` file.**

   An entry in the `kadm5.acl` file must have the following format:

   principal *privileges* [*principal-target*]

   principal — Specifies the principal to which the privileges are granted. Any part of the principal name can include the '*' wildcard, which is useful for providing the same privileges for a group of principals. For example, if you want to specify all principals with the `admin` instance, you would use `*/admin@`*realm*.

   Note that a common use of an `admin` instance is to grant separate privileges (such as administration access to the Kerberos database) to a separate Kerberos principal. For example, the user `jdb` might have a principal for his administrative use, called `jdb/admin`. This way, the user `jdb` obtains `jdb/admin` tickets only when he or she actually needs to use those privileges.

   privileges — Specifies which operations can or cannot be performed by the principal. This field consists of a string of one or more of the following list of characters or their uppercase counterparts. If the character is uppercase (or not specified), then the operation is disallowed. If the character is lowercase, then the operation is permitted.

`a`	[Dis]allows the addition of principals or policies.
`d`	[Dis]allows the deletion of principals or policies.
`m`	[Dis]allows the modification of principals or polices.
`c`	[Dis]allows the changing of passwords for principals.
`i`	[Dis]allows inquiries to the Kerberos database.
`l`	[Dis]allows the listing of principals or policies in the Kerberos database.
`x` or `*`	Allows all privileges (`admcil`).

   principal-target — When a principal is specified in this field, the *privileges* apply to *principal* only when the *principal* operates on the *principal_target*. Any part of the principal name can include the '*' wildcard, which is useful to group principals.

Example—Modifying the Kerberos Administration Privileges

The following entry in the `kadm5.acl` file gives any principal in the `EXAMPLE.COM` realm with the `admin` instance all the privileges on the Kerberos database.

```
*/admin@EXAMPLE.COM *
```

The following entry in the `kadm5.acl` file gives the `jdb@EXAMPLE.COM` principal the privilege to add, list, and inquire about any principal that has the `root` instance.

```
jdb@EXAMPLE.COM ali */root@EXAMPLE.COM
```

Administering Policies

This section provides step-by-step instructions to administer policies with the SEAM Tool. This section also provides examples of equivalent command lines, when available.

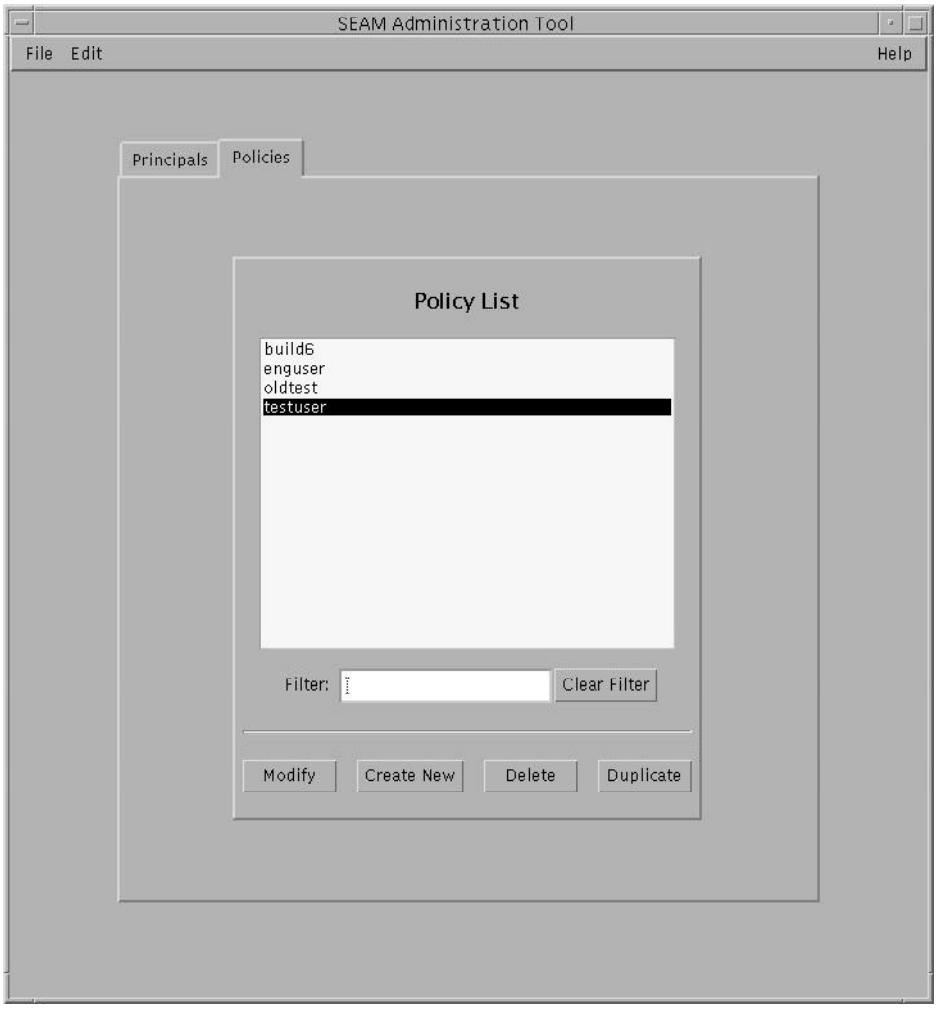
Administering Policies (Task Map)

Task	Description	For Instructions
View the list of policies	View the list of policies by clicking the Policies tab.	"How to View the List of Policies" on page 292
View a policy's attributes	View a policy's attributes by selecting the policy in the Policy List, then clicking the Modify button.	"How to View a Policy's Attributes" on page 293
Create a new policy	Create a new policy by clicking the Create New button in the Policy List panel.	"How to Create a New Policy" on page 295
Duplicate a policy	Duplicate a policy by selecting the policy to duplicate in the Policy List, then clicking the Duplicate button.	"How to Duplicate a Policy" on page 297
Modify a policy	Modify a policy by selecting the policy to modify in the Policy List, then clicking the Modify button. Note that you cannot modify a policy's name. To rename a policy, you must duplicate the policy, specify a new name for it, save it, and then delete the old policy.	"How to Modify a Policy" on page 297
Delete a policy	Delete a policy by selecting the policy to delete in the Policy List, then clicking the Delete button.	"How to Delete a Policy" on page 298

▼ How to View the List of Policies

An example of the command-line equivalent follows this procedure.

1. **If necessary, start the SEAM Tool.**
   See "How to Start the SEAM Tool" on page 277 for details.
2. **Click the Policies tab.**
   The list of policies is displayed.



3. **Display a specific policy or a sublist of policies.**

Type a filter string in the Filter field, and press Return. If the filter succeeds, the list of policies that match the filter is displayed.

The filter string must consist of one or more characters. Because the filter mechanism is case sensitive, you need to use the appropriate uppercase and lowercase letters for the filter. For example, if you type the filter string `ge`, the filter mechanism displays only the policies with the `ge` string in them (for example, `george` or `edge`).

If you want to display the entire list of policies, click Clear Filter.

Example—Viewing the List of Policies (Command Line)

In the following example, the `list_policies` command of `kadmin` is used to list all the policies that match `*user*`. Wildcards can be used with the `list_policies` command.

```
kadmin: list_policies *user*
testuser
enguser
kadmin: quit
```

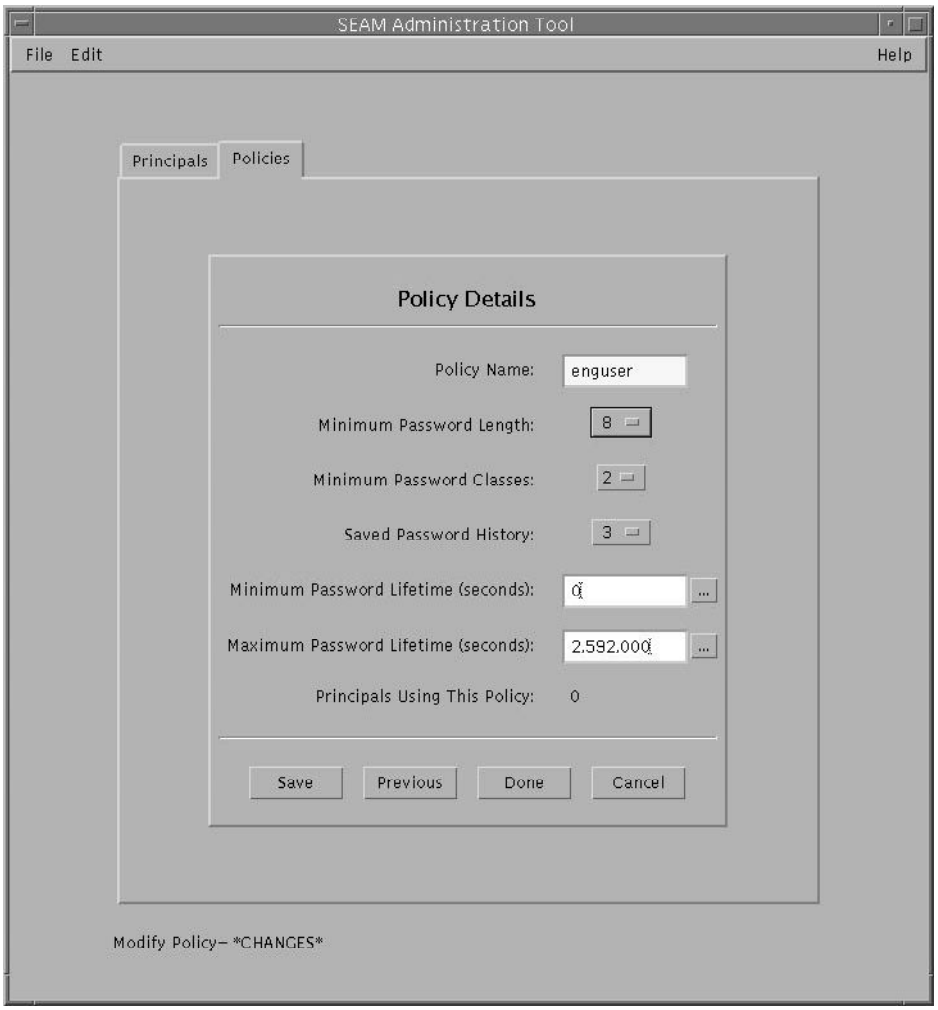
▼ How to View a Policy's Attributes

An example of the command-line equivalent follows this procedure.

1. **If necessary, start the SEAM Tool.**
   See "How to Start the SEAM Tool" on page 277 for details.
2. **Click the Policies tab.**
3. **Select the policy in the list that you want to view, then click Modify.**
   The Policy Details panel is displayed.
4. **When you are finished viewing, click Cancel.**

Example—Viewing a Policy's Attributes

The following example shows the Policy Details panel when you are viewing the `test` policy.



Example—Viewing a Policy's Attributes (Command Line)

In the following example, the `get_policy` command of `kadmin` is used to view the attributes of the `enguser` policy.

```
kadmin: get_policy enguser
Policy: enguser
Maximum password life: 2592000
Minimum password life: 0
Minimum password length: 8
Minimum number of password character classes: 2
Number of old keys kept: 3
Reference count: 0
```

```
kadmin: quit
```

The reference count is the number of principals that use this policy.

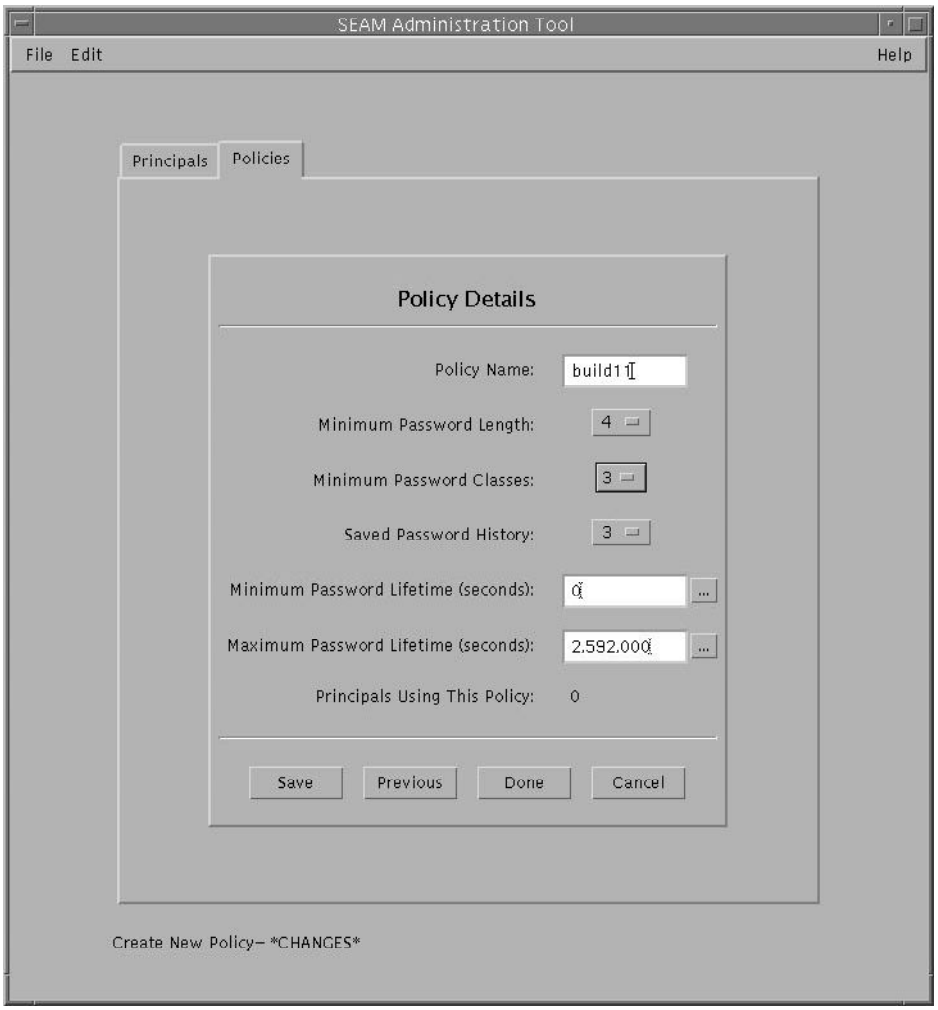
▼ How to Create a New Policy

An example of the command-line equivalent follows this procedure.

1. **If necessary, start the SEAM Tool.**
   See "How to Start the SEAM Tool" on page 277 for details.
2. **Click the Policies tab.**
3. **Click New.**
   The Policy Details panel is displayed.
4. **Specify a name for the policy in the Policy Name field.**
   The policy name is mandatory.
5. **Specify values for the policy's attributes.**
   Choose Context-Sensitive Help from the Help menu for information about the various attributes in this window. Or, go to Table 17–5 for all the policy attribute descriptions.
6. **Click Save to save the policy, or click Done.**

Example—Creating a New Policy

In the following example, a new policy called `build11` is created. The Minimum Password Classes is set to 3.



Example—Creating a New Policy (Command Line)

In the following example, the `add_policy` command of `kadmin` is used to create the `build11` policy. This policy requires at least 3 character classes in a password.

```
$ kadmin
kadmin: add_policy -minclasses 3 build11
kadmin: quit
```

▼ How to Duplicate a Policy

This procedure explains how to use all or some of the attributes of an existing policy to create a new policy. No command-line equivalent exists for this procedure.

1. **If necessary, start the SEAM Tool.**
   See "How to Start the SEAM Tool" on page 277 for details.
2. **Click the Policies tab.**
3. **Select the policy in the list that you want to duplicate, then click Duplicate.**
   The Policy Details panel is displayed. All the attributes of the selected policy are duplicated, except for the Policy Name field, which is empty.
4. **Specify a name for the duplicated policy in the Policy Name field.**
   The policy name is mandatory. To make an exact duplicate of the policy you selected, click Save and skip to Step 6.
5. **Specify different values for the policy's attributes.**
   Choose Context-Sensitive Help from the Help menu for information about the various attributes in this window. Or, go to Table 17–5 for all the policy attribute descriptions.
6. **Click Save to save the policy, or click Done.**

▼ How to Modify a Policy

An example of the command-line equivalent follows this procedure.

1. **If necessary, start the SEAM Tool.**
   See "How to Start the SEAM Tool" on page 277 for details.
2. **Click the Policies tab.**
3. **Select the policy in the list that you want to modify, then click Modify.**
   The Policy Details panel is displayed.
4. **Modify the policy's attributes.**
   Choose Context-Sensitive Help from the Help menu for information about the various attributes in this window. Or, go to Table 17–5 for all the policy attribute descriptions.

   Note – You cannot modify a policy's name. To rename a policy, you must duplicate the policy, specify a new name for it, save it, and then delete the old policy.

   ---

5. **Click Save to save the policy, or click Done.**

Example—Modifying a Policy (Command Line)

In the following example, the `modify_policy` command of `kadmin` is used to modify the minimum length of a password to five characters for the `build11` policy.

```
$ kadmin
kadmin: modify_policy -minlength 5 build11
kadmin: quit
```

▼ How to Delete a Policy

An example of the command-line equivalent follows this procedure.

1. **If necessary, start the SEAM Tool.**
   See "How to Start the SEAM Tool" on page 277 for details.
2. **Click the Policies tab.**

Note – Before you delete a policy, you must cancel the policy from all principals that are currently using it. To do so, you need to modify the principals' Policy attribute. The policy cannot be deleted if any principal is using it.

---

3. **Select the policy in the list that you want to delete, then click Delete.**
   After you confirm the deletion, the policy is deleted.

Example—Deleting a Policy (Command Line)

In the following example, the `delete_policy` command of the `kadmin` command is used to delete the `build11` policy.

```
kadmin: delete_policy build11
Are you sure you want to delete the policy "build11"? (yes/no): yes
kadmin: quit
```

Before you delete a policy, you must cancel the policy from all principals that are currently using it. To do so, you need to use the `modify_principal -policy` command of `kadmin` on the affected principals. The `delete_policy` command fails if the policy is in use by a principal.

SEAM Tool Reference

This section provides reference information for the SEAM Tool.

SEAM Tool Panel Descriptions

This section provides descriptions for each principal and policy attribute that you can either specify or view in the SEAM Tool. The attributes are organized by the panel in which they are displayed.

TABLE 17–2 Attributes for the Principal Basics Panel

Attribute	Description
Principal Name	The name of the principal (the *primary/instance* part of a fully-qualified principal name). A principal is a unique identity to which the KDC can assign tickets. If you are modifying a principal, you cannot edit a principal's name.
Password	The password for the principal. You can use the Generate Random Password button to create a random password for the principal.
Policy	A menu of available policies for the principal.
Account Expires	The date and time on which the principal's account expires. When the account expires, the principal can no longer get a ticket-granting ticket (TGT) and might be unable to log in.
Last Principal Change	The date on which information for the principal was last modified. (Read-only)
Last Changed By	The name of the principal that last modified the account for this principal. (Read-only)
Comments	Comments that are related to the principal (for example, "Temporary Account").

TABLE 17–3 Attributes for the Principal Details Panel

Attribute	Description
Last Success	The date and time when the principal last logged in successfully. (Read-only)
Last Failure	The date and time when the last login failure for the principal occurred. (Read-only)
Failure Count	The number of times that there has been a login failure for the principal. (Read-only)
Last Password Change	The date and time when the principal's password was last changed. (Read-only)

TABLE 17–3 Attributes for the Principal Details Panel *(Continued)*

Attribute	Description
Password Expires	The date and time when the principal's current password expires.
Key Version	The key version number for the principal. This attribute is normally changed only when a password has been compromised.
Maximum Lifetime (seconds)	The maximum length of time for which a ticket can be granted for the principal (without renewal).
Maximum Renewal (seconds)	The maximum length of time for which an existing ticket can be renewed for the principal.

TABLE 17–4 Attributes of the Principal Flags Panel

Attribute (Radio Buttons)	Description
Disable Account	When checked, prevents the principal from logging in. This attribute provides an easy way to temporarily freeze a principal account.
Require Password Change	When checked, expires the principal's current password, which forces the user to use the `kpasswd` command to create a new password. This attribute is useful if there is a security breach, and you need to make sure that old passwords are replaced.
Allow Postdated Tickets	When checked, allows the principal to obtain postdated tickets. For example, you might need to use postdated tickets for `cron` jobs that must run after hours, but you cannot obtain tickets in advance because of short ticket lifetimes.
Allow Forwardable Tickets	When checked, allows the principal to obtain forwardable tickets. Forwardable tickets are tickets that are forwarded to the remote host to provide a single-sign-on session. For example, if you are using forwardable tickets and you authenticate yourself through `ftp` or `rsh`, then other services, such as NFS services, are available without your being prompted for another password.
Allow Renewable Tickets	When checked, allows the principal to obtain renewable tickets. A principal can automatically extend the expiration date or time of a ticket that is renewable (rather than having to get a new ticket after the first ticket expires). Currently, the NFS service is the ticket service that can renew tickets.
Allow Proxiable Tickets	When checked, allows the principal to obtain proxiable tickets. A proxiable ticket is a ticket that can be used by a service on behalf of a client to perform an operation for the client. With a proxiable ticket, a service can take on the identity of a client and obtain a ticket for another service, but the service cannot obtain a ticket-granting ticket.

TABLE 17–4 Attributes of the Principal Flags Panel *(Continued)*

Attribute (Radio Buttons)	Description
Allow Service Tickets	When checked, allows service tickets to be issued for the principal. You should not allow service tickets to be issued for the `kadmin/`*hostname* and `changepw/`*hostname* principals. This practice ensures that these principals can only update the KDC database.
Allow TGT-Based Authentication	When checked, allows the service principal to provide services to another principal. More specifically, this attribute allows the KDC to issue a service ticket for the service principal. This attribute is valid only for service principals. When unchecked, service tickets cannot be issued for the service principal.
Allow Duplicate Authentication	When checked, allows the user principal to obtain service tickets for other user principals. This attribute is valid only for user principals. When unchecked, the user principal can still obtain service tickets for service principals, but not for other user principals.
Required Preauthentication	When checked, the KDC will not send a requested ticket-granting ticket (TGT) to the principal until the KDC can authenticate (through software) that the principal is really the principal that is requesting the TGT. This preauthentication is usually done through an extra password, for example, from a DES card. When unchecked, the KDC does not need to preauthenticate the principal before the KDC sends a requested TGT to the principal.
Required Hardware Authentication	When checked, the KDC will not send a requested ticket-granting ticket (TGT) to the principal until the KDC can authenticate (through hardware) that it is really the principal that is requesting the TGT. Hardware preauthentication can occur, for example, on a Java ring reader. When unchecked, the KDC does not need to preauthenticate the principal before the KDC sends a requested TGT to the principal.

TABLE 17–5 Attributes for the Policy Basics Pane

Attribute	Description
Policy Name	The name of the policy. A policy is a set of rules that govern a principal's password and tickets. If you are modifying a policy, you cannot edit a policy's name.
Minimum Password Length	The minimum length for the principal's password.

TABLE 17–5 Attributes for the Policy Basics Pane *(Continued)*

Attribute	Description
Minimum Password Classes	The minimum number of different character types that are required in the principal's password. For example, a minimum classes value of 2 means that the password must have at least two different character types, such as letters and numbers (hi2mom). A value of 3 means that the password must have at least three different character types, such as letters, numbers, and punctuation (hi2mom!). And so on. A value of 1 sets no restriction on the number of password character types.
Saved Password History	The number of previous passwords that have been used by the principal, and a list of the previous passwords that cannot be reused.
Minimum Password Lifetime (seconds)	The minimum time that the password must be used before it can be changed.
Maximum Password Lifetime (seconds)	The maximum time that the password can be used before it must be changed.
Principals Using This Policy	The number of principals to which this policy currently applies. (Read-only)

Using the SEAM Tool With Limited Kerberos Administration Privileges

All features of the SEAM Administration Tool are available if your `admin` principal has all the privileges to administer the Kerberos database. But it is possible to have limited privileges, such as being allowed to view only the list of principals or to change a principal's password. With limited Kerberos administration privileges, you can still use the SEAM Tool. However, various parts of the SEAM Tool will change based on the Kerberos administration privileges that you do not have. Table 17–6 shows how the SEAM Tool changes based on your Kerberos administration privileges.

The most visual change to the SEAM Tool occurs when you don't have the list privilege. Without the list privilege, the List panels do not display the list of principals and polices for you to manipulate. Instead, you must use the Name field in the List panels to specify a principal or policy that you want to manipulate.

If you login to the SEAM Tool, and you do not have sufficient privileges to perform tasks with it, the following message displays and you are sent back to the SEAM Administration Login window:

```
Insufficient privileges to use gkadmin: ADMCIL. Please try using another principal.
```

To change the privileges for a principal to administer the Kerberos database, go to "How to Modify the Kerberos Administration Privileges" on page 289.

TABLE 17–6 Using SEAM Tool With Limited Kerberos Administration Privileges

Disallowed Privilege	Change the SEAM Tool
`a` (add)	The Create New and Duplicate buttons are unavailable in the Principal List and Policy List panels. Without the add privilege, you cannot create new principals or policies or duplicate them.
`d` (delete)	The Delete button is unavailable in the Principal List and Policy List panels. Without the delete privilege, you cannot delete principal or policies.
`m` (modify)	The Modify button is unavailable in the Principal List and Policy List panels. Without the modify privilege, you cannot modify principal or policies. Also, with the Modify button unavailable, you cannot modify a principal's password, even if you have the change password privilege.
`c` (change password)	The Password field in the Principal Basics panel is read-only and cannot be changed. Without the change password privilege, you cannot modify a principal's password. Note that even if you have the change password privilege, you must also have the modify privilege to change a principal's password.
`i` (inquiry to database)	The Modify and Duplicate buttons are unavailable in the Principal List and Policy List panels. Without the inquiry privilege, you cannot modify or duplicate a principal or policy. Also, with the Modify button unavailable, you cannot modify a principal's password, even if you have the change password privilege.
`l` (list)	The list of principals and policies in the List panels are unavailable. Without the list privilege, you must use the Name field in the List panels to specify the principal or policy that you want to manipulate.

Administering Keytab Files

Every host that provides a service must have a local file, called a *keytab* (short for key table). The keytab contains the principal for the appropriate service, called a *service key*. A service key is used by a service to authenticate itself to the KDC and is known only by Kerberos and the service itself. For example, if you have a Kerberized NFS server, that server must have a keytab file that contains its `nfs` service principal.

To add a service key to a keytab file, you add the appropriate service principal to a host's keytab file by using the `ktadd` command of `kadmin`. Because you are adding a service principal to a keytab file, the principal must already exist in the Kerberos database so that `kadmin` can verify its existence. On the master KDC, the keytab file is located at `/etc/krb5/kadm5.keytab`, by default. On application servers that provide Kerberized services, the keytab file is located at `/etc/krb5/krb5.keytab`, by default.

A keytab is analogous to a user's password. Just as it is important for users to protect their passwords, it is equally important for application servers to protect their keytab files. You should always store keytab files on a local disk, and make them readable only by the `root` user. Also, you should never send a keytab file over an unsecured network.

There is also a special instance to add a `root` principal to a host's keytab file. If you want a user on the SEAM client to mount Kerberized NFS file systems that use Kerberos authentication automatically, you must add the client's `root` principal to the client's keytab file. Otherwise, users must use the `kinit` command as root to obtain credentials for the client's `root` principal whenever they want to mount a Kerberized NFS file system, even when they are using the automounter. See "Setting Up `Root` Authentication to Mount NFS File Systems" on page 247 for detailed information.

Note – When you set up a master KDC, you need to add the `kadmind` and `changepw` principals to the `kadm5.keytab` file. This step enables the KDC to decrypt administrators' Kerberos tickets to determine whether it should give the administrators access to the database.

Another command that you can use to administer keytab files is the `ktutil` command. `ktutil` is an interactive command that enables you to manage a local host's keytab file without having Kerberos administration privileges, because `ktutil` doesn't interact with the Kerberos database as `kadmin` does. So, after a principal is added to a keytab file, you can use `ktutil` to view the keylist in a keytab file or to temporarily disable authentication for a service.

Administering Keytabs Task Map

Task	Description	For Instructions
Add a service principal to a keytab file	Use the `ktadd` command of `kadmin` to add a service principal to a keytab file.	"How to Add a Service Principal to a Keytab File" on page 305

Task	Description	For Instructions
Remove a service principal from a keytab file	Use the `ktremove` command of `kadmin` to remove a service from a keytab file.	"How to Remove a Service Principal From a Keytab File" on page 306
Display the keylist (Principals) in a keytab file	Use the `ktutil` command to display the keylist in a keytab file.	"How to Display the Keylist (Principals) in a Keytab File" on page 307
Temporarily disable authentication for a service on a host	This procedure is a quick way to temporarily disable authentication for a service on a host without having to have `kadmin` privileges. Before you use `ktutil` to delete the service principal from the server's keytab file, copy the original keytab file to a temporary location. When you want to enable the service again, copy the original keytab file back to its proper location.	"How to Temporarily Disable Authentication for a Service on a Host" on page 308

▼ How to Add a Service Principal to a Keytab File

1. **Make sure that the principal already exists in the Kerberos database.**
   See "How to View the List of Principals" on page 280 for more information.
2. **Become superuser on the host that needs a principal added to its keytab file.**
3. **Start the `kadmin` command.**

   ```
   # /usr/sbin/kadmin
   ```

4. **Add a principal to a keytab file by using the `ktadd` command.**

   ```
   kadmin: ktadd [-k keytab] [-q] [principal | -glob principal-exp]
   ```

`-k` *keytab*	Specifies the keytab file. By default, `/etc/krb5/krb5.keytab` is used.
`-q`	Displays less verbose information.
principal	Specifies the principal to be added to the keytab file. You can add the following service principals: `host`, `root`, `nfs`, and `ftp`.
`-glob` *principal-exp*	Specifies the principal expressions. All principals that match the principal.are added to the keytab file. The rules for principal expression are the same as for the `list_principals` command of `kadmin`.

5. **Quit the `kadmin` command.**

```
kadmin: quit
```

Example—Adding a Service Principal to a Keytab File

In the following example, the `kadmin/admin` and `kadmin/changepw` principals are added to a master KDC's keytab file. For this example, the keytab file must be the file that is specified in the `kdc.conf` file.

```
kdc1 # /usr/sbin/kadmin.local
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab kadmin/admin kadmin/changepw
Entry for principal kadmin/admin@EXAMPLE.COM with kvno 3, encryption type DES-CBC-CRC
  added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw@EXAMPLE.COM with kvno 3, encryption type DES-CBC-CRC
  added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local: quit
```

In the following example, `denver`'s `host` principal is added to `denver`'s keytab file, so that the KDC can authenticate `denver`'s network services.

```
denver # /usr/sbin/kadmin
kadmin: ktadd host/denver@example.com@EXAMPLE.COM
kadmin: Entry for principal host/denver@example.com@EXAMPLE.COM with kvno 2,
  encryption type DES-CBC-CRC added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin: quit
```

▼ How to Remove a Service Principal From a Keytab File

1. **Become superuser on the host with a service principal that must be removed from its keytab file.**
2. **Start the `kadmin` command.**

   ```
   # /usr/sbin/kadmin
   ```

3. **(Optional) To display the current list of principals (keys) in the keytab file, use the `ktutil` command.**

   See "How to Display the Keylist (Principals) in a Keytab File" on page 307 for detailed instructions.

4. **Remove a principal from the keytab file by using the `ktremove` command.**

   ```
   kadmin: ktremove [-k keytab] [-q] principal [kvno | all | old ]
   ```

   `-k` *keytab* — Specifies the keytab file. By default, `/etc/krb5/krb5.keytab` is used.

`-q`	Displays less verbose information.
principal	Specifies the principal to be removed from the keytab file.
kvno	Removes all entries for the specified principal whose key version number matches *kvno*.
`all`	Removes all entries for the specified principal.
`old`	Removes all entries for the specified principal, except those principals with the highest key version number.

5. **Quit the `kadmin` command.**

```
kadmin: quit
```

Example—Removing a Service Principal From a Keytab

In the following example, `denver`'s `host` principal is removed from `denver`'s keytab file.

```
denver # /usr/sbin/kadmin
kadmin: ktremove host/denver.example.com@EXAMPLE.COM
kadmin: Entry for principal host/denver.example.com@EXAMPLE.COM with kvno 3
  removed from keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin: quit
```

▼ How to Display the Keylist (Principals) in a Keytab File

1. **Become superuser on the host with the keytab file.**

Note – Although you can create keytab files that are owned by other users, the default location for the keytab file requires `root` ownership.

---

2. **Start the `ktutil` command.**

```
# /usr/bin/ktutil
```

3. **Read the keytab file into the keylist buffer by using the `read_kt` command.**

```
ktutil: read_kt keytab
```

4. **Display the keylist buffer by using the `list` command.**

```
ktutil: list
```

The current keylist buffer is displayed.

5. **Quit the `ktutil` command.**

```
ktutil: quit
```

Example—Displaying the Keylist (Principals) in a Keytab File

The following example displays the keylist in the `/etc/krb5/krb5.keytab` file on the `denver` host.

```
denver # /usr/bin/ktutil
    ktutil: read_kt /etc/krb5/krb5.keytab
    ktutil: list
slot KVNO Principal
---- ---- ---------------------------------------
   1    5 host/denver@EXAMPLE.COM
    ktutil: quit
```

▼ How to Temporarily Disable Authentication for a Service on a Host

At times, you might need to temporarily disable the authentication mechanism for a service, such as `rlogin` or `ftp`, on a network application server. For example, you might want to stop users from logging in to a system while you are performing maintenance procedures. The `ktutil` command enables you to accomplish this task by removing the service principal from the server's keytab file, without requiring `kadmin` privileges. To enable authentication again, you just need to copy the original keytab file that you saved back to its original location.

Note – By default, most services are set up to require authentication. If a service is not set up to require authentication, then the service will still work, even if you disable authentication for the service.

---

1. **Become superuser on the host with the keytab file.**

Note – Although you can create keytab files that are owned by other users, the default location for the keytab file requires `root` ownership.

---

2. **Save the current keytab file to a temporary file.**
3. **Start the `ktutil` command.**

```
# /usr/bin/ktutil
```

4. **Read the keytab file into the keylist buffer by using the `read_kt` command.**

   ```
   ktutil: read_kt keytab
   ```

5. **Display the keylist buffer by using the `list` command.**

   ```
   ktutil: list
   ```

   The current keylist buffer is displayed. Note the slot number for the service that you want to disable.

6. **To temporarily disable a host's service, remove the specific service principal from the keylist buffer by using the `delete_entry` command.**

   ```
   ktutil: delete_entry slot-number
   ```

   In this example, *slot-number* specifies the slot number of the service principal to be deleted, which is displayed by the `list` command.

7. **Write the keylist buffer to the keytab file by using the `write_kt` command.**

   ```
   ktutil: write_kt keytab
   ```

8. **Quit the `ktutil` command.**

   ```
   ktutil: quit
   ```

9. **When you want to re-enable the service, copy the temporary (original) keytab file back to its original location.**

Example—Temporarily Disabling a Service on a Host

In the following example, the `host` service on the `denver` host is temporarily disabled. To enable the host service back on `denver`, you would copy the `krb5.keytab.temp` file to the `/etc/krb5/krb5.keytab` file.

```
denver # cp /etc/krb5/krb5.keytab /etc/krb5/krb5.keytab.temp
denver # /usr/bin/ktutil
    ktutil:read_kt /etc/krb5/krb5.keytab
    ktutil:list
slot KVNO Principal
---- ---- ---------------------------------------
   1    8 root/denver@EXAMPLE.COM
   2    5 host/denver@EXAMPLE.COM
    ktutil:delete_entry 2
    ktutil:list
slot KVNO Principal
---- ---- ---------------------------------------
   1    8 root/denver@EXAMPLE.COM
    ktutil:write_kt /etc/krb5/krb5.keytab
    ktutil: quit
```

CHAPTER 18

Using SEAM (Tasks)

This chapter is intended for anyone on a system with SEAM installed on it. This chapter includes information on tickets: obtaining, viewing, and destroying them. This chapter also includes information on choosing or changing a Kerberos password.

This is a list of the information in this chapter:

For an overview of SEAM, see Chapter 13.

Ticket Management

This section explains how to obtain, view, and destroy tickets. For an introduction to tickets, see "How SEAM Works" on page 210.

Do You Need to Worry About Tickets?

With SEAM 1.0 or 1.0.1 installed, Kerberos is built into the `login` command, and you will obtain tickets automatically when you log in.

Most of the Kerberized commands also automatically destroy your tickets when they exit. However, you might want to explicitly destroy your Kerberos tickets with `kdestroy` when you are finished with them, just to be sure. See "How to Destroy Tickets" on page 314 for more information on `kdestroy`.

For information on ticket lifetimes, see "Ticket Lifetimes" on page 325.

How to Create a Ticket

Normally, a ticket is created automatically when you log in, and you need not do anything special to obtain a ticket. However, you might need to create a ticket if your ticket expires.

To create a ticket, use the `kinit` command.

```
% /usr/bin/kinit
```

`kinit` prompts you for your password. For the full syntax of the `kinit` command, see the `kinit`(1) man page.

Example—Creating a Ticket

This example shows a user, `jennifer`, creating a ticket on her own system:

```
% kinit
Password for jennifer@ENG.EXAMPLE.COM:       <type password>
```

Here, the user `david` creates a ticket that is valid for three hours with the `-l` option:

```
% kinit -l 3h david@EXAMPLE.ORG
Password for david@EXAMPLE.ORG:      <type password>
```

This example shows the user `david` creating a forwardable ticket (with the `-f` option) for himself. With this forwardable ticket, he can, for example, log in to a second system.

```
% kinit -f david@EXAMPLE.ORG
Password for david@EXAMPLE.ORG:          <type password>
```

For more on how forwarding tickets works, see "Types of Tickets" on page 324.

How to View Tickets

Not all tickets are alike. One ticket might be, for example, *forwardable*; another ticket might be *postdated*; while a third ticket might be both forwardable and postdated. You can see which tickets you have, and what their attributes are, by using the `klist` command with the `-f` option:

```
% /usr/bin/klist -f
```

The following symbols indicate the attributes that are associated with each ticket, as displayed by `klist`:

F	Forwardable
f	Forwarded
P	Proxiable
p	Proxy
D	Postdatable
d	Postdated
R	Renewable
I	Initial
i	Invalid

"Types of Tickets" on page 324 describes the various attributes that a ticket can have.

Example—Viewing Tickets

This example shows that the user `jennifer` has an *initial* ticket, which is *forwardable* (F) and *postdated* (d), but not yet validated (i):

```
% /usr/bin/klist -f
Ticket cache: /tmp/krb5cc_74287
Default principal: jenniferm@ENG.EXAMPLE.COM

Valid starting                 Expires                  Service principal
09 Mar 99 15:09:51  09 Mar 99 21:09:51  nfs/EXAMPLE.SUN.COM@EXAMPLE.SUN.COM
        renew until 10 Mar 99 15:12:51, Flags: Fdi
```

The following example shows that the user `david` has two tickets that were *forwarded* (f) to his host from another host. The tickets are also *forwardable* (F):

```
% klist -f
Ticket cache: /tmp/krb5cc_74287
Default principal: david@EXAMPLE.SUN.COM

Valid starting                 Expires                  Service principal
07 Mar 99 06:09:51  09 Mar 99 23:33:51  host/EXAMPLE.COM@EXAMPLE.COM
        renew until 10 Mar 99 17:09:51, Flags: fF

Valid starting                 Expires                  Service principal
08 Mar 99 08:09:51  09 Mar 99 12:54:51  nfs/EXAMPLE.COM@EXAMPLE.COM
        renew until 10 Mar 99 15:22:51, Flags: fF
```

How to Destroy Tickets

Usually, tickets are destroyed automatically when the commands that created them exit. However, you might want to explicitly destroy your Kerberos tickets when you are finished with them, just to be sure. Tickets can be stolen. If tickets are stolen, the person who has stolen them can use them until they expire (although stolen tickets must be decrypted).

To destroy your tickets, use the `kdestroy` command.

```
% /usr/bin/kdestroy
```

`kdestroy` destroys *all* your tickets. You cannot use this command to selectively destroy a particular ticket.

If you are going to be away from your system and are concerned about an intruder using your permissions, you should use either `kdestroy` or a screen saver that locks the screen.

Note – One way to help ensure that your tickets are always destroyed is to add the `kdestroy` command to the `.logout` file in your home directory.

In instances where the PAM module has been configured (which is the default and usual case), tickets are destroyed automatically upon logout. So, adding a call to `kdestroy` to your `.login` file is not necessary. However, if the PAM module has not been configured, or if you don't know whether it has been, you might want to add `kdestroy` to your `.login` file to ensure that your tickets are destroyed when you exit your system.

Password Management

With SEAM installed, you now have two passwords: your regular Solaris password, and a Kerberos password. You can make both passwords the same, or they can be different.

Non-Kerberized commands, such as `login`, are typically set up through PAM to authenticate with both Kerberos and UNIX. If you have different passwords, you must provide both passwords to log on with the appropriate authentication. However, if both passwords are the same, the first password you enter for UNIX is also accepted by Kerberos.

Unfortunately, using the same password for both Kerberos and UNIX can compromise security. That is, if someone discovers your Kerberos password, then your UNIX password is no longer a secret. However, using the same passwords for UNIX and Kerberos is still more secure than in a site without Kerberos, because passwords in a Kerberos environment are not sent across the network. Usually, your site will have a policy to help you determine your options.

Your Kerberos password is the only way Kerberos can verify your identity. If someone discovers your Kerberos password, Kerberos security becomes meaningless, because that person can masquerade as you. That person can send email that comes from "you," read, edit, or delete your files, or log into other hosts as you. No one will be able to tell the difference. For this reason, it is vital that you choose a good password and keep it secret. You should *never* reveal your password to anyone else, not even your system administrator. Additionally, you should change your password frequently, particularly any time that you believe someone might have discovered it.

Advice on Choosing a Password

Your password can include almost any character that you can type. The main exceptions are the Control keys and the Return key. A good password is a password that you can remember readily, but which no one else can easily guess. Examples of bad passwords include the following:

- Words that can be found in a dictionary
- Any common or popular name
- The name of a famous person or character
- Your name or user name in any form (for example: your name spelled backward, repeated twice, and so forth)
- A spouse's name, child's name, or pet's name
- Your birth date or a relative's birth date
- Your social security number, driver's license number, passport number, or other similar identifying number
- Any sample password that appears in this manual or any other manual

A good password is at least eight characters long. Moreover, a password should include a mix of characters, such as uppercase and lowercase letters, numbers, and punctuation marks. Examples of passwords that would be good if they didn't appear in this manual include the following:

- Acronyms, such as "I2LMHinSF" (which is recalled as "I too left my heart in San Francisco")
- Easy-to-pronounce nonsense words, such as "WumpaBun" or "WangDangdoodle!"
- Deliberately misspelled phrases, such as "6o'cluck" or "RrriotGrrrlsRrrule!"

Caution – Don't use these examples. Passwords that appear in manuals are the first passwords that an intruder will try.

Changing Your Password

You can change your Kerberos password in two ways:

- With the usual UNIX `passwd` command. With SEAM installed, the Solaris `passwd` command also automatically prompts for a new Kerberos password.

  The advantage of using `passwd` instead of `kpasswd` is that you can set both passwords (UNIX and Kerberos) at the same time. However, you generally do not *have* to change both passwords with `passwd`. Often, you can change only your UNIX password and leave the Kerberos password untouched, or vice-versa.

  Note – The behavior of `passwd` depends on how the PAM module is configured. You might be required to change both passwords in some configurations. For some sites, the UNIX password must be changed, while other sites require the Kerberos password to change.

- With the `kpasswd` command. `kpasswd` is very similar to `passwd`. One difference is that `kpasswd` changes only Kerberos passwords. You must use `passwd` if you want to change your UNIX password.

  Another difference is that `kpasswd` can change a password for a Kerberos principal that is not a valid UNIX user. For example, `david/admin` is a Kerberos principal, but not an actual UNIX user, so you must use `kpasswd` instead of `passwd`.

After you change your password, it takes some time for the change to propagate through a system (especially over a large network). Depending on how your system is set up, this delay might take anywhere from a few minutes to an hour or more. If you need to get new Kerberos tickets shortly after you change your password, try the new password first. If the new password doesn't work, try again using the old password.

Kerberos V5 allows system administrators to set criteria about allowable passwords for each user. Such criteria is defined by the *policy* set for each user (or by a default policy). See "Administering Policies" on page 291 for more on policies.

For example, suppose that user `jennifer`'s policy (call it `jenpol`) mandates that passwords be at least eight letters long and include a mix of at least two kinds of characters. `kpasswd` will therefore reject an attempt to use "sloth" as a password.

```
% kpasswd
kpasswd: Changing password for jennifer@ENG.EXAMPLE.COM.
```

```
Old password:       <jennifer types her existing password>
kpasswd: jennifer@ENG.EXAMPLE.COM's password is controlled by
the policy jenpol
which requires a minimum of 8 characters from at least 2 classes
(the five classes are lowercase, uppercase, numbers, punctuation,
and all other characters).
New password:     <jennifer types 'sloth'>
New password (again):      <jennifer re-types 'sloth'>
kpasswd: New password is too short.
Please choose a password which is at least 4 characters long.
```

Here, `jennifer` uses "slothrop49" as a password. "slothrop49" meets the criteria, because it is over eight letters long and contains two different kinds of characters (numbers and lowercase letters).

```
% kpasswd
kpasswd: Changing password for jennifer@ENG.EXAMPLE.COM.
Old password:      <jennifer types her existing password>
kpasswd: jennifer@ENG.EXAMPLE.COM's password is controlled by
the policy jenpol
which requires a minimum of 8 characters from at least 2 classes
(the five classes are lowercase, uppercase, numbers, punctuation,
and all other characters).
New password:      <jennifer types 'slothrop49'>
New password (again):      <jennifer re-types 'slothrop49'>
Kerberos password changed.
```

Examples—Changing Your Password

In the following example, user `david` changes both his UNIX password and Kerberos password with `passwd`.

```
% passwd
    passwd:  Changing password for david
    Enter login (NIS+) password:            <type the current UNIX password>
    New password:                           <type the new UNIX password>
    Re-enter password:                      <confirm the new UNIX password>
    Old KRB5 password:                      <type the current Kerberos password>
    New KRB5 password:                      <type the new Kerberos password>
    Re-enter new KRB5 password:             <confirm the new Kerberos password>
```

In the preceding example `passwd` asks for both the UNIX password and the Kerberos password. However, if *try_first_pass* is set in the PAM module, the Kerberos password is automatically set to the UNIX password. This is the default configuration. In that case, user `david` must use `kpasswd` to set his Kerberos password to something else, as shown next.

This example shows user `david` changing only his Kerberos password with `kpasswd`.

```
% kpasswd
kpasswd: Changing password for david@ENG.EXAMPLE.COM.
Old password:               <type the current Kerberos password>
```

```
New password:                <type the new Kerberos password>
New password (again):        <confirm the new Kerberos password>
Kerberos password changed.
```

In this example, user `david` changes the password for the Kerberos principal `david/admin` (which is not a valid UNIX user). He must use `kpasswd`.

```
% kpasswd david/admin
kpasswd:  Changing password for david/admin.
Old password:                <type the current Kerberos password>
New password:                <type the new Kerberos password>
New password (again):        <type the new Kerberos password>
Kerberos password changed.
```

CHAPTER 19

SEAM (Reference)

This chapter lists many of the files, commands, and daemons that are part of the SEAM product. In addition, this chapter provides detailed information about how the Kerberos authentication system works.

This is a list of the reference information in this chapter.

SEAM Files

TABLE 19–1 SEAM Files

File Name	Description
`~/.gkadmin`	Default values for creating new principals in the SEAM Administration Tool
`~/.k5login`	List of principals to grant access to a Kerberos account
`/etc/init.d/kdc`	`init` script to start or stop `krb5kdc`
`/etc/init.d/kdc.master`	`init` script to start or stop `kadmind`

TABLE 19–1 SEAM Files *(Continued)*

File Name	Description
`/etc/krb5/kadm5.acl`	Kerberos access control list file; includes principal names of KDC administrators and their Kerberos administration privileges
`/etc/krb5/kadm5.keytab`	Keytab file for `kadmin` service on master KDC
`/etc/krb5/kdc.conf`	KDC configuration file
`/etc/krb5/kpropd.acl`	Kerberos database propagation configuration file
`/etc/krb5/krb5.conf`	Kerberos realm configuration file
`/etc/krb5/krb5.keytab`	Keytab file for network application servers
`/etc/krb5/warn.conf`	Kerberos warning configuration file
`/etc/pam.conf`	PAM configuration file
`/tmp/krb5cc_`*uid*	Default credentials cache (*uid* is the decimal UID of the user)
`/tmp/ovsec_adm.`*xxxxxx*	Temporary credentials cache for the lifetime of the password changing operation (*xxxxxx* is a random string)
`/var/krb5/.k5.`*REALM*	KDC stash file; contains encrypted copy of the KDC master key
`/var/krb5/kadmin.log`	Log file for `kadmind`
`/var/krb5/kdc.log`	Log file for the KDC
`/var/krb5/principal.db`	Kerberos principal database
`/var/krb5/principal.kadm5`	Kerberos administrative database; contains policy information
`/var/krb5/principal.kadm5.lock`	Kerberos administrative database lock file
`/var/krb5/principal.ok`	Kerberos principal database initialization file; created when the Kerberos database is initialized successfully
`/var/krb5/slave_datatrans`	Backup file of the KDC that the `kprop_script` script uses for propagation

PAM Configuration File

The default PAM configuration file includes entries for the authentication service, account management, session management, and password management modules.

For the authentication module, the new entries are created for `rlogin`, `login`, and `dtlogin` if SEAM 1.0 or 1.0.1 are installed. An example of these entries follows. All these services use the new PAM library, `/usr/lib/security/pam_krb5.so.1`, to provide Kerberos authentication.

These entries use the `try_first_pass` option, which requests authentication by using the user's initial password. Using the initial password means that the user is not prompted for another password, even if multiple mechanisms are listed.

```
# cat /etc/pam.conf
 .
 .
rlogin auth optional /usr/lib/security/pam_krb5.so.1 try_first_pass
login auth optional /usr/lib/security/pam_krb5.so.1 try_first_pass
dtlogin auth optional /usr/lib/security/pam_krb5.so.1 try_first_pass
other auth optional /usr/lib/security/pam_krb5.so.1 try_first_pass
```

For the account management module, `dtlogin` has a new entry that uses the Kerberos library, as follows. An `other` entry is included to provide a default rule. Currently, no actions are taken by the `other` entry.

```
dtlogin account optional /usr/lib/security/pam_krb5.so.1
other account optional /usr/lib/security/pam_krb5.so.1
```

The last two entries in the `/etc/pam.conf` file are shown next. The `other` entry for session management destroys user credentials. The new `other` entry for password management selects the Kerberos library.

```
other session optional /usr/lib/security/pam_krb5.so.1
other password optional /usr/lib/security/pam_krb5.so.1 try_first_pass
```

SEAM Commands

This section lists some commands that are included in the SEAM product.

TABLE 19–2 SEAM Commands

Command	Description
`/usr/lib/krb5/kprop`	Kerberos database propagation program
`/usr/sbin/gkadmin`	Kerberos database administration GUI program; used to manage principals and policies
`/usr/sbin/kadmin`	Remote Kerberos database administration program (run with Kerberos authentication); used to manage principals, policies, and keytab files

TABLE 19–2 SEAM Commands *(Continued)*

Command	Description
`/usr/sbin/kadmin.local`	Local Kerberos database administration program (run without Kerberos authentication; must be run on master KDC); used to manage principals, policies, and keytab files
`/usr/sbin/kdb5_util`	Creates Kerberos databases and stash files

SEAM Daemons

The following table lists the daemons that the SEAM product uses.

TABLE 19–3 SEAM Daemons

Daemon	Description
`/usr/lib/krb5/kadmind`	Kerberos database administration daemon
`/usr/lib/krb5/kpropd`	Kerberos database propagation daemon
`/usr/lib/krb5/krb5kdc`	Kerberos ticket processing daemon

SEAM Terminology

The following section presents terms and their definitions. Those terms are used throughout the SEAM documentation. In order to grasp SEAM concepts, an understanding of these terms is essential.

Kerberos-Specific Terminology

You need to understand the terms in this section in order to administer KDCs.

The *Key Distribution Center* or *KDC* is the component of SEAM that is responsible for issuing credentials. These credentials are created by using information that is stored in the KDC database. Each realm needs at least two KDCs, a master and at least one slave. All KDCs generate credentials, but only the master KDC handles any changes to the KDC database.

A *stash file* contains an encrypted copy of the master key for the KDC. This key is used when a server is rebooted to automatically authenticate the KDC before starting the `kadmind` and `krb5kdc` commands. Because this file includes the master key, the file and any backups of the file should be kept secure. If the encryption is compromised, then the key could be used to access or modify the KDC database.

Authentication-Specific Terminology

You need to know the terms in this section to understand the authentication process. Programmers and system administrators should be familiar with these terms.

A *client* is the software that runs on a user's workstation. The SEAM software that runs on the client makes many requests during this process. So, it is important to differentiate the actions of this software from the user.

The terms server and service are often used interchangeably. To clarify, the term *server* is used to define the physical system that SEAM software is running on. The term *service* corresponds to a particular function that is being supported on a server (for instance, `nfs`). Documentation often mentions servers as part of a service, but this definition clouds the meaning of the terms. Therefore, the term *server* refers to the physical system. The term *service* refers to the software.

The SEAM product includes three types of keys. One key is the *private key*. The private key is given to each user principal and is known only to the user of the principal and to the KDC. For user principals, the key is based on the user's password. For servers and services, the key is known as a *service key*. The service key serves the same purpose as the private key, but is used by servers and services. The third type of key is a *session key*. A session key is a key that is generated by the authentication service or the ticket-granting service. A session key is generated to provide secure transactions between a client and a service.

A *ticket* is an information packet that is used to securely pass the identity of a user to a server or service. A ticket is valid for only a single client and a particular service on a specific server. A ticket contains the principal name of the service, the principal name of the user, the IP address of the user's host, a time stamp, and a value to define the lifetime of the ticket. A ticket is created with a random session key to be used by the client and the service. After a ticket has been created, it can be reused until the ticket expires.

A *credential* is a packet of information that includes a ticket and a matching session key. Credentials are often encrypted by using either a private key or a service key, depending on which software decrypts the credential.

An *authenticator* is another type of information. When used with a ticket, an authenticator can be used to authenticate a user principal. An authenticator includes the principal name of the user, the IP address of the user's host, and a time stamp. Unlike a ticket, an authenticator can be used once only, usually when access to a service is requested. An authenticator is encrypted by using the session key for that client and that server.

Types of Tickets

Tickets have properties that govern how they can be used. These properties are assigned to the ticket when it is created, although you can modify a ticket's properties later. For example, a ticket can change from *forwardable* to *forwarded*. You can view ticket properties with the `klist` command. See "How to View Tickets" on page 312.

Tickets can be described by one or more of the following terms:

Forwardable/forwarded	A forwardable ticket can be sent from one host to another, obviating the need for a client to reauthenticate itself. For example, if the user `david` obtains a forwardable ticket while on user `jennifer`'s machine, he can log in to his own machine without having to get a new ticket (and thus authenticate himself again). See "Example—Creating a Ticket" on page 312 for an example of a forwardable ticket.
Initial	An *initial* ticket is a ticket that is issued directly, not based on a ticket-granting ticket. Some services, such as applications that change passwords, can require tickets to be marked *initial* in order to assure themselves that the client can demonstrate a knowledge of its secret key. An *initial* ticket indicates that the client has recently authenticated itself (instead of relying on a ticket-granting ticket, which might have been around for a long time).
Invalid	An *invalid* ticket is a postdated ticket that has not yet become usable. An invalid ticket will be rejected by an application server until it becomes validated. To be validated, a ticket must be presented to the KDC by the client in a TGS request, with the `VALIDATE` flag set, after its start time has passed.
Postdatable/postdated	A *postdated* ticket is a ticket that does not become valid until some specified time after its creation. Such a ticket is useful, for example, for batch jobs that are intended to be run late at night, since the ticket, if stolen, cannot be used until the batch job is to be run. When a *postdated* ticket is issued, it is issued as *Invalid* and remains that way until:

	its start time has passed, and the client requests validation by the KDC. A *postdated* ticket is normally valid until the expiration time of the ticket-granting ticket. However, if the ticket is marked *renewable*, its lifetime is normally set to be equal to the duration of the full life of the ticket-granting ticket.
Proxiable/proxy	At times, it is necessary for a principal to allow a service to perform an operation on its behalf. An example might be when a principal requests a service to run a print job on a third host. The service must be able to take on the identity of the client, but need only do so for that single operation. In that case, the server is said to be acting as a *proxy* for the client. The principal name of the proxy must be specified when the ticket is created. A *proxiable* ticket is similar to a *forwardable* ticket, except that it is valid only for a single service, whereas a *forwardable* ticket grants the service the complete use of the client's identity. A *forwardable* ticket can therefore be thought of as a sort of super-proxy.
Renewable	Because it is a security risk to have tickets with very long lives, tickets can be designated as *renewable*. A *renewable* ticket has two expiration times: the time at which the current instance of the ticket expires, and the maximum lifetime for any ticket. If a client wants to continue to use a ticket, the client renews it before the first expiration occurs. For example, a ticket can be valid for one hour, with all tickets having a maximum lifetime of 10 hours. If the client that is holding the ticket wants to keep it for more than an hour, the client must renew it within that hour. When a ticket reaches the maximum ticket lifetime (10 hours), it automatically expires and cannot be renewed.

For information on how to view tickets to see what their attributes are, see "How to View Tickets" on page 312.

Ticket Lifetimes

Any time a principal obtains a ticket, including a ticket–granting ticket, the ticket's lifetime is set as the smallest of the following lifetime values:

- The lifetime value that is specified by the `-l` option of `kinit`, if `kinit` is used to get the ticket.
- The maximum lifetime value (`max_life`) that is specified in the `kdc.conf` file.

- The maximum lifetime value that is specified in the Kerberos database for the service principal that provides the ticket. In the case of `kinit`, the service principal is `krbtgt/`*realm*.
- The maximum lifetime value that is specified in the Kerberos database for the user principal that requests the ticket.

Figure 19–1 shows how a TGT's lifetime is determined and where the four lifetime values come from. Even though this figure shows how a TGT's lifetime is determined, basically the same thing happens when any principal obtains a ticket. The only differences are that `kinit` doesn't provide a lifetime value, and the service principal that provides the ticket provides a maximum lifetime value (instead of the `krbtgt/`*realm* principal).

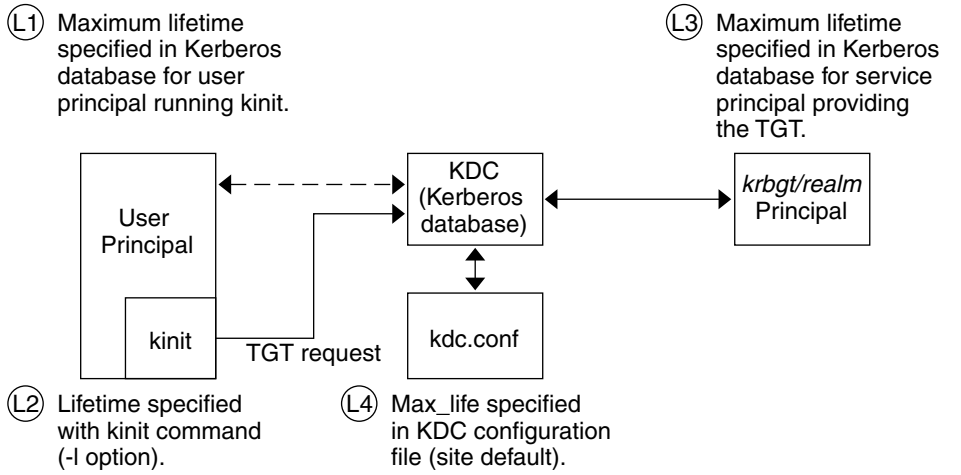


FIGURE 19–1 How a TGT's Lifetime is Determined

The *renewable* ticket lifetime is also determined from the minimum of four values, but renewable lifetime values are used instead, as follows:

- The renewable lifetime value that is specified by the `-r` option of `kinit`, if `kinit` is used to obtain or renew the ticket.
- The maximum renewable lifetime value (`max_renewable_life`) that is specified in the `kdc.conf` file.
- The maximum lifetime renewable value that is specified in the Kerberos database for the service principal that provides the ticket. In the case of `kinit`, the service principal is `krbtgt/`*realm*.
- The maximum lifetime renewable value that is specified in the Kerberos database for the user principal that requests the ticket.

Principal Names

Each ticket is identified by a principal name. The principal name can identify a user or a service. Here are examples of several principal names.

TABLE 19–4 Examples of Principal Names

Principal Name	Description
`root/boston.example.com@EXAMPLE.COM`	A principal that is associated with the `root` account on an NFS client. This principal is called a `root` principal and is needed for authenticated NFS-mounting to succeed.
`host/boston.example.com@EXAMPLE.COM`	A principal that is used by the network applications servers, such as `ftpd` and `telnetd`. This principal is also used with the `pam_krb5` authentication module. This principal is called a `host` or service principal.
username`@EXAMPLE.COM`	A principal for a user.
username`/admin@EXAMPLE.COM`	An `admin` principal that can be used to administer the KDC database.
`nfs/boston.example.com@EXAMPLE.COM`	A principal that is used by the NFS service. This principal can be used instead of a `host` principal.
`K/M@EXAMPLE.COM`	The master key name principal. There is one master key name principal that is associated with each master KDC.
`kadmin/history@EXAMPLE.COM`	A principal which includes a key used to keep password histories for other principals. Each master KDC has one of these principals.
`kadmin/kdc1.example.com@EXAMPLE.COM`	A principal for the master KDC server that allows access to the KDC by using `kadmind`.
`changepw/kdc1.example.com@EXAMPLE.COM`	A principal for the master KDC server that allows access to the KDC when you are changing passwords.
`krbtgt/EXAMPLE.COM@EXAMPLE.COM`	This principal is used when you generate a ticket-granting ticket.

How the Authentication System Works

Applications allow you to log in to a remote system if you can provide a ticket that proves your identity, and a matching session key. The session key contains information that is specific to the user and the service that is being accessed. A ticket and session key are created by the KDC for all users when they first log in. The ticket

and the matching session key form a credential. While using multiple networking services, a user can gather many credentials. The user needs to have a credential for each service that runs on a particular server. For instance, access to the `ftp` service on a server named `boston` requires one credential. Access to the `ftp` service on another server requires its own credential.

The process of creating and storing the credentials is transparent. Credentials are created by the KDC that sends the credential to the requester. When received, the credential is stored in a credential cache.

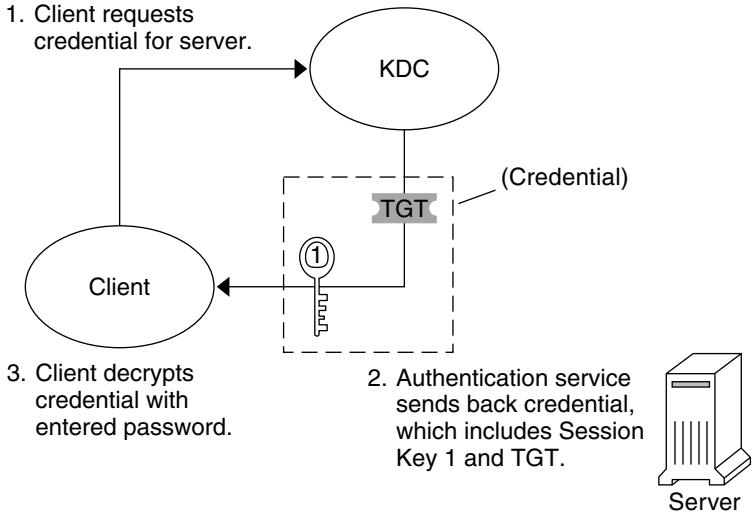
Gaining Access to a Service Using SEAM

In order for a user to access a specific service on a specific server, the user must obtain two credentials. The first credential is for the ticket-granting service (known as the TGT). Once the ticket-granting service has decrypted this credential, the service creates a second credential for the server that the user is requesting access to. This second credential can then be used to request access to the service on the server. After the server has successfully decrypted the second credential, then the user is given access. The following sections describe this process in more detail.

Obtaining a Credential for the Ticket-Granting Service

1. To start the authentication process, the client sends a request to the authentication server for a specific user principal. This request is sent without encryption. There is no secure information included in the request, so it is not necessary to use encryption.
2. When the request is received by the authentication service, the principal name of the user is looked up in the KDC database. If a principal matches, the authentication service obtains the private key for that principal. The authentication service then generates a session key to be used by the client and the ticket-granting service (call it session key 1) and a ticket for the ticket-granting service (ticket 1). This ticket is also known as the ticket-granting ticket (TGT). Both the session key and the ticket are encrypted by using the user's private key, and the information is sent back to the client.
3. The client uses this information to decrypt session key 1 and ticket 1, by using the private key for the user principal. Since the private key should only be known by the user and the KDC database, the information in the packet should be safe. The client stores the information in the credentials cache.

During this process, a user is normally prompted for a password. If the password the user enters is the same as the password that was used to build the private key stored in the KDC database, then the client can successfully decrypt the information that is sent by the authentication service. Now the client has a credential to be used with the ticket-granting service. The client is ready to request a credential for a server.



TGT = Ticket-granting ticket
KDC = Key Distribution Center

FIGURE 19–2 Obtaining a Credential for the Ticket-Granting Service

Obtaining a Credential for a Server

1. To request access to a specific server, a client must first have obtained a credential for that server from the authentication service. See "Obtaining a Credential for the Ticket-Granting Service" on page 328. The client then sends a request to the ticket-granting service, which includes the service principal name, ticket 1, and an authenticator that was encrypted with session key 1. Ticket 1 was originally encrypted by the authentication service by using the service key of the ticket-granting service.
2. Because the service key of the ticket-granting service is known to the ticket-granting service, ticket 1 can be decrypted. The information in ticket 1 includes session key 1, so the ticket-granting service can decrypt the authenticator. At this point, the user principal is authenticated with the ticket-granting service.
3. Once the authentication is successful, the ticket-granting service generates a session key for the user principal and the server (session key 2), and a ticket for the server (ticket 2). Session key 2 and ticket 2 are then encrypted by using session key 1. Since session key 1 is known only to the client and the ticket-granting service, this

information is secure and can be safely sent over the network.

4. When the client receives this information packet, the client decrypts the information by using session key 1, which it had stored in the credential cache. The client has obtained a credential to be used with the server. Now the client is ready to request access to a particular service on that server.

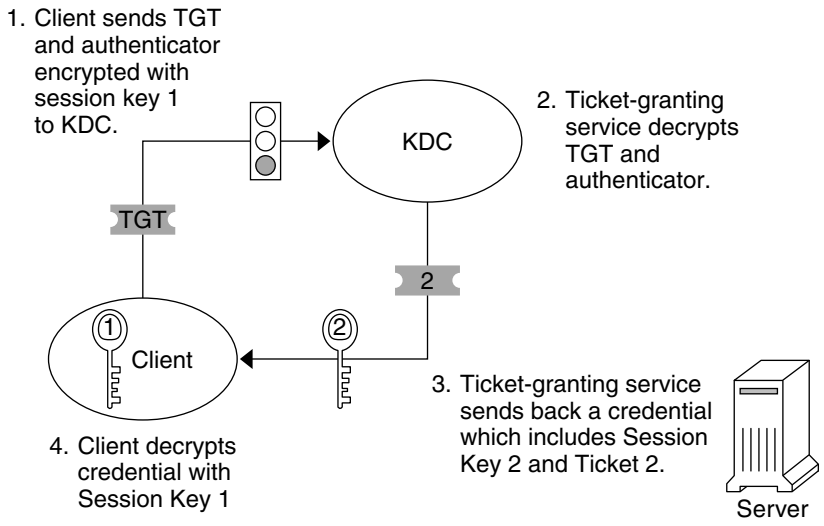


FIGURE 19–3 Obtaining a Credential for a Server

Obtaining Access to a Specific Service

1. To request access to a specific service, the client must first have obtained a credential for the ticket-granting service from the authentication server, and a server credential from the ticket-granting service. See "Obtaining a Credential for the Ticket-Granting Service" on page 328 and "Obtaining a Credential for a Server" on page 329. The client can send a request to the server including ticket 2 and another authenticator. The authenticator is encrypted by using session key 2.
2. Ticket 2 was encrypted by the ticket-granting service with the service key for the service. Since the service key is known by the service principal, the service can decrypt ticket 2 and get session key 2. Session key 2 can then be used to decrypt the authenticator. If the authenticator is successfully decrypted, the client is given access to the service.

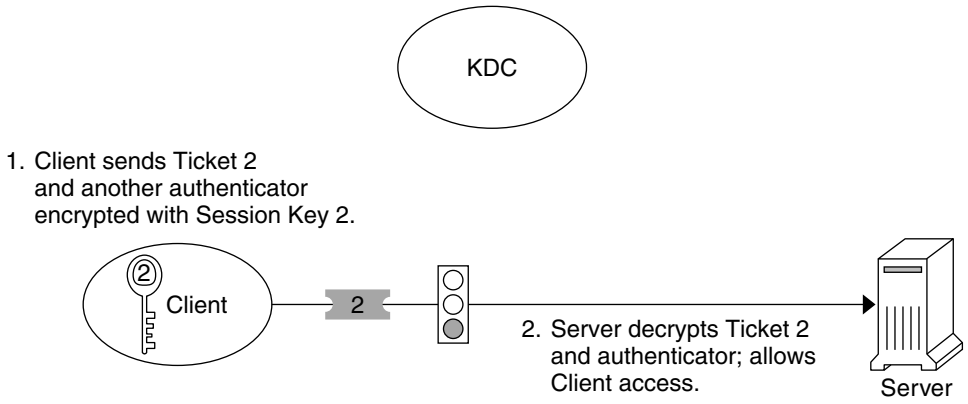


FIGURE 19–4 Obtaining Access to a Specific Service

Using the `gsscred` Table

The `gsscred` table is used by an NFS server when the server is trying to identify a SEAM user. The NFS services use UNIX IDs to identify users. These IDs are not part of a user principal or credential. The `gsscred` table provides a mapping from UNIX UIDs (from the password file) to principal names. The table must be created and administered after the KDC database is populated.

When a client request comes in, the NFS services try to map the principal name to a UNIX ID. If the mapping fails, the `gsscred` table is consulted. With the `kerberos_v5` mechanism, a `root/hostname` principal is automatically mapped to UID 0, and the `gsscred` table is not consulted. Thus, there is no way to do special remappings of `root` through the `gsscred` table.

PART IV Auditing and Device Management

Chapter 20	Provides overview information about the Basic Security Module, BSM. The module provides security through auditing and through device management.
Chapter 21	Provides information to help with implementing the use of auditing at your site.
Chapter 22	Provides step-by-step instructions to configure and manage auditing.
Chapter 23	Provides information about the files that are associated with auditing. Describes the structure of the audit tokens. Describes the components of device management.

CHAPTER 20

BSM (Overview)

The Basic Security Module (BSM) provides two security features. The first feature is an auditing mechanism, which includes tools to assist with the analysis of the auditing data. The second feature is a device-allocation mechanism, which provides the required object-reuse characteristics for removable devices or assignable devices.

This chapter introduces the concepts behind BSM. The following is a list of the information in this chapter.

What Is Auditing?

Auditing is the collection of data about the use of machine resources. The audit data provides a record of security-related system events. This data can then be used to assign responsibility to actions that take place on a host. Successful auditing starts with two security features: identification and authentication. At login, after a user supplies a user name and password, a unique audit ID is associated with the user's process. The audit ID is inherited by every process that is started during the login session. Even if a user changes identity, all user actions are tracked with the same audit ID. See the `su`(1M) man page.

The auditing subsystem makes the following possible:

- Monitor security-relevant events that take place on the host
- Record the events in a network-wide audit trail
- Detect misuse or unauthorized activity

- Review patterns of access, and see the access histories of individuals and objects
- Discover attempts to bypass the protection mechanisms
- Discover which users are using root capabilities

During system configuration, you select which activities to monitor. You can also fine-tune the degree of auditing that is done for individual users.

After audit data is collected, audit-reduction and interpretation tools allow you to examine interesting parts of the audit trail. For example, you can choose to review audit records for individual users or specific groups. You can examine all records for a certain type of event on a specific day. Or you can select records that were generated at a certain time of day.

How Does Auditing Work?

Auditing is the generation of audit records when specified events occur. Most commonly, events that generate audit records include the following:

- System startup and system shutdown
- Login and logout
- Process creation or process destruction, or thread creation or thread destruction
- Opening, closing, creating, destroying, or renaming of objects
- Use of root capabilities or role capabilities
- Identification, and authentication actions
- Discretionary Access Control (DAC) changes by a process or user
- Installation-specific administrative actions

Audit records are generated from three sources:

- By an application
- As a result of an asynchronous event
- As a result of a process system call

Once the relevant event information has been captured, the information is formatted into an audit record. The record is then placed in a kernel buffer known as the audit queue. From this temporary location within the kernel, audit records are written to audit files. Where the audit files are located is determined by entries in the `audit_control` file. The location can include multiple partitions on the same machine, partitions on different machines, or partitions on machines on different but linked networks. The collection of audit files that are linked together is considered an audit trail.

Audit records accumulate in audit files chronologically. Contained in each audit record is information that identifies the event, what caused the event, the time of the event, and other relevant information.

How Is Auditing Related to Security?

To secure a computer system, especially a system on a network, is complex. Security requires mechanisms that control activities before system processes or user processes begin. Security requires tools that monitor activities as the activities occur. Security also requires reports of activities after the activities have happened. Initial configuration of Solaris auditing requires that parameters be set before users log in or machine processes begin. Most auditing activities involve monitoring current events and reporting those events that meet the specified parameters. How Solaris auditing monitors and reports these events is discussed in detail in Chapter 21 and Chapter 22.

Auditing cannot prevent hackers from unauthorized entry. However, the audit subsystem can report, for example, that a specific user performed specific actions at a specific time and date. The audit report can identify the user by entry path and user name. Such information can be reported immediately to your terminal and to a file for later analysis. Thus, the audit subsystem provides data that helps you determine the following:

- How system security was compromised
- What loopholes need to be closed to ensure the desired level of security

BSM Terminology

The following terms are used to describe the BSM service. Some definitions include pointers to more complete descriptions.

TABLE 20–1 BSM Terms

Term	Definition
Audit class	A grouping of audit events. Audit classes provide a way to manage a group of events. For more information, see “Audit Classes” on page 340.

TABLE 20–1 BSM Terms *(Continued)*

Term	Definition
Audit directory	A repository of audit files. For a description of the types of audit directories, see "Audit Directory" on page 341.
Audit event	A security-related system action that is audited. For a discussion of the types of audit events, see "Audit Events" on page 338.
Audit flag	A short name for a class. The audit flag is used to determine which classes of events to audit and when to audit those classes of events. For more information about audit flags, see "Audit Flags" on page 340.
Audit policy	A set of auditing options that you can enable or disable for a particular configuration.These options include whether to record certain kinds of audit data. The options also include whether to suspend auditable actions when the audit trail is full.
Audit record	Audit data. Audit data is stored in binary form. An audit record describes a single audit event. Each audit record is composed of audit tokens. For more information about audit records, see "Audit Records and Audit Tokens" on page 340.
Audit token	A set of audit data. Audit tokens are stored in binary form. Each audit token describes an attribute of an audit event, such as a process, a path, or other object. For descriptions of all the audit tokens, see "Audit Token Formats" on page 391.
Audit trail	A collection of one or more audit files that might reside in separate audit file partitions.
Device allocation	A mechanism that enables you to restrict use of a device, as well as to erase any leftover data after a device has been used. For a description of device allocation, see "Device Allocation" on page 341.
Public objects	A public object is a file that is in a system directory, such as the `/etc` or the `/usr/bin` directory. System directories are readable by everyone. The files in system directories are often read for information or for execution. In the Solaris 9 8/03 release, public objects, by default, are no longer audited for read-only events. For example, even if the `fr` audit flag is turned on, the reading of public objects is not audited. The `public` policy flag argument to the `auditconfig -setpolicy` command determines the policy setting.

Audit Events

Security-relevant system actions can be audited. These auditable actions are defined as *audit events*. Audit events are listed in the `/etc/security/audit_event` file. Each auditable event is defined in the file by a symbolic name, an event number, a set of preselection classes, and a short description. See the `audit_event`(4) man page.

There are several categories of audit events. The primary distinction is between kernel-level events and user-level events. Events that are generated by the kernel are called kernel-level events. Events that are generated by applications are called user-level events. Kernel-level events have a lower audit event number than a user-level event, as shown in the following table.

TABLE 20–2 Audit Event Categories

Number Range	Type of Event
1–2047	Kernel-level audit events
2048–65535	User-level audit events
2048–32767	Reserved for SunOS user-level programs
32768–65535	Available for third-party applications

Kernel-Level Audit Events

Events that are generated by the kernel are system calls. System calls have audit event numbers between 1 and 2047. The event names for kernel events begin with `AUE_`, followed by an uppercase mnemonic for the event. For example, the event number for the `creat()` system call is `4`, and the event name is `AUE_CREAT`.

User-Level Audit Events

Events that are generated by application software are outside the kernel. Application software generates user-level events. User-level events range in number from 2048 to 65535. The event names begin with `AUE_`, followed by a lowercase mnemonic for the event. For example, the event number for the `rlogin` command is `6155`, and the event name is `AUE_rlogin`. Table 20–2 shows general categories of user-related events.

Nonattributable Audit Events

Most events are attributable to an individual user, but some events are not. Events are *nonattributable* if the events occur at the kernel-interrupt level, or if the events occur before a user is identified and authenticated. Nonattributable events are auditable. The following example lists two nonattributable events from the `/etc/security/audit_event` file:

```
153:AUE_ENTERPROM:enter prom:na
6156:AUE_mountd_mount:mount:na
```

`AUE_ENTERPROM` is a kernel-level `na` event. `AUE_mountd_mount` is a user-level `na` event.

Audit Classes

Each audit event is also defined as belonging to an *audit class* or classes. Audit classes are convenient containers for large numbers of audit events. When you preselect a class to be audited, you preselect for auditing all the events in that class. Audit classes are defined in the `/etc/security/audit_class` file. Each entry contains the name of the class, its audit mask, and its short name.

```
0x00000010:fc:file create
0x00000400:na:non-attribute
```

The mapping of audit events to classes is configurable. These configuration changes are made in the `audit_event` file.

An auditable event is recorded in the audit trail when you preselect an audit class for auditing that includes the specific event. There are 32 possible audit classes. The classes include the two global classes: `all` and `no`. The audit classes are described in Table 23–1. See also the `audit_class`(4) man page.

Audit Flags

Audit flags are the short names for audit classes. Audit flags are used in the `audit_control` file to specify machine-wide defaults for auditing on the machine. The `audit_control` file is described in "The `audit_control` File" on page 380.

You can make exceptions to the machine-wide auditing defaults for individual users. You put audit flags in a user's entry in the `audit_user` file. The audit flags are also used as arguments to the `auditconfig` command. See the `auditconfig`(1M) and `audit_user`(4) man pages.

Audit Records and Audit Tokens

Each *audit record* describes the occurrence of a single audited event. The record includes information such as who did the action, which files were affected, what action was attempted, and where and when the action occurred. For example, the following line shows an audit record when processed by the `praudit` command:

```
header,81,2,login - local,,Mon May  6 16:55:48 PDT 2002, + 486 msec
subject,root,root,other,root,other,378,378,0 0 example_machine
text,successful login
return,success,0
```

Audit records are collected in an audit file. The set of audit files from a machine or site is called the *audit trail*. For a description of how audit files are handled, see the `audit.log`(4) man page. Audit records can be converted to a readable format by the `praudit` command. For examples of `praudit` output, see "The `praudit` Command" on page 377. See also the `praudit`(1M) man page.

The type of information that is saved for each audit event is defined in a set of *audit tokens*. Each time an audit record is created for an event, the record contains some or all of the tokens that are defined for the event. The nature of the event determines which tokens are recorded. You can generate audit record descriptions with the `bsmrecord` command. The following output shows the structure of the audit record that is generated when the `creat()` system call is used. The lines beginning with `header` are the audit tokens.

```
creat
  system call creat                see creat(2)
  event ID    4                    AUE_CREAT
  class       fc                   (0x00000010)
      header
      path
      [attribute]
      subject
      return
```

For more information, see "How to Display Audit Record Formats" on page 364. For a description of the structure of each audit token, see "Audit Token Formats" on page 391. The `audit.log`(4) man page also lists the audit tokens.

Audit Directory

An *audit directory* holds a collection of audit files. A typical installation uses many audit directories. The three types of audit directories are as follows:

- **Primary audit directory –** A directory where the audit files for a system are placed under normal conditions.
- **Secondary audit directory –** A directory where the audit files for a system are placed if the primary directory is full or not available.
- **Directory of last resort –** A local audit directory that is used if the primary audit directory and all secondary audit directories are not available.

Device Allocation

The device-allocation mechanism enables you to restrict access to a device, such as a CD-ROM. Device clean scripts erase any leftover data after a device has been used. These actions increase the security of a device. For more information, see "Managing Device Allocation (Tasks)" on page 369 or "Device Allocation Reference" on page 409.

CHAPTER 21

Audit Planning

This chapter describes how to set up auditing for your Solaris installation. In particular, the chapter covers issues that you need to consider before you enable the auditing service. The following is a list of the planning information in this chapter:

For an overview of auditing, see Chapter 20. For procedures to configure auditing at your site, see Chapter 22.

Handling the Audit Trail

File space for the audit trail is one of the biggest issues with auditing. Each host should have several audit directories that are configured for audit files. You should decide how to configure the audit directories as one of the first steps before you enable auditing on any hosts. The following table lists the issues that should be resolved when you plan for audit trail storage.

Issues Related to Audit Trail Storage	What You Need to Plan For
1. Determine how much auditing your site needs.	Balance your site's security needs against the availability of disk space for the audit trail. For guidance on how to reduce space requirements while still maintaining site security, as well as how to design audit storage, see "Controlling Auditing Costs" on page 348 and "Auditing Efficiently" on page 350.
2. Determine which systems are to be audited. Determine which systems are to store audit files.	Decide which hosts in your network need to be audited. Make sure to create at least one local audit directory for each host that is to be audited. Then, decide which hosts are to hold most of the audit trail.
3. Determine the names and locations of the audit directories.	Create a list of all the audit directories that you plan to use.
4. Plan which hosts are to use which audit directories.	Create a map that shows which host should use which audit directory. This step helps you to balance the auditing activity.

Deciding Who and What to Audit

You want to be selective about what kinds of activities are audited, and you want to collect useful audit information. Audit files can quickly grow to fill the available space, so you should plan what to audit.

Issues Related to What to Audit	What You Need to Plan For
1. Determine the audit classes that you need for your site	The best time to add audit classes or to change the default classes is before you start the auditing service. See "Audit Classes and Their Audit Flags" on page 385 for information about auditing classes.
2. Determine event-to-class mappings	In many situations, the default mapping is sufficient. However, if you have added new classes or have changed class definitions, you might also need to move an event to a different class.

Issues Related to What to Audit	What You Need to Plan For
3. Decide what classes should be audited for all users on all machines	The system-wide audit flags in the `audit_control` file apply to all users and processes. Audit flags determine whether an audit class is audited for success, for failure, or for both. Every machine in the installation should have the same audit flag settings in the machines' `audit_control` file.
4. Determine user exceptions to the installation-wide audit settings	If you decide that some users should be audited differently from the system-wide settings, modify the users' entries in the `audit_user` file on each machine. For more information, see "How to Change Users' Audit Characteristics" on page 354.
5. Determine the minimum free disk space (`minfree`) that should be available on an audit file system before a warning is sent	When disk space on an audit file system drops below the `minfree` percentage, the audit daemon switches to the next available audit directory. The daemon then sends a warning that the soft limit has been exceeded. For more information, see "Example—Changing the Soft Limit for Warnings" on page 353.
6. Decide which audit policies your site needs	The policy variable is a dynamic kernel variable, so its value is not saved when the system is brought down. Therefore, you should set the desired policy by using an appropriate startup script. For more information, see "How to Enable or Disable an Audit Policy" on page 362.
7. Decide how to manage the `audit_warn` mail alias	The `audit_warn` script is run by the `auditd` daemon whenever the daemon switches audit directories. The script also runs when the daemon encounters a difficulty, such as a lack of disk space. By default, the `audit_warn` script sends mail to an `audit_warn` alias and sends a message to the console. You need to either modify the alias, or change the script to send the message to a different alias. For more information, see "How to Configure the `audit_warn` Alias" on page 361.

Issues Related to What to Audit	What You Need to Plan For
8. Decide what to do when all the audit directories are full	To permit the system to continue functioning in the event of an audit trail overflow, you can enable the `cnt` policy. For a `cnt` policy example, see "Example—Setting the `cnt` Policy" on page 362. Alternatively, you can create an account that can log in and work without being audited. See "Example—Creating an Audit Admin Login" on page 355 for an audit account example.

Determining Which Audit Policies to Use

Audit policies determine the characteristics of the audit records for the local host. The policies are set by a startup script. The `bsmconv` script, which enables the auditing service, creates the `/etc/security/audit_startup` script. The `audit_startup` script executes the `auditconfig` command to establish audit policy. See the `audit_startup`(1M) man page.

The audit policies are disabled by default to minimize storage requirements and system processing demands. You can enable and disable audit policies dynamically with the `auditconfig` command. You can enable and disable the policies permanently with the `audit_startup` script. Use the following table to determine if the needs of your site justify the additional overhead that results from enabling one or more audit policies.

TABLE 21–1 Effects of Audit Policies

Policy Name	Description	Why Change the Policy?
`arge`	When disabled, this policy omits environment variables of an executed program script from the `exec` audit record. When enabled, this policy adds the environment variables of an executed program script to the `exec` audit record. The resulting audit records contain much more detail than when this policy is disabled.	The disabled option collects much less information than the enabled option. The enabled option makes sense when you are auditing a few users. The option is also useful when you have suspicions about the environment variables that are being used in `exec` programs.
`argv`	When disabled, this policy omits the arguments of an executed program script from the `exec` audit record. When enabled, this policy adds the arguments of an executed program script to the `exec` audit record. The resulting audit records contain much more detail than when this policy is disabled.	The disabled option collects much less information than the enabled option. The enabled option makes sense when you are auditing a few users. The option is also useful when you have reason to believe that unusual `exec` programs are being run.
`cnt`	When disabled, this policy blocks a user or application from running. The blocking happens when audit records cannot be added to the audit trail because no disk space is available. When enabled, this policy allows the event to complete without an audit record being generated. The policy maintains a count of audit records that are dropped.	The disabled option makes sense in an environment where security is paramount. The enabled option makes sense when system availability is more important than security.
`group`	When disabled, this policy does not add a groups list to audit records. When enabled, this policy adds a groups list to every audit record as a special token.	The disabled option usually satisfies requirements for site security. The enabled option makes sense when you need to audit which groups are generating auditable events.
`path`	When disabled, this policy records in an audit record at most one path that is used during a system call. When enabled, this policy records every path that is used in conjunction with an audit event to every audit record.	The disabled option places at most one path in an audit record. The enabled option enters each file name or path that is used during a system call in the audit record as a path token.

TABLE 21–1 Effects of Audit Policies *(Continued)*

Policy Name	Description	Why Change the Policy?
`public`	New in the Solaris 9 8/03 release. When disabled, this policy does not add read-only events of public objects to the audit trail when the reading of files is preselected. Audit flags that contain read-only events include `fr`, `fa`, and `cl`. When enabled, this policy records every read-only audit event of public objects if an appropriate audit flag is preselected.	The disabled option usually satisfies requirements for site security. The enabled option is rarely useful.
`seq`	When disabled, this policy does not add a sequence number to every audit record. When enabled, this policy adds a sequence number to every audit record. The `seq` token holds the sequence number.	The disabled option is sufficient when auditing is running smoothly. The enabled option makes sense when you are checking that audit files are being written correctly. In the case of file corruption, you might be able to spot bad records quickly. The sequence numbers might be out of order, or some numbers might be missing. A partially written audit record is an example of file corruption.
`trail`	When disabled, this policy does not add a `trailer` token to audit records. When enabled, this policy adds a `trailer` token to every audit record.	The disabled option creates a smaller audit record. The enabled option marks the end of each audit record clearly with a trailer token. The trailer token is often used in conjunction with the sequence token when debugging. In the case of file corruption , the `auditreduce` command resyncs faster on good records. A partially written audit record is an example of file corruption.

Controlling Auditing Costs

Because auditing consumes system resources, you must control the degree of detail that is recorded. When you decide what to audit, consider the following costs of auditing:

- Cost of increased processing time
- Cost of analysis of audit data

- Cost of storage of audit data

Cost of Increased Processing Time of Audit Data

The cost of increased processing time is the least significant of the costs of auditing. The first reason is that auditing generally does not occur during computation-intensive tasks, such as image processing, complex calculations, and so forth. The other reason is that the cost for single-user systems is usually small enough to ignore.

Cost of Analysis of Audit Data

The cost of analysis is roughly proportional to the amount of audit data that is collected. The cost of analysis includes the time that is required to merge and review audit records. Cost also includes the time that is required to archive the records and keep the records in a safe place.

The fewer records that you generate, the less time that is required to analyze the audit trail. Upcoming sections, "Cost of Storage of Audit Data" on page 349 and "Auditing Efficiently" on page 350, describe ways to audit efficiently. You can reduce the amount of data that you collect, while still providing enough coverage to achieve your site's security goals.

Cost of Storage of Audit Data

Storage cost is the most significant cost of auditing. The amount of audit data depends on the following:

- Number of users
- Number of machines
- Amount of use
- Degree of security that is required

Because these factors vary from site to site, no formula can determine in advance the amount of disk space to set aside for audit data storage.

Full auditing, that is, with the `all` flag, fills up disks quickly. Even a simple task such as compiling a program could generate a large audit file. A program of modest size could generate thousands of audit records in less than a minute. For example, the audit trail of a program of five files and 5000 lines would occupy many megabytes of disk space. You judiciously use the preselection features to reduce the volume of records that are generated. For example, by omitting the `fr` class, you can reduce the audit volume by more than two-thirds. Efficient audit file management is also important. After the audit records are created, file management reduces the amount of storage that is required.

Before you configure auditing, you should understand the audit flags. You should understand the types of events that the flags audit. Develop a philosophy of auditing for your site that is based on sensible measures. Such measures include the amount of security that your site requires, and the types of users that you administer.

Auditing Efficiently

The techniques in this section can help you achieve your organization's security goals while auditing more efficiently:

- Randomly audit only a certain percentage of users at any one time
- Reduce the disk-storage requirements for audit files by combining, reducing, and compressing the files. Develop procedures for archiving the files, for transferring the files to removable media, and for storing the files offline.
- Monitor the audit data in real time for unusual behaviors. You can set up procedures to monitor the audit trail as the trail is generated for certain activities. You can write a script to trigger an automatic increase in the auditing of certain users or certain machines in response to detection of unusual events.

  For example, write a script that (1) monitors the creation of audit files on all the audit file servers, and (2) processes them with the `tail` command. See the `tail`(1) man page. Pipe the output of `tail -0f` through the `praudit` command. The command yields a stream of audit records as the records are generated. This stream can be analyzed for unusual message types or other indicators, and delivered to the auditor. Or, the script can be used to trigger automatic responses.

  In addition, the script should include code that (3) constantly monitors the audit directories for the appearance of new `not_terminated` audit files. The script should (4) also terminate outstanding `tail` processes when their files are no longer being written to.

CHAPTER 22

Managing the BSM Service (Tasks)

This chapter presents procedures that are designed to help you set up and manage a Solaris environment that includes auditing. This chapter also includes instructions for administering the audit trail and for administering device allocation. The following is a list of the task maps in this chapter.

For an overview of auditing, see Chapter 20. For planning suggestions, see Chapter 21.

Managing the BSM Service (Task Map)

The following task map shows the major tasks that are required to administer the BSM service.

Task	Description	For Instructions
Plan for auditing	Configuration issues to consider and make decisions about, before you configure auditing.	Chapter 21
Configure audit files	Defines which events, classes, and users require auditing.	"Configuring Audit Files (Task Map)" on page 352
Configure auditing	Configures each host so that you can use auditing.	"Configuring the Auditing Service (Task Map)" on page 358

Task	Description	For Instructions
Manage audit records	Merges and analyzes the audit data.	"Managing Audit Records (Task Map)" on page 364
Manage device allocation	Defines which devices should be accessed through the device allocation mechanism.	"Managing Device Allocation (Tasks)" on page 369

Configuring Audit Files (Task Map)

Before you enable auditing on your network, you might want to edit the audit configuration files. Many of the following procedures require you to restart the service or reboot the local system. You should make as many of these changes as possible before you start the service.

The following task map describes the tasks in this section.

Task	Description	For Instructions
Select audit flags, change `audit_control` settings	Preselects the events are being to be audited. Changes preset values in the `audit_control` file.	"How to Select Audit Flags" on page 352
Change audit characteristics for users	Sets user-specific exceptions to the system-wide audit flag settings.	"How to Change Users' Audit Characteristics" on page 354
Add audit classes	Defines new audit classes.	"How to Add Audit Classes" on page 355
Change event-to-class mappings	Changes the class that certain events belong to	"How to Change an Audit Event's Class Membership" on page 356
Add audit events	Adds new user-level events to the `audit_event` file.	"How to Add Audit Events" on page 357

▼ How to Select Audit Flags

Audit flags are defined in the `/etc/security/audit_control` file. The audit flags select which classes of audit records are written to the audit log.

1. **Become superuser or assume an equivalent role.**
2. **(Optional) Save a backup copy of the `audit_control` file.**

```
# cp /etc/security/audit_control /etc/security/audit_control.save
```

3. **Add new entries to the `audit_control` file.**

   Each entry has the following format:

   title`:`*string*

   title Defines the type of line. Options are `dir:`, `flags:`, `minfree:`, or `naflags:`.

   string Lists specific data that is associated with the line type.

4. **Instruct the audit daemon to read the new `audit_control` file.**

   The audit daemon stores the information internally. To use the new information, either reboot the system or type the following command:

   ```
   # audit -s
   ```

Example—Changing the Location of the Audit Trail File

Lines that start with `dir:` define which audit file systems can be used to store audit trail files. In this example, two additional locations for audit trail files are defined.

```
# cat /etc/security/audit_control
dir:/etc/security/audit/host.1/files
dir:/etc/security/audit/host.2/files
dir:/var/audit
flags:
minfree:10
naflags:lo
```

Example—Changing Audit Flags for All Users

The `flags` line in the `audit_control` file defines which classes of events are audited for all users on the host. The classes are separated by commas, with no spaces. In this example, the events in the `lo` class are audited for all users.

```
# cat /etc/security/audit_control
dir:/var/audit
flags:lo
minfree:10
naflags:lo
```

Example—Changing the Soft Limit for Warnings

The `minfree` line in the `audit_control` file defines the minimum free-space level for all audit file systems. In this example, the soft limit is set so that a warning is issued when only 10 percent of the file system is available.

```
# cat /etc/security/audit_control
dir:/var/audit
flags:
minfree:10
naflags:lo
```

Example—Changing Auditing of Nonattributable Events

The `naflags:` line in the `audit_control` file defines which classes of nonattributable events are audited for all users on the host. The classes are separated by commas, with no spaces. In this example, the `na` event class was added.

```
# cat /etc/security/audit_control
dir:/var/audit
flags:
minfree:10
naflags:lo,na
```

▼ How to Change Users' Audit Characteristics

Definitions for each user are stored in the `/etc/security/audit_user` file. These definitions are exceptions to the flags in the `audit_control` file.

1. **Become superuser or assume an equivalent role.**
2. **(Optional) Save a backup copy of the `audit_user` file.**

   ```
   # cp /etc/security/audit_user /etc/security/audit_user.save
   ```

3. **Add new entries to the `audit_user` file.**

   Each entry has the following format:

   username:*always*:*never*

username	Selects the name of the user to be audited.
always	Selects the list of audit classes that should always be audited.
never	Selects the list of audit classes that should never be audited.

   You can specify multiple flags by separating the audit classes with commas. For more information about audit flags, see "Audit Classes and Their Audit Flags" on page 385.

4. **Make the new data available to the auditing daemon.**

   To use the new data, you can reboot the system. You can also have the user log out and then log back in again.

Example—Changing Auditing for One User

This example shows an entry that causes audit records to be generated any time that the user `sue` accesses any programs in the login class (`lo`).

```
# grep sue /etc/security/audit_user
sue:lo:
```

Example—Creating an Audit Admin Login

If all the audit partitions are full and logins are audited, then users might not be able to log in to a host. To avoid this situation, you can set up a special account that is not audited. The special account could log in to the host even when the audit partitions are full, and fix the problem with the full partitions. In this example, the account `auditadm` is defined so that no auditing takes place.

```
# grep auditadm /etc/security/audit_user
auditadmin:no:yes
```

Note – The user who is selected to use the audit admin account might need to be monitored in another way.

▼ How to Add Audit Classes

Audit classes are defined in the `/etc/security/audit_class` file.

1. **Become superuser or assume an equivalent role.**
2. **(Optional) Save a backup copy of the `audit_class` file.**

   ```
   # cp /etc/security/audit_class /etc/security/audit_class.save
   ```

3. **Add new entries to the `audit_class` file.**

   Each entry has the following format:

   `0x`*number*`:`*name*`:`*description*

0x	Identifies *number* as hexadecimal.
number	Defines the unique audit class mask.
name	Defines the two-letter name of the audit class.
description	Defines the descriptive name of the audit class.

4. **Make the new data available to the BSM service .**

   To use the new data, either reboot the system, or type the following command:

```
# auditconfig -conf
```

Example—Setting a New Audit Class

In this example, add an entry to the `audit_class` file that resembles the following entry. The entry creates a new audit class that is called `ta`.

```
0x01000000:ta:test application
```

▼ How to Change an Audit Event's Class Membership

Event-class mappings are defined in the `/etc/security/audit_event` file.

1. **Become superuser or assume an equivalent role.**
2. **(Optional) Save a backup copy of the `audit_event` file.**

   ```
   # cp /etc/security/audit_event /etc/security/audit_event.orig
   ```

3. **Change the class to which particular events belong by changing the *flag* of the events.**

   Each entry has the following format:

   number : *event* : *program* : *flag*

number	Defines the audit event ID.
event	Defines the name of the audit event.
program	Defines the system call or user-level program executable that triggers the creation of an audit record.
flag	Defines the two-letter name of the audit class.

4. **Make the new data available to the BSM service.**

   To use the new data, either reboot the system, or type the following commands:

   ```
   # auditconfig -conf
   # audit -s
   ```

Example—Creating a Site-Specific Audit Event Mapping

In this example, you define a new class, and then add events to that class. To use the mapping, put the new class in the `audit_control` file, then reboot the system.

1. In the `audit_class` file, define a site-specific class to collect just those audit events that you want to monitor.

   ```
   0x00000800:sc:site class
   ```

2. In the `audit_event` file, change a set of audit events to the new class.

   ```
   26:AUE_SETGROUPS:setgroups(2):sc
   27:AUE_SETPGRP:setpgrp(2):sc
   40:AUE_SETREUID:setreuid(2):sc
   41:AUE_SETREGID:setregid(2):sc
   214:AUE_SETEGID:setegid(2):sc
   215:AUE_SETEUID:seteuid(2):sc
   ```

3. Use the new flag in the `audit_control` file. The following entry audits logins, and audits all successful invocations of the events in the `sc` class.

   ```
   flags:lo,+sc
   ```

4. To ensure that the new configuration audits all processes, reboot the system. Or, you can use the following set of commands to ensure that each user who uses the machine is correctly audited. *auid* is the user ID.

   ```
   # auditconfig -conf
   # audit -s
   # setumask auid lo,+sc
   ```

▼ How to Add Audit Events

Audit event definitions are stored in the `/etc/security/audit_event` file.

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

2. **(Optional) Save a backup copy of the `audit_event` file.**

   ```
   # cp /etc/security/audit_event /etc/security/audit_event.save
   ```

3. **Add new entries to the `audit_event` file.**

   Each entry has the following format:

   number : *name* : *description* : *classes*

number	Defines a unique audit event number, which must start after 32767.
name	Defines the unique audit event name.
description	Describes the audit event. Often includes the name of the man page for the audit event.
classes	Selects the audit classes that include this event.

4. **Make the new data available to the auditing daemon.**

   To use the new data, either reboot the system, or type the following command:

```
# auditconfig -conf
```

Example—Adding a New Audit Event

This example shows an entry that defines a new audit event for a local application.

```
# grep localapp /etc/security/audit_event
32768:AUE_localapp:localapp(1):ta
```

Configuring the Auditing Service (Task Map)

This section covers the tasks that are required to configure and enable the auditing service. The following task map describes the tasks that are required to configure the auditing service.

Task	Description	For Instructions
1. (Optional) Change the audit configuration files	Selects which events, classes, and users require auditing.	"Configuring Audit Files (Task Map)" on page 352
2. Create audit partitions	Creates the partitions for the audit files.	"How to Create Partitions for Auditing" on page 359
3. Create the `audit_warn` alias	Defines who should get email warnings.	"How to Configure the `audit_warn` Alias" on page 361
4. (Optional) Change audit policies	Defines additional audit records or auditing conditions.	"How to Enable or Disable an Audit Policy" on page 362
5. Enable auditing	Turns on auditing.	"How to Enable Auditing" on page 363
6. (Optional) Disable auditing	Turns off auditing.	"How to Disable Auditing" on page 363
7. (Optional) Start device allocation	Selects which removable media should be accessed in a more secure mode.	"Managing Device Allocation (Tasks)" on page 369

▼ How to Create Partitions for Auditing

The following procedure shows how to create partitions for audit files, as well as the corresponding file systems and directories. Skip steps as necessary, depending on if you already have an empty partition, or if you have already mounted an empty file system.

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

2. **Determine the amount of disk space that is required.**

   Assign at least 200 Mbytes of disk space per host. However, the disk space requirements are based on how much auditing you perform. So, your requirements might be far greater than this figure. Remember to include a partition for a directory of last resort.

3. **Create dedicated audit partitions, as needed.**

   This step is most easily done during server installation. You can also create the partitions on disks that have not yet been mounted on the server. For complete instructions on how to create the partitions, see "Creating a UFS File System" in *System Administration Guide: Basic Administration*.

   ```
   # newfs /dev/rdsk/cwtxdysz
   ```

   where /dev/rdsk/c*w*t*x*d*y*s*z* is the raw device name for the partition.

   If the local host is to be audited, create an audit directory of last resort for the local host as well.

4. **Create mount points for each new partition.**

   ```
   # mkdir /var/audit/server-name.n
   ```

   Where *server-name.n* is the name of the server plus a number that identifies each partition. The number is optional, but the number is useful when there are many audit directories.

5. **Add entries to automatically mount the new partitions.**

   Add a line to the `/etc/vfstab` file that resembles the following:

   ```
   /dev/dsk/cwtxdysz /dev/rdsk/cwtxdysz /var/audit/server-name.n   ufs  2  yes
   ```

6. **(Optional) Remove the minimum free space threshold on each partition.**

   If you use the default configuration, a warning is generated when the directory is 80 percent full. The warning removes the reason to reserve free space on the partition.

   ```
   # tunefs -m 0 /var/audit/server-name.n
   ```

7. **Mount the new audit partitions.**

   ```
   # mount /var/audit/server-name.n
   ```

8. **Create audit directories on the new partitions.**

```
# mkdir /var/audit/server-name.n/files
```

9. **Correct the permissions on the mount points and new directories.**

```
# chmod -R 750 /var/audit/server-name.n/files
```

10. **(Optional) On a file server, define the file systems to be made available to other hosts.**

Often, disk farms are installed to store the audit records. If an audit directory is to be used by several systems, then the directory must be shared through the NFS service. Add an entry that resembles the following for each directory to the `/etc/dfs/dfstab` file.

```
share -F nfs /var/audit/server-name.n/files
```

11. **(Optional) On a file server, restart the NFS service.**

If this command is the first `share` command or set of `share` commands that you have initiated, the NFS daemons are probably not running. The following commands kill the daemons and restart the daemons. Refer to "Setting Up NFS Services" in *System Administration Guide: Resource Management and Network Services* for more information about the NFS service.

```
# /etc/init.d/nfs.server stop
# /etc/init.d/nfs.server start
```

Example—Creating an Audit Directory of Last Resort

All systems that run the auditing subsystem should have a local file system that can be used if no other file system is available. In this example, a file system is being added to a system that is named `egret`. Since this file system is only used locally, none of the steps for a file server are followed.

```
# newfs /dev/rdsk/c0t2d0
# mkdir /var/audit/egret
# grep egret /etc/vfstab
/dev/dsk/c0t2d0s1  /dev/rdsk/c0t2d0s1  /var/audit/egret ufs  2  yes  -
# tunefs -m 0 /var/audit/egret
# mount /var/audit/egret
# mkdir /var/audit/egret/files
# chmod -R 750 /var/audit/egret/files
```

Example—Creating New Audit Partitions

In this example, a new file system is created on two new disks that are to be used by other systems in the network.

```
# newfs /dev/rdsk/c0t2d0
# newfs /dev/rdsk/c0t2d1
# mkdir /var/audit/egret.1
```

```
# mkdir /var/audit/egret.2
# grep egret /etc/vfstab
/dev/dsk/c0t2d0s1  /dev/rdsk/c0t2d0s1  /var/audit/egret.1 ufs  2  yes  -
/dev/dsk/c0t2d1s1  /dev/rdsk/c0t2d1s1  /var/audit/egret.2 ufs  2  yes  -
# tunefs -m 0 /var/audit/egret.1
# tunefs -m 0 /var/audit/egret.2
# mount /var/audit/egret.1
# mount /var/audit/egret.2
# mkdir /var/audit/egret.1/files
# mkdir /var/audit/egret.2/files
# chmod -R 750 /var/audit/egret.1/files /var/audit/egret.2/files
# grep egret /etc/dfs/dfstab
 share -F nfs /var/audit/egret.1/files
 share -F nfs /var/audit/egret.2/files
# /etc/init.d/nfs.server stop
# /etc/init.d/nfs.server start
```

▼ How to Configure the `audit_warn` Alias

The `audit_warn` script generates mail to an alias that is called `audit_warn`. To send this mail to a valid email address, you can follow either of the following options:

1. **Become superuser or assume an equivalent role.**
2. **Configure the `audit_warn` mail alias.**

   OPTION 1 – Replace the `audit_warn` alias with another mail account in the `audit_warn` script.

   After you replace `audit_warn` with the `root` account, the line that sends the email message would resemble the following:

   ```
   /usr/ucb/mail -s "$SUBJECT" root
   ```

   Ten lines in the `audit_warn` script require this change.

   OPTION 2 – Redirect the `audit_warn` email to another mail account.

   In this case, you would add the `audit_warn` alias to the appropriate mail aliases file. You could add the alias to the local `/etc/mail/aliases` file or to the `mail_aliases` database in the name space. The new entry would resemble the following if the `root` mail account was made a member of the `audit_warn` alias:

   ```
   audit_warn: root
   ```

▼ How to Enable or Disable an Audit Policy

Audit policies determine the characteristics of the audit records for the local host. By default, all audit policies are disabled. You need to enable any audit policies that you want to use. For a description of each policy, see "Audit Policies" on page 387.

You can inspect, enable, or disable the current audit policy with the `auditon()` system call at the program level. Or, to do the same task, you can run the `auditconfig` command. You can also modify the policy options to the `auditconfig` command in the `audit_startup` script to make more permanent audit policy changes.

1. **Become superuser or assume an equivalent role.**
2. **(Optional) Review the existing audit policies.**

   Ensure that you are aware of all the policies that are being used before you change any policies. The following command lists the enabled policies:

   ```
   # auditconfig -lspolicy
   ```

3. **Enable or disable the audit policy.**

   ```
   # auditconfig -setpolicy flagpolicyname
   ```

   flag A *flag* value of + enables the policy. A *flag* value of - disables the policy.

   policyname Selects the policy to be enabled or to be disabled.

   The policy is in effect until the next boot, or until the policy is modified by the `auditconfig -setpolicy` command.

Example—Setting the `cnt` Policy

The `cnt` policy can be set so that if the audit partitions become full, then processes are not blocked. The records are discarded when the partitions are full, but the system still functions. The `cnt` policy keeps a count of the number of discarded audit records. The `cnt` policy should not be set if security is paramount, since unrecorded events can occur if the file system is full.

The following command enables the `cnt` policy:

```
# auditconfig -setpolicy +cnt
```

To maintain the policy across reboots, you should place the `auditconfig -setpolicy +cnt` command in the `audit_startup` file.

▼ How to Enable Auditing

This task starts the auditing service. If the service has been configured, then rebooting the host also starts the service.

1. **Become superuser or assume an equivalent role.**
2. **Bring the system into single-user mode.**

   ```
   # /etc/telinit 1
   ```

   See the `telinit`(1M) man page for more information.
3. **Run the script to configure the system to run auditing.**

   Go to the `/etc/security` directory, and execute the `bsmconv` script there. The script sets up a standard Solaris machine to run auditing after a reboot. See the `bsmconv`(1M) man page.

   ```
   # cd /etc/security
   # ./bsmconv
   ```
4. **Bring the system into multiuser mode.**

   ```
   # /etc/telinit 6
   ```

   The startup file `/etc/security/audit_startup` causes the audit daemon to run automatically when the system enters multiuser mode.

▼ How to Disable Auditing

If auditing is no longer required at some point, you can disable the auditing subsystem by running the `bsmunconv` command. See the `bsmconv`(1M) man page.

1. **Become superuser or assume an equivalent role.**
2. **Bring the system into single-user mode.**

   ```
   # /etc/telinit 1
   ```

   See the `telinit`(1M) man page for more information.
3. **Run the script to disable auditing.**

   Change to the `/etc/security` directory, and execute the `bsmunconv` script there.

   ```
   # cd /etc/security
   # ./bsmunconv
   ```
4. **Bring the system into multiuser mode.**

   ```
   # /etc/telinit 6
   ```

Managing Audit Records (Task Map)

By managing the audit trail, you can monitor the actions of users on your network. Auditing can generate large amounts of data. The following tasks show you how to work with all this data.

The following task map describes the tasks in this section.

Task	Description	For Instructions
Display the formats of audit records	Displays the order of tokens for a particular audit event.	"How to Display Audit Record Formats" on page 364
Display audit records	Displays the audit records in readable format.	"How to Display Audit Records" on page 368
Merge audit records	Combines audit files from several machines into one audit trail.	"How to Merge Audit Records" on page 366
Prevent audit trail overflow	Prevents the audit file systems from completely filling up.	"How to Prevent Audit Trail Overflow" on page 368

▼ How to Display Audit Record Formats

The `bsmrecord` command displays the audit id, audit class, selection mask, and record format of an audit event. The command operates on records in the `audit_class` and `audit_event` files.

The `-a` option in the following command lists all audit event record formats. The `-h` option puts the list in HTML format. The resulting file can be displayed in a browser.

- **Use the `bsmrecord` command to put the format of all audit event records in an HTML file.**

```
% bsmrecord -a -h > audit.events.html
```

You can display the `*html` file in a browser. Use the browser's Find tool to find specific records.

See the `bsmrecord`(1M) man page for more information.

Example—Displaying the Audit Record Formats of a Program

In this example, the format of all audit records that are generated by the `login` program are displayed.

```
% bsmrecord -p login

terminal login
  program     /usr/sbin/login      see login(1)
  event ID    6152                 AUE_login
  class       lo                   (0x00001000)
      header
      subject
      text                         error message or "successful login"
      return

login: logout
  program     /usr/sbin/login      see login(1)
  event ID    6153                 AUE_logout
  class       lo                   (0x00001000)
      header
      subject
      text                         "logout" username
      return

rlogin
  program     /usr/sbin/login      see login(1) - rlogin
  event ID    6155                 AUE_rlogin
  class       lo                   (0x00001000)
      header
      subject
      text                         success/fail message
      return

telnet login
  program     /usr/sbin/login      see login(1) - telnet
  event ID    6154                 AUE_telnet
  class       lo                   (0x00001000)
      header
      subject
      text                         success/fail message
      return
```

Example—Displaying the Audit Record Formats of an Audit Class

In this example, the format of all audit records in the `fd` class are displayed.

```
% bsmrecord -c fd

ftruncate
    Not used.

truncate
    Not used.

unlink
  system call unlink               see unlink(2)
```

```
event ID    6                   AUE_UNLINK
class       fd                  (0x00000020)
    header
    path
    [attribute]
    subject
    return
```

▼ How to Merge Audit Records

This task shows you how to merge all audit files in all the audit directories. Follow these steps when you want to analyze the contents of the audit trail.

1. **Become superuser or assume an equivalent role.**
2. **Change directories to the primary audit directory.**

   # **cd /etc/security/audit/***server-name***.1/files**

   The merged file is placed in the `/etc/security/audit/`*server-name*`.1/files` directory. This directory is a protected directory.
3. **Merge the audit records.**

   # **auditreduce > merged.log**

   All directories that are listed in the `dir:` lines of the `audit_control` file on *server-name* are merged. The merged records are then placed in the `merged.log` file in the current directory.

Example—Displaying the Entire Audit Trail

To display the entire audit trail at once, pipe the output of the `auditreduce` command into the `praudit` command.

auditreduce | praudit

Example—Printing the Entire Audit Trail

With a pipe to the `lp` command, the output goes to the printer.

auditreduce | praudit | lp

Example—Combining and Reducing Audit Files

Use the `auditreduce` command with the `-O` option to combine several audit files into one file and to save the files in a specified output file. `auditreduce` can do this type of combination and deletion automatically. See the `-C` and `-D` options in the `auditreduce`(1M) man page. However, you can select the files manually to good effect. Use the `find` command, then use `auditreduce` to combine just the named set of files.

When used in this way, the `auditreduce` command merges all the records from its input files into a single output file. The input files should then be deleted. In addition, the output file should be kept in a directory that is named `/etc/security/audit/`*server-name*`/files` so that `auditreduce` can find the output file.

```
# auditreduce -O combined-filename
```

The `auditreduce` command can also reduce the number of records in its output file. The command can eliminate the less interesting records as it combines the input files. For example, you might use the `auditreduce` command to retain only the login and logout records in audit files that are over a month old. If you need to retrieve the complete audit trail, you could recover the trail from backup tapes.

```
# auditreduce -O daily.summary -b 19990413 -c lo; compress *daily.summary
# mv *daily.summary /etc/security/summary.dir
```

Example—Displaying User Activity From a Selected Date

In the following example, the system administrator checks to see when user `tamiko` logged in and logged out on April 13, 1999. The administrator requests the `lo` event class. The short-form date is in the form *yymmdd*. The long form is described in the `auditreduce`(1M) man page.

```
# auditreduce -d 990413 -u tamiko -c lo | praudit
```

Example—Copying Selected Records to a Single File

In this example, login and logout messages for a particular day are selected from the audit trail. The messages are merged into a target file. The target file is written in a directory other than the normal audit root directory.

```
# auditreduce -c lo -d 990413 -O /usr/audit_summary/logins
```

The `-O` option creates an audit file with 14-character timestamps for both the start-time and the end-time, with the suffix `logins`:

```
/usr/audit_summary/19990413000000.19990413235959.logins
```

Example—Cleaning Up a not_terminated Audit File

Occasionally, an audit daemon dies while its audit file is still open. Or, a server becomes inaccessible and forces the machine to switch to a new server. In such instances, an audit file remains with the string not_terminated as the end-time, even though the file is no longer used for audit records. When you find such a file, you can manually verify that the file is no longer in use. You can clean up the open file by specifying the name of the file with the correct options.

```
# audit -s
19990414121112.not_terminated.egret
# auditreduce -O egret 19990413120429.not_terminated.egret
```

The audit command checks the name of the current audit file. The auditreduce command creates a new audit file with the correct name and correct timestamps. The correct name includes the correct suffix (egret). The auditreduce then copies all the records into the file.

▼ How to Display Audit Records

1. **Become superuser or assume an equivalent role.**
2. **Change directories to an audit files directory, such as /usr/audit_summary/logins.**

   ```
   # cd /usr/audit_summary/logins
   ```

3. **Read a file by using the praudit command.**

   ```
   # praudit 19990413000000.19990413235959.logins | more
   ```

Example—Putting Audit Records in XML Format

In this example, the audit records are converted to XML format. XML format can be displayed in a browser. The format can also be used to create a report.

```
# praudit -x 19990413000000.19990413235959.logins > 19990413.logins.xml
```

The *xml file can be displayed in a browser. The contents of the file can be operated on by a script to extract the relevant information.

▼ How to Prevent Audit Trail Overflow

If your security policy requires that all audit data be saved, do the following:

1. **Set up a schedule to regularly archive audit files. Set up a schedule to delete the archived audit files from the audit file system.**

2. **Manually archive audit files by backing up the files on tape. You can also move the files to an archive file system.**
3. **Store context-sensitive information that is necessary to interpret audit records, along with the audit trail.**
4. **Keep records of which audit files are moved offline.**
5. **Store the archived tapes appropriately.**
6. **Reduce the volume of audit data that you store by creating summary files.**

   You can extract summary files from the audit trail by using options to the `auditreduce` command. The summary files then contain only records for certain specified types of audit events. For examples, see "Example—Combining and Reducing Audit Files" on page 367 and "Example—Copying Selected Records to a Single File" on page 367.

Managing Device Allocation (Tasks)

You can use device allocation to decrease the security risk that is associated with various removable media.

Adding an Allocatable Device (Task Map)

The following task map describes the major steps that are required to define a new allocatable device.

Task	Description	For Instructions
1. Create or change an entry in the `device_allocate` file	Defines which devices are controlled by the device-allocation mechanism.	"How to Change Which Devices Can Be Allocated" on page 370
2. Create a lock file	Enables the device allocation mechanism to work on a specific device.	"How to Set Up Lock Files for an Allocatable Device" on page 370
3. (Optional) Create a device-clean script	Purges data from a physical device.	"Device-Clean Scripts" on page 414
4. Allocate the device	Adds a device to the device-allocation mechanism.	"How to Allocate a Device" on page 371

Task	Description	For Instructions
5. (Optional) Deallocate the device	Removes a device from use.	"How to Deallocate a Device" on page 371

▼ How to Set Up Lock Files for an Allocatable Device

The lock files are zero-length files that are created in the `/etc/security/dev` directory. One file is created for each allocatable device. If no lock file exists for a device, the device cannot be allocated, so no one can access the device.

1. **Become superuser or assume an equivalent role.**
2. **Obtain the device name for the device from its entry in the `device_maps` file by using the `dminfo` command.**

   See "The `device_maps` File" on page 411 and the `dminfo`(1M) and `device_maps`(4) man pages. For example, the device name for device type `st` is `st0`. In the next step, you use the device name as the name of the lock file.
3. **Create an empty lock file for the device by using the `touch` command.**

   Use the device name for the file name in place of *device-name*.

   ```
   # cd /etc/security/dev
   # touch device-name
   # chmod 600 device-name
   # chown bin device-name
   # chgrp bin device-name
   ```

▼ How to Change Which Devices Can Be Allocated

This procedure defines which devices can be used with the device allocation mechanism.

1. **Become superuser or assume an equivalent role.**
2. **Determine which devices are listed in the `/etc/security/device_allocate` file.**
3. **Decide if there are devices that are not in the `device_allocate` file, yet should be made allocatable.**
4. **Edit the `device_allocate` file and add the new device.**

   Each entry should use the following format:

   device-name`;`*device-type*`;;;;`*program*

   device-name Specifies the name of the device

device-type	Specifies the device type
program	Specifies the purge program to be run

▼ How to Allocate a Device

1. **Become superuser or assume an equivalent role.**
2. **Use the `allocate` command with a device that is specified by device name.**

   ```
   sarl% allocate st0
   ```

   You can also allocate a device by device type by using the `-g` option to the `allocate` command.

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

Example—Allocating a Printer

Only the user who ran the `allocate` command can use the printer.

```
sarl% allocate /dev/lp/chestnut
```

▼ How to Deallocate a Device

Deallocation enables other users to allocate and use the device when you are finished.

- **Deallocate a device by using the `deallocate` command followed by the device file name.**

  ```
  sarl% deallocate st0
  ```

Example—Deallocating a Printer

To deallocate a printer that is named `chestnut`, type the following command:

```
# deallocate /dev/lp/chestnut
```

Example—Forcing a Deallocation

Devices that a user has allocated are not automatically deallocated when the process terminates or when this user logs out. You most commonly need to use the following form of the `deallocate` command when a user forgets to deallocate a specific device. The following command deallocates the device so that others users can allocate the device.

```
# deallocate -F st0
```

Example—Deallocating All Devices

Caution – You can deallocate all devices only at system initialization time.

```
# deallocate -I
```

CHAPTER 23

BSM Service (Reference)

This chapter describes the important components of the BSM service, which are the auditing subsystem and the device allocation mechanism.

The auditing mechanism helps you detect potential security breaches by revealing suspicious or abnormal patterns of system usage. The auditing mechanism also provides a means to trace suspect actions back to a particular user, thus serving as a deterrent. If users know that their activities are likely to be audited, they might be less likely to attempt malicious activities.

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

For an overview of auditing, see Chapter 20. For planning suggestions, see Chapter 21. For procedures to configure auditing at your site, see Chapter 22.

Audit Commands

This section provides information about the commands that are used with the auditing service.

The Audit Daemon

The following list summarizes what the audit daemon, `auditd`, does.

- `auditd` opens and closes audit log files in the directories that are specified in the `audit_control` file, in the order in which they are specified.
- `auditd` reads audit data from the kernel and writes the data to an audit log file.
- `auditd` executes the `audit_warn` script when the audit directories fill past limits that are specified in the `audit_control` file. The script, by default, sends warnings to the `audit_warn` mail alias and to the console.
- By default, when all audit directories are full, processes that generate audit records are suspended. In addition, the `auditd` command writes a message to the console and to the `audit_warn` mail alias. The audit policy can be reconfigured with the `auditconfig` command. At this point, only the system administrator fix the auditing mechanism. The administrator can log in to write audit files to tape, can delete audit files from the system, and can do other cleanup.

The `auditd` daemon can be started automatically when the machine is brought up to multiuser mode, or you can start it from the command line. When the audit daemon is started, it determines the amount of free space necessary for audit log files.

The daemon uses the list of audit directories in the `audit_control` file as possible locations for creating audit files. The audit daemon maintains a pointer into this list of directories, starting with the first directory. Every time the audit daemon needs to create an audit file, it puts the file into the first available directory in the list. The list starts at the audit daemon's current pointer. You can reset the pointer to the beginning of the list by running the `audit -s` command. The `audit -n` command instructs the daemon to switch to a new audit file. The new file is created in the same directory as the current file.

The `audit` Command

The `audit` command controls the actions of the audit daemon. The `audit` command can do the following tasks:

- Enable and disable auditing
- Reset the audit daemon
- Adjust the auditing preselection mask on the local machine
- Write audit records to a different audit log file

See the `audit`(1M) man page for a discussion of the available options.

The bsmrecord Command

The bsmrecord command displays the format of audit events that are defined in the /etc/security/audit_event file. The output includes the event's audit ID, audit class, audit flag, and the record's tokens in order. With no option, the bsmrecord output displays well in a terminal window. With the -h option, the output is suitable for viewing in a browser. See "How to Display Audit Record Formats" on page 364 for examples of its use. For more information, see the bsmrecord(1M) man page.

The auditreduce Command

Use the auditreduce command to merge audit records from one or more input audit files. The command can also be used to perform a post selection of audit records. See the auditreduce(1M) man page. To merge the entire audit trail, run this command on the audit server. The audit server is the machine that mounts all the audit file systems for the installation.

The auditreduce command enables you to track all auditable actions on multiple machines from a single location. The command can read the logical combination of all audit files as a single audit trail. You must identically configure all machines at a site for auditing, and create servers and local directories for the audit log files. The auditreduce command ignores how the records were generated or where they are stored. Without options, the auditreduce command merges audit records from all the audit files in all of the subdirectories in the audit root directory. Typically, /etc/security/audit is the audit root directory. The auditreduce command sends the merge result to standard output. You can also place the result into a single, chronologically ordered output file. The file contains binary data.

The auditreduce command also can select particular types of records for analysis. The merging functions and selecting functions of the auditreduce command are logically independent. auditreduce captures data from the input files as the records are read, before the files are merged and then written to disk.

The praudit command makes the binary output of the auditreduce command readable.

By specifying options to the auditreduce command, you can also do the following:

- Request audit records that were generated by only certain audit flags
- Request audit records that were generated by one particular user
- Request audit records that were generated on specific dates

With no arguments, auditreduce checks the subdirectories within the /etc/security/audit directory, the default audit root directory. The command checks for a files directory in which the *start-time.end-time.hostname* files reside. The auditreduce command is very useful when audit data resides in separate directories. Figure 23–1 illustrates audit data in separate directories for different hosts. Figure 23–2) illustrates audit data in separate directories for different audit servers.

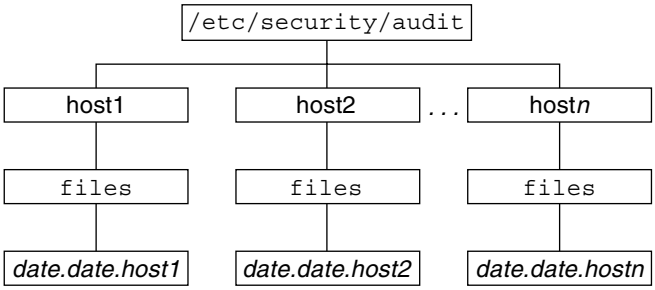


FIGURE 23–1 Audit Trail Storage Sorted by Host

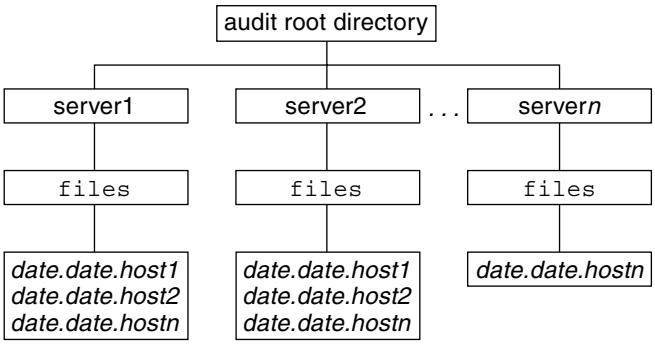


FIGURE 23–2 Audit Trail Storage Sorted by Server

If the partition for `/etc/security/audit` is very small, you might not store audit data in the default directory. You can pass the `auditreduce` command another directory by using the `-R` option:

```
# auditreduce -R /var/audit-alt
```

You can also specify a particular subdirectory by using the `-S` option:

```
# auditreduce -S /var/audit-alt/host1
```

You can direct `auditreduce` to process only certain audit log files by specifying them as command arguments:

```
# auditreduce /var/audit/egret/files/2001*.2001*egret
```

For other options and additional examples, see the `auditreduce`(1M) man page.

The praudit Command

The praudit command reads audit records in binary format from standard input and displays the records in a presentable format. The input can be piped from the auditreduce command or from a single audit file. Input can also be produced with the cat command to concatenate several files, or the tail command for a current audit file.

The praudit command can generate five output formats:

- **Default –** The default option displays one audit token per line. The default option displays the audit event by its description, such as ioctl(2), and displays any value that could be text in text format. For example, a user is displayed as the user name, not as the user ID.
- **–l option –** The long option displays one audit record per line. The -d option changes the delimiter used between token fields, and between tokens. The default delimiter is a comma.
- **–r option –** The raw option displays as a number any value that could be numeric. For example, a user is displayed by user ID, Internet addresses are in hexadecimal format, and modes are in octal format. The audit event is displayed as its event number, such as 158.
- **–s option –** The short option displays the audit event by its table name, for example, AUE_IOCTL. The option displays the other tokens as the default option displays them.
- **–x option –** The XML option displays the audit record in XML format. This option is useful as input to browsers, or as input to scripts that manipulate XML.

  The XML is described by a DTD that the audit subsystem provides. Solaris software also provides a style sheet. The DTD and the style sheet are in the /usr/share/lib/xml directory.

In the default output format of praudit, each record is easily identified as a sequence of audit tokens. Each token is on a separate line. Each record begins with a header token. You could, for example, further process the output with the awk command.

Here is the default output from the praudit command for a header token:

```
header,240,1,ioctl(2),es,Tue Sept  7 16:11:44 1999, + 270 msec
```

Here is the output from the praudit -r command for the same header token:

```
20,240,1,158,0003,699754304, + 270 msec
```

Example—Processing `praudit` Output With a Script

Sometimes, you might want to manipulate output from the `praudit` command as lines of text. For example, you might want to select records that the `auditreduce` command cannot select. You can use a simple shell script to process the output of `praudit`. The following simple example script puts one audit record on one line, searches for a user-specified string, then returns the audit file to its original form. Specifically, the script does the following:

1. Marks the `header` tokens by prefixing them with Control-A
2. Combines all the audit tokens for one record onto one line while preserving the line breaks as Control-A
3. Runs the `grep` command
4. Restores the original newline breaks

```
#!/bin/sh
praudit | sed -e '1,2d' -e '$s/^file.*$//' -e 's/^header/^aheader/' \\
| tr '\\012\\001' '\\002\\012' \\
| grep "$1" \\
| tr '\\002' '\\012'
```

Note that the `^a` in the script is Control-A, not the two characters `^` and `a`. The prefix distinguishes the header token from the string `header` that might appear as text.

The `auditconfig` Command

The `auditconfig` command provides a command-line interface to retrieve and set audit configuration parameters. The `auditconfig` command can do the following tasks:

- Display, check, and configure audit policy
- Determine if auditing is turned on or turned off
- Turn auditing off and turn auditing on
- Manage the audit directory and the audit file
- Manage the audit queue
- Get and set preselection masks
- Get and set audit event to audit class mappings
- Get and set configuration information, such as session ID and audit ID
- Configure audit characteristics for a process, a shell, and a session
- Reset audit statistics

See the `auditconfig`(1M) man page for a discussion of the command options.

Audit Service Files

Auditing uses the following files:

The /etc/system File

The /etc/system file contains commands that the kernel reads during initialization to customize the system operations. The bsmconv and bsmunconv shell scripts, which are used to activate and deactivate auditing, modify the /etc/system file. The bsmconv shell script adds the following line to the /etc/system file:

```
set c2audit:audit_load=1
```

The set c2audit:audit_load=1 command causes the auditing module, a kernel module, to be loaded when the system is booted. The bsmunconv shell script disables auditing when the system is rebooted. The command removes the c2audit line from the /etc/system file.

The audit_class File

The /etc/security/audit_class file contains definitions of the existing audit classes. Audit classes are groups of audit events. Each class has an associated audit flag, which is the short name that stands for the class. You use the short name in the audit_control file to preselect the classes whose events you want to audit. The flags accept prefixes for finer–grained selection. See "Audit Flag Syntax" on page 386 for more information.

The root user, or an administrator in an equivalent role, can modify the definitions of audit classes. This administrator can define new audit classes, rename existing classes, or otherwise change existing classes by editing the audit_class file in a text editor. See the audit_class(4) man page for more information. For descriptions of the audit flags, see "Definitions of Audit Flags" on page 385.

The audit_control File

An /etc/security/audit_control file on each machine is read by the audit daemon. See the audit_control(4) man page. The audit_control file is located in the /etc/security directory. Each machine has its own local audit_control file. The file enables every machine to mount their audit file systems from different locations or in a different order. For example, the primary audit file system for machineA might be the secondary audit file system for machineB.

You specify four kinds of information in the audit_control file. Each line of information begins with a keyword.

- The *audit flags* line begins with flags:. The line contains the audit flags that preselect which classes of events are audited for all users on the machine. The audit flags specified here are referred to as the *machine-wide audit flags* or the *machine-wide audit preselection mask*. Audit flags are separated by commas, with no spaces.
- The *nonattributable flags* line begins with naflags:. The line contains the audit flags that preselect which classes of events are audited when an action cannot be attributed to a specific user. The flags are separated by commas, with no spaces.
- The *audit threshold* line begins with minfree:. The line defines the minimum free-space level for all audit file systems. The minfree percentage must be greater than or equal to 0. The default is 20 percent. When an audit file system is 80 percent filled, the audit data is then stored in the next available audit directory. See the audit_warn(1M) man page.
- The *directory definition* lines begin with dir:. Each line defines an audit file system and directory that the machine uses to store its audit log files. You can define one or more directory definition lines. The order of the dir: lines is significant. The auditd daemon creates audit files in the directories in the specified order. The first directory is the primary audit directory for the machine. The second directory is the secondary audit directory where the audit daemon creates audit trail files when the first directory becomes full, and so forth. See the audit(1M) man page.

An audit_control file is created during the configuration process on each machine.

When you make changes to the audit_control file, you then run the audit -s command to instruct the audit daemon to reread the file.

Note – The audit -s command does not change the preselection mask for existing processes. Use auditconfig, setaudit, or auditon for existing processes. See the getaudit(2) and auditconfig(1M) man pages for more information.

Sample audit_control File

The following is a sample audit_control file for the machine dopey. dopey uses two audit file systems on the audit server blinken, and a third audit file system that is mounted from the second audit server winken. The third file system is used only when the audit file systems on blinken become full or unavailable. The minfree value of 20 percent specifies that the warning script is run when the file systems are 80 percent filled. The flags specify that logins and administrative operations are to be audited. The operations are audited for success and for failure. Failures of all types, except failures to create a file system object, are to be audited. Non-attributable events are also audited.

```
flags:lo,am,-all,^-fc
naflags:lo,nt
minfree:20
dir:/etc/security/audit/blinken/files
dir:/etc/security/audit/blinken.1/files
#
# Audit filesystem used when blinken fills up
#
dir:/etc/security/audit/winken
```

The audit_data File

When the auditd daemon starts on each machine, it creates the file /etc/security/audit_data. The format of the file consists of a single entry with the two fields separated by a colon. See the audit_data(4) man page. The first field is the audit daemon's process ID. The second field is the path name of the audit file to which the audit daemon is currently writing audit records. Here is an example:

```
# cat /etc/security/audit_data
116:/etc/security/audit/blinken.1/files/19990320100002.not_terminated.dopey
```

The audit_event File

The /etc/security/audit_event file contains the default event-to-class mappings.You can edit this file to change the class mappings. However, if you do so, you must reboot the system or run auditconfig -conf to read the changed mappings into the kernel. See the audit_event(4) man page.

The audit_startup Script

The /etc/security/audit_startup script automatically starts the audit daemon when the system enters multiuser mode. The script is invoked as part of the startup sequence, just prior to the execution of the audit daemon. See the audit_startup(1M) man page for more information.

A default `audit_startup` script that automatically configures the event-to-class mappings and sets the audit policies. The script is created during the BSM package installation.

The `audit_user` File

To audit some users differently from others, you can edit the `/etc/security/audit_user` file to add audit flags for individual users. If specified, these flags are combined with the system-wide flags in the `audit_control` file to determine which classes of events to audit for that user. The flags that you add to the user's entry in the `audit_user` file modify the defaults from the `audit_control` file in two ways:

- By specifying event classes that are always to be audited for this user
- By specifying event classes that are never to be audited for this user

Each user entry in the `audit_user` file contains three fields.

- The *username* field
- The *always-audit* field
- The *never-audit* field

The audit fields are processed in sequence. The *always-audit* field turns on the auditing of the classes in that field. The *never-audit* field turns off the auditing of the classes in that field.

Note – Avoid the common mistake of leaving the `all` audit flag in the *never-audit* field. This mistake causes all auditing to be turned off for that user, which overrides the flags that are set in the *always-audit* field. The flag also overrides machine-wide audit flags set in the `audit_control` file.

The *never-audit* flags for a user override the system defaults. You might not want to overrride system defaults. For example, suppose you want to audit everything for user `tamiko` for except for successful reads of file system objects. This strategy audits almost everything for a user. However, the strategy generates about three-quarters of the audit data that would be produced if all data reads were audited. You also want to apply the system defaults to `tamiko`. Here are two possible `audit_user` entries:

The correct entry:

```
tamiko:all,^+fr:
```

The incorrect entry:

```
tamiko:all:+fr
```

The first example means, “always audit everything except for successful file-reads.” The second example means, “always audit everything, but never audit successful file-reads.” The second example is incorrect because the *never-audit* field would override the system defaults. The first example achieves the desired effect: the *always-audit* flags include the exception to the `all` flag. Since no flag is in the *never-audit* field, the system defaults from the `audit_control` file are not overridden here.

Note – Successful events and failed events are treated separately. A process could generate more audit records when an error occurs than when an event is successful.

The `audit_warn` Script

Whenever the audit daemon encounters an unusual condition while writing audit records, it invokes the `/etc/security/audit_warn` script. See the `audit_warn`(1M) man page. You can customize this script for your site to warn of conditions that might require manual intervention. Or, you could specify how to handle those conditions automatically. For all error conditions, `audit_warn` writes a message to the console. `audit_warn` also sends a message to the `audit_warn` mail alias. You should set up this alias when you enable auditing.

When audit daemon detects the following conditions, it invokes the `audit_warn` script.

- An audit directory has become more full than the `minfree` value allows. The `minfree` value or soft limit is a percentage of the space available on an audit file system.

  The `audit_warn` script is invoked with the string `soft` and the name of the directory whose available space has gone below the minimum value. The audit daemon switches automatically to the next suitable directory. The daemon writes the audit files in this new directory until the directory reaches its `minfree` limit. The audit daemon then goes to each remaining directory in the order that is listed in the `audit_control` file. The daemon writes audit records until each directory is at its `minfree` limit.

- All the audit directories have reached the `minfree` threshold.

  The `audit_warn` script is invoked with the string `allsoft`. A message is written to the console. Mail is also sent to the `audit_warn` alias.

  When all audit directories that are listed in the `audit_control` file have reached their `minfree` threshold, the audit daemon switches back to the first directory. The daemon writes audit records until the directory becomes completely full.

- An audit directory has become completely full with no space remaining.

  The `audit_warn` script is invoked with the string `hard` and the name of the directory. A message is written to the console. Mail is also sent to the `audit_warn` alias.

The audit daemon switches automatically to the next suitable directory with any space available. The audit daemon goes to each remaining directory in the order that is listed in the `audit_control` file. The daemon writes audit records until each directory is full.

- All the audit directories are completely full. The `audit_warn` script is invoked with the string `allhard` as an argument.

  In the default configuration, a message is written to the console. Mail is also sent to the `audit_warn` alias. The processes that generate audit records are suspended. The audit daemon goes into a loop, waiting for space to become available, and resumes processing audit records when that happens. While audit records are not being processed, no auditable activities occur. Every process that attempts to generate an audit record is suspended. For this reason, you should set up a separate audit administration account that could operate without any auditing enabled. Then, you could continue operations without being suspended.
- An internal error occurs. Possible internal errors include the following:
  - `ebusy` – Another audit daemon process is already running
  - `tmpfile` – A temporary file cannot be used
  - `auditsvc` – The `auditsvc()` system call fails
  - `postsigterm` – A signal was received during auditing shutdown

  Mail is sent to the `audit_warn` alias.
- A problem is discovered with the syntax of the `audit_control` file. By default, a message is sent to the console. Mail is also sent to the `audit_warn` alias.

Audit Administration Profiles

The Solaris operating environment provides profiles for configuring the audit service and for analyzing the audit trail.

- **Audit Control –** Profile that enables a role to configure Solaris auditing. The profile grants authorizations to configure auditing files and to run auditing commands. Roles with the Audit Control profile can run the following commands: `audit`, `auditd`, `auditconfig`, `bsmconv`, and `bsmunconv`.
- **Audit Review –** Profile that enables a role to analyze Solaris auditing records. The profile grants authorization to read audit records with the `praudit` and `auditreduce` commands. The role can also run the `auditstat` command.
- **System Administrator –** Profile that includes the Audit Review profile. A role with the System Administrator profile can analyze audit records.

To assign a profile to a role, see “Creating Roles” on page 105.

Audit Classes and Their Audit Flags

Audit flags indicate classes of events to audit. Machine-wide defaults for auditing are specified for all users on each machine by flags in the `audit_control` file. The file is described in "The `audit_control` File" on page 380.

You can modify what is audited for individual users by putting audit flags in a user's entry in the `audit_user` file. The audit flags are also used as arguments to the `auditconfig` command. See the `auditconfig`(1M) man page.

Definitions of Audit Flags

The following table shows each predefined audit class. The table shows the audit flag, the long name, and a short description. The audit flag is the short name that stands for the class. You use these audit flags in the auditing configuration files to specify which classes of events to audit. You also use them as arguments to auditing commands, such as `auditconfig`. You can define new classes by modifying the `audit_class` file. You can also rename existing classes. See the `audit_class`(4) man page for more information.

TABLE 23–1 Predefined Audit Flags

Short Name	Long Name	Short Description
`all`	`all`	All classes (meta-class)
`no`	`no_class`	Null value for turning off event preselection
`na`	`non_attrib`	Nonattributable events
`fr`	`file_read`	Read of data, open for reading
`fw`	`file_write`	Write of data, open for writing
`fa`	`file_attr_acc`	Access of object attributes: `stat`, `pathconf`
`fm`	`file_attr_mod`	Change of object attributes: `chown`, `flock`
`fc`	`file_creation`	Creation of object
`fd`	`file_deletion`	Deletion of object
`cl`	`file_close`	`close` system call
`ap`	`application`	Application-defined event
`ad`	`administrative`	Administrative actions (old administrative meta-class)

TABLE 23–1 Predefined Audit Flags *(Continued)*

Short Name	Long Name	Short Description
am	administrative	Administrative actions (meta-class)
ss	system state	Change system state
as	system-wide administration	System-wide administration
ua	user administration	User administration
aa	audit administration	Audit utilization
ps	process start	Process start and process stop
pm	process modify	Process modify
pc	process	Process (meta-class)
ex	exec	Program execution
io	ioctl	`ioctl` system call
ip	ipc	System V IPC operations
lo	login_logout	Login and logout events
nt	network	Network events: `bind`, `connect`, `accept`
ot	other	Miscellaneous

Audit Flag Syntax

The prefixes to the audit flags determine whether a class of events is audited for success, or for failure. Without a prefix, a class is audited for success and for failure. The following table shows the format of the audit flag and some possible representations.

TABLE 23–2 Plus and Minus Prefixes to Audit Flags

prefixflag	Explanation
lo	Audit all successful attempts to log in and log out, and all failed attempts to log in. You cannot fail an attempt to log out.
+lo	Audit all successful attempts to log in and log out.
-all	Audit all failed events.
+all	Audit all successful events.

Caution – The `all` flag can generate large amounts of data and fill up audit file systems quickly. Use the `all` flag only if you have extraordinary reasons to audit all activities.

Prefixes That Modify Audit Flags

Audit flags that were previously selected can be further modified by a caret prefix, `^`. The following table shows how the caret prefix modifies a preselected audit flag.

TABLE 23–3 Caret Prefix That Modifies Already-Specified Audit Flags

^prefixflag	**Explanation**
`-all,^-fc`	Audit all failed events, except do not audit failed attempts to create file system objects
`am,^+aa`	Audit all administrative events for success and for failure, except do not audit successful attempts to administer auditing
`am,^ua`	Audit all administrative events for success and for failure, except do not audit user administration events

The prefixes to the audit flags can be used in the following files and commands:

- In the `flags` line in the `audit_control` file
- In `flags` field in the user's entry in the `audit_user` file
- With arguments to the `auditconfig` command

See "The `audit_control` File" on page 380 for an example of using the prefixes in the `audit_control` file.

Audit Policies

Audit policies determine if additional information is added to the audit trail. The audit policies are described in "Determining Which Audit Policies to Use" on page 346.

Process Audit Characteristics

The following audit characteristics are set at initial login:

- **Process preselection mask –** A combination of the audit flags from the `audit_control` file and the `audit_user` file. When a user logs in, the `login` command combines the flags to establish the *process preselection mask* for the user's processes. The process preselection mask specifies whether events in each audit event class are to generate audit records.

  The algorithm for obtaining the process preselection mask is described in the following equation:

user's process preselection mask = (flags: line + always-audit flags) - never-audit flags

  Add the audit flags from the `flags:` line in the `audit_control` file to the flags from the *always-audit* field in the user's entry in the `audit_user` file. Then, subtract from the total the flags from the user's *never-audit* field.
- **Audit ID –** A process acquires an audit ID when the user logs in. The audit ID is inherited by all child processes that were started by the user's initial process. The audit ID helps enforce accountability. Even after a user becomes root, the audit ID remains the same. The audit ID that is saved in each audit record always allows you to trace actions back to the original user who had logged in.
- **Audit Session ID –** The audit session ID is assigned at login. The session ID is inherited by all descendant processes.
- **Terminal ID (port ID, machine ID) –** The terminal ID consists of the host name and the Internet address, followed by a unique number that identifies the physical device on which the user logged in. Most often, the login is through the console and the number that corresponds to the console device is `0`.

Audit Trail

The *audit trail* is created by the audit daemon. The audit daemon starts on each machine when the machine is brought up. After the `auditd` daemon starts at boot time, it is responsible for collecting the audit trail data and writing the audit records into *audit files*, which are also called *audit log files*. For a description of the file format, see the `audit.log`(4) man page. See also the `auditd`(1M) man page.

Even though you can physically locate audit directories within file systems that are not dedicated to auditing, do not do so except for directories of last resort. Directories of last resort are directories where audit files are written only when there is no other suitable directory available.

There is one other scenario where locating audit directories outside of dedicated audit file systems could be acceptable. You might do so in a software development environment where auditing is optional. To make full use of disk space might be more important than to keep an audit trail. However, in a security-conscious environment, audit directories within other file systems is not acceptable.

You should also consider the following factors.

- A host should have at least one local audit directory. The local directory can be used as a directory of last resort if the host is unable to communicate with the audit server.
- Mount audit directories with the read-write (`rw`) option. When you mount audit directories remotely, also use the `intr` and `noac` options.
- List the audit file systems on the audit server where they reside. The export list should include all machines in the configuration.

Naming Conventions for Audit Files

Each audit file is a self-contained collection of records. The file's name identifies the time span during which the records were generated and the machine that generated them.

Audit File Naming

Audit files that are complete have names of the following form:

start-time.finish-time.machine

where

- *start-time* – Is the time that the first audit record in the audit file was generated.
- *finish-time* – Is the time that the last record was written to the file.
- *machine* – Is the name of the machine that generated the file.

For an example of these names, see "Example of a Closed Audit File Name" on page 390.

An audit log file that is still active has a name of the following form:

start-time`.not_terminated.`*machine*

How Audit File Names Are Used

The time stamps in file names are used by the `auditreduce` command to locate records within a specific time range. These time stamps are important because there can be a month's accumulation or more of audit files online. To search all the files for records that were generated in the last 24 hours would be unacceptably expensive.

Time-Stamp Format and Interpretation

The *start-time* and *end-time* are timestamps with one-second resolution. They are specified in Greenwich Mean Time (GMT). The format is four digits for the year, followed by two digits for each month, day, hour, minute, and second, as follows:

YYYYMMDDHHMMSS

The timestamps are in GMT to ensure that they sort in proper order even across a daylight savings time boundary. Because they are in GMT, the date and hour must be translated to the current time zone to be meaningful. Beware of this point whenever you manipulate these files with standard file commands rather than with the `auditreduce` command.

Example of a File Name for a Still-Active File

The format of a file name of a still-active file is as follows:

YYYYMMDDHHMMSS`.not_terminated.`*machine*

Here is an example:

```
19990327225243.not_terminated.dopey
```

The audit log files are named by the beginning date. So, in the example above the audit file was created in 1999, on March 27, at 10:52:43 p.m, GMT. The `not_terminated` in the file name means either that the file is still active or that the `auditd` daemon was unexpectedly interrupted. The name `dopey` at the end is the host name of the machine whose audit data is being collected.

Example of a Closed Audit File Name

The format of the name of a closed audit log file is as follows:

YYYYMMDDHHMMSS.YYYYMMDDHHMMSS.hostname

Here is an example:

```
19990320005243.19900327225351.dopey
```

In this example, the audit log file was created in 1999, on March 20, at 12:52:43 a.m., GMT. The file was closed March 27, at 10:53:51 p.m., GMT. The name `dopey` at the end is the host name of the machine whose audit data was collected.

Whenever `auditd` is unexpectedly interrupted, the audit file that is open at the time retains the `not_terminated` file name designation. For example, when a machine is writing to a remotely mounted audit file, the file server can become inaccessible. When the mounted audit file cannot be reached, the `not_terminated` designation remains in the file's name. When service is restored, the audit daemon opens a new audit file and keeps the old audit file name intact.

Audit Record Structure

An audit record is a sequence of audit tokens. Each audit token contains event information such as user ID, time, and date. A `header` token begins an audit record, and an optional `trailer` token concludes the record. Other audit tokens contain information relevant to the auditable event. The following figure shows a typical audit record.

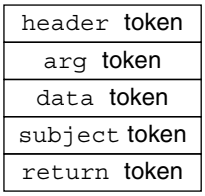


FIGURE 23–3 Typical Audit Record Structure

Audit Token Formats

Each token has a token type identifier followed by data that is specific to the token. Each token type has its own format. The following table shows the token names with a description of each token.

TABLE 23–4 Audit Tokens for the Basic Security Module

Token Name	Description	For More Information
`acl`	Access Control List information	"`acl` Token" on page 393
`arbitrary`	Data with format and type information	"`arbitrary` Token" on page 393
`arg`	System call argument value	"`arg` Token" on page 394
`attr`	file vnode tokens	"`attr` Token" on page 395
`exec_args`	Exec system call arguments	"`exec_args` Token" on page 395
`exec_env`	Exec system call environment variables	"`exec_env` Token" on page 396
`exit`	Program exit information	"`exit` Token" on page 397
`file`	Audit file information	"`file` Token" on page 397
`groups`	Process groups information	"`group` Token (Obsolete)" on page 398
`header`	Indicates start of audit record	"`header` Token" on page 398
`in_addr`	Internet address	"`in_addr` Token" on page 399
`ip`	IP header information	"`ip` Token (Obsolete)" on page 400
`ipc`	System V IPC information	"`ipc` Token" on page 400
`ipc_perm`	System V IPC object tokens	"`ipc_perm` Token" on page 401
`iport`	Internet port address	"`iport` Token" on page 402
`newgroups`	Process groups information	"`newgroups` Token" on page 402
`opaque`	Unstructured data (unspecified format)	"`opaque` Token" on page 403
`path`	Path information	"`path` Token" on page 403
`process`	Process token information	"`process` Token" on page 404
`return`	Status of system call	"`return` Token" on page 405
`seq`	Sequence number token	"`seq` Token" on page 406
`socket`	Socket type and addresses	"`socket` Token" on page 406
`subject`	Subject token information (same format as `process` token)	"`subject` Token" on page 407
`text`	ASCII string	"`text` Token" on page 408
`trailer`	Indicates end of audit record	"`trailer` Token" on page 409

An audit record always contains a `header` token. The `header` token indicates where the audit record begins in the audit trail. Every audit record contains a `subject` token, except for audit records from some nonattributable events. In the case of attributable events, these two tokens refer to the values of the process that caused the event. In the case of asynchronous events, the `process` tokens refer to the system.

`acl` Token

The `acl` token records information about Access Control Lists. This token consists of four fixed fields:

- A token ID that identifies this token as an `acl` token
- A field that specifies the ACL type
- An ACL ID field
- A field that lists the permissions associated with this ACL

The `praudit` command displays the `acl` token as follows:

```
acl,tpanero,staff,0755
```

The following figure shows the format of the `acl` token.

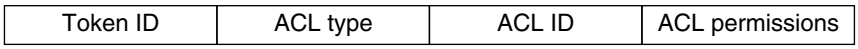


FIGURE 23–4 `acl` Token Format

`arbitrary` Token

The `arbitrary` token encapsulates data for the audit trail. This token consists of four fixed fields and an array of data. The fixed fields are as follows:

- A token ID that identifies this token as an `arbitrary` token
- A suggested format field, such as hexadecimal
- A size field that specifies the size of the data that is encapsulated, such as short
- A count field that provides the number of following items

The remainder of the token is composed of one or more items of the specified type. The `praudit` command displays the `arbitrary` token as follows:

```
arbitrary,decimal,int,1
42
```

The following figure shows the format of the `arbitrary` token.

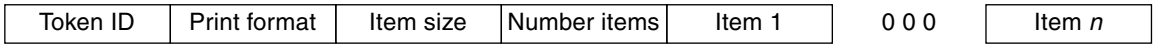


FIGURE 23–5 `arbitrary` Token Format

The following table shows the possible values of the print format field. Table 23–5.

TABLE 23–5 Values for the `arbitrary` Token's Print Format Field

Value	Action
`AUP_BINARY`	Prints the date in binary format
`AUP_OCTAL`	Prints the date in octal format
`AUP_DECIMAL`	Prints the date in decimal format
`AUP_HEX`	Prints the date in hexadecimal format
`AUP_STRING`	Prints the date as a string

The following table shows the possible values of the item size field.

TABLE 23–6 Values for the `arbitrary` Token's Item Size Field

Value	Action
`AUR_BYTE`	Data is printed in units of bytes in 1 byte
`AUR_SHORT`	Data is printed in units of shorts in 2 bytes
`AUR_LONG`	Data is printed in units of longs in 4 bytes

`arg` Token

The `arg` token contains information about the arguments to a system call: the argument number of the system call, the argument value, and an optional description. This token allows a 32-bit integer system-call argument in an audit record. The `arg` token has five fields:

- A token ID that identifies this token as an `arg` token
- An argument ID that tells which system call argument that the token refers to
- The argument value
- The length of the descriptive text string
- The text string

The `praudit` command displays the `arg` token as follows:

```
argument,1,0x00000000,addr
```

The following figure shows the format of the `arg` token.

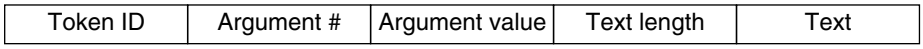


FIGURE 23–6 `arg` Token Format

`attr` Token

The `attr` token contains information from the file vnode. This token has seven fields:

- A token ID that identifies this token as an `attr` token
- The file access mode and type
- The owner user ID
- The owner group ID
- The file system ID
- The inode ID
- The device ID the file might represent

See the `statvfs`(2) man page for further information about the file system ID and the device ID.

The `attr` token usually accompanies a `path` token. The `attr` token is produced during path searches. In the event of a path-search error, there is no vnode available to obtain the necessary file information. Therefore, the `attr` token is not included as part of the audit record. The `praudit` command displays the `attr` token as follows:

```
attribute,100555,root,staff,1805,13871,-4288
```

The following figure shows the format of an `attr` token.

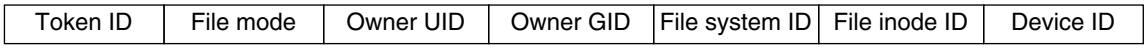


FIGURE 23–7 `attr` Token Format

`exec_args` Token

The `exec_args` token records the arguments to an `exec()` system call. The `exec_args` token has two fixed fields:

- A token ID field that identifies this token as an `exec_args` token
- A count that represents the number of arguments that are passed to the `exec()` system call

The remainder of this token is composed of zero or more null-terminated strings. The `praudit` command displays the `exec_args` token as follows:

```
vi,/etc/security/audit_user
```

The following figure shows the format of an `exec_args` token.

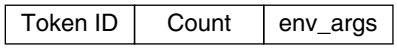


FIGURE 23–8 `exec_args` Token Format

Note – The `exec_args` token is output only when the audit policy `argv` is active.

`exec_env` Token

The `exec_env` token records the current environment variables to an `exec()` system call. The `exec_env` token has two fixed fields:

- A token ID field that identifies this token as an `exec_env` token
- A count that represents the number of arguments that are passed to the `exec()` system call

The remainder of this token is composed of zero or more null-terminated strings. The `praudit` command displays the `exec_env` token as follows:

```
exec_env,25,
GROUP=staff,HOME=/export/home/matrix,HOST=mestrix,HOSTTYPE=sun4u,HZ=100,
LC_COLLATE=en_US.ISO8859-1,LC_CTYPE=en_US.ISO8859-1,LC_MESSAGES=C,
LC_MONETARY=en_US.ISO8859-1,LC_NUMERIC=en_US.ISO8859-1,
LC_TIME=en_US.ISO8859-1,LOGNAME=matrix,MACHTYPE=sparc,
MAIL=/var/mail/matrix,OSTYPE=solaris,PATH=/usr/sbin:/usr/bin,PS1=#,
PWD=/var/audit,REMOTEHOST=192.168.13.5,SHELL=/usr/bin/csh,SHLVL=1,
TERM=dtterm,TZ=US/Pacific,USER=matrix,VENDOR=sun
```

The following figure shows the format of an `exec_env` token.

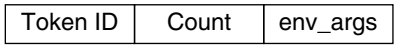


FIGURE 23–9 `exec_env` Token Format

Note – The `exec_env` token is output only when the audit policy `arge` is active.

`exit` Token

The `exit` token records the exit status of a program. The `exit` token contains the following fields:

- A token ID that identifies this token as an `exit` token
- A program exit status as passed to the `exit()` system call
- A return value that describes the exit status or that provides a system error number

The `praudit` command displays the `exit` token as follows:

```
exit,Error 0,0
```

The following figure shows the format of an `exit` token.

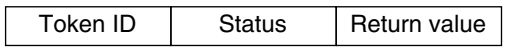


FIGURE 23–10 `exit` Token Format

`file` Token

The `file` token is a special token that is generated by the audit daemon. The token marks the beginning of a new audit file and the end of an old audit file as the old file is deactivated. The audit daemon builds a special audit record that contains this token to "link" together successive audit files into one audit trail. The `file` token has four fields:

- A token ID that identifies this token as a `file` token
- A time and date stamp that identifies the time that the file was created or was closed
- A byte count of the file name that includes a null terminator
- A field that holds the file null-terminated name

The `praudit` command displays the `file` token as follows:

```
file,Tue Sep  1 13:32:42 1992, + 79249 msec,
    /var/audit/localhost/files/19990901202558.19990901203241.quisp
```

The following figure shows the format of a `file` token.

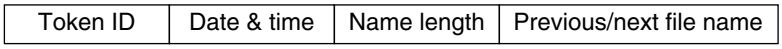


FIGURE 23–11 `file` Token Format

`group` Token (Obsolete)

This token has been replaced by the `newgroups` token, which provides the same type of information but requires less space. A description of the `group` token is provided here for completeness, but the application designer should use the `newgroups` token. Notice that `praudit` does not distinguish between the two tokens, as both token IDs are labeled `group` in `praudit` output.

The `group` token records the groups entries from the process's credential. The `group` token has two fixed fields:

- A token ID that identifies this token as a `group` token
- An array of group entries of size `NGROUPS_MAX` (16)

The remainder of the token consists of zero or more group entries. The `praudit` command displays the `group` token as follows:

```
group,staff,admin,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
```

The following figure shows the format of a `group` token.

FIGURE 23–12 `group` Token Format

Note – The `group` token is output only when the audit policy `group` is active.

`header` Token

The `header` token is special in that it marks the beginning of an audit record. The `header` token combines with the `trailer` token to bracket all the other tokens in the record. The `header` token has six fields:

- A token ID field that identifies this token as a `header` token
- A byte count of the total length of the audit record, including both the header and the trailer
- A version number that identifies the version of the audit record structure

- The audit event ID that identifies the type of audit event that the record represents
- The ID modifier that identifies special characteristics of the audit event
- The time and date that the record was created

On 64-bit systems, the `header` token is displayed with a 64-bit time stamp, in place of the 32-bit time stamp.

The `praudit` command displays the `header` token for a `ioctl()` system call as follows:

```
header,240,1,ioctl(2),es,Tue Sept  1 16:11:44 2001, + 270000 msec
```

The following figure shows the format of a `header` token.

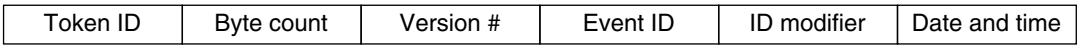


FIGURE 23–13 `header` Token Format

The ID modifier field has the following flags defined:

```
0x4000          PAD_NOTATTR                     nonattributable event
0x8000          PAD_FAILURE                     fail audit event
```

`in_addr` Token

The `in_addr` token contains an Internet Protocol address. Since the Solaris 8 release, the Internet address can be displayed in IPv4 format or IPv6 format. The IPv4 address uses 4 bytes. The IPv6 address uses 16 bytes to describe the type, and 16 bytes to describe the address. The `in_addr` token has two fields:

- A token ID that identifies this token as an `in_addr` token
- An Internet address

The `praudit` command displays the `in_addr` token as follows:

```
ip address,129.150.113.7
```

The following figure shows the format of an `in_addr` token.

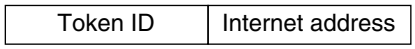


FIGURE 23–14 `in_addr` Token Format

`ip` Token (Obsolete)

The `ip` token contains a copy of an Internet Protocol header. The `ip` token has two fields:

- A token ID that identifies this token as an `ip` token
- A copy of the IP header, that is, all 20 bytes

The `praudit` command displays the `ip` token as follows:

```
ip address,0.0.0.0
```

The IP header structure is defined in the `/usr/include/netinet/ip.h` file. The following figure shows the format of an `ip` token.

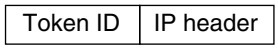


FIGURE 23–15 `ip` Token Format

`ipc` Token

The `ipc` token contains the System V IPC message/semaphore/shared-memory handle that is used by the caller to identify a particular IPC object. The `ipc` token has three fields:

- A token ID that identifies this token as an `IPC` token
- A type field that specifies the type of IPC object
- The handle that identifies the IPC object

The `praudit` command displays the `ipc` token as follows:

```
IPC,msg,3
```

Note – The IPC object identifiers violate the context-free nature of the Solaris audit tokens. No global "name" uniquely identifies IPC objects. Instead, IPC objects are identified by their handles. The handles are valid only during the time that the IPC objects are active. However, the identification of IPC objects should not be a problem. The System V IPC mechanisms are seldom used, and the mechanisms all share the same audit class.

The following table shows the possible values for the IPC object type field. The values are defined in the `/usr/include/bsm/audit.h` file.

TABLE 23–7 Values for the IPC Object Type Field

Name	Value	Description
`AU_IPC_MSG`	1	IPC message object
`AU_IPC_SEM`	2	IPC semaphore object
`AU_IPC_SHM`	3	IPC shared-memory object

The following figure shows the format of an `ipc` token.

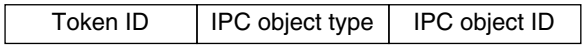


FIGURE 23–16 `ipc` Token Format

`ipc_perm` Token

The `ipc_perm` token contains a copy of the System V IPC access information. This token is added to audit records that are generated by IPC shared-memory events, IPC semaphore events, and IPC message events. The `ipc_perm` token has eight fields:

- A token ID that identifies this token as an `ipc_perm` token
- The user ID of the IPC owner
- The group ID of the IPC owner
- The user ID of the IPC creator
- The group ID of the IPC creator
- The access modes of the IPC
- The sequence number of the IPC
- The IPC key value

The `praudit` command displays the `ipc_perm` token as follows:

```
IPC perm,root,wheel,root,wheel,0,0,0x00000000
```

The values are taken from the `ipc_perm` structure that is associated with the IPC object. The following figure shows the format of an `ipc_perm` token.

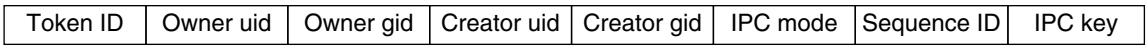


FIGURE 23–17 `ipc_perm` Token Format

`iport` Token

The `iport` token contains the TCP or UDP port address. The `iport` token has two fields:

- A token ID that identifies this token as an `iport` token
- The TCP or UDP port address

The `praudit` command displays the `iport` token as follows:

```
ip port,0xf6d6
```

The following figure shows the format of an `iport` token.

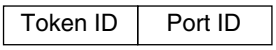


FIGURE 23–18 `iport` Token Format

`newgroups` Token

This token replaces the `group` token. Notice that the `praudit` command does not distinguish between the two tokens, as both token IDs are labeled `group` in `praudit` output.

The `newgroups` token records the group entries from the process's credential. The `newgroups` token has two fixed fields:

- A token ID field that identifies this token as a `newgroups` token
- A count that represents the number of groups that are contained in this audit record

The remainder of this token is composed of zero or more group entries. The `praudit` command displays the `newgroups` token as follows:

```
group, staff, admin
```

The following figure shows the format of a `newgroups` token.

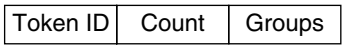


FIGURE 23–19 `newgroups` Token Format

Note – The `newgroups` token is output only when the `group` audit policy is active.

`opaque` Token

The `opaque` token contains unformatted data as a sequence of bytes. The `opaque` token has three fields:

- A token ID that identifies this token as an `opaque` token
- A byte count of the data
- An array of byte data

The `praudit` command displays the `opaque` token as follows:

```
opaque,12,0x4f5041515545204441544100
```

The following figure shows the format of an `opaque` token.

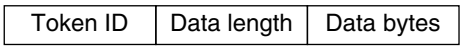


FIGURE 23–20 `opaque` Token Format

`path` Token

The `path` token contains access path information for an object. This token contains the following fields:

- A token ID that identifies this token as an `path` token
- A byte count of the path length
- The absolute path to the object that is based on the real root of the system

The `praudit` command displays the `path` token as follows. Note that the path length field is not displayed.

```
path,/etc/security/audit_user
```

The following figure shows the format of a `path` token.

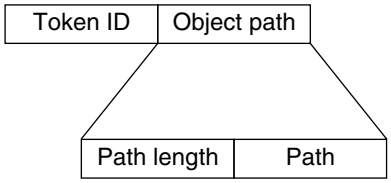


FIGURE 23–21 `path` Token Format

`process` Token

The `process` token contains information about a user who is associated with a process, such as the recipient of a signal. The `process` token has nine fields:

- A token ID that identifies this token as a `process` token
- The invariant audit ID
- The effective user ID
- The effective group ID
- The real user ID
- The real group ID
- The process ID
- The audit session ID
- A terminal ID that consists of a device ID and a machine ID

The audit ID, user ID, group ID, process ID, and session ID are long instead of short.

Note – The `process` token fields for the session ID, the real user ID, or the real group ID might be unavailable. The value is then set to -1.

Any token that contains a terminal ID has several variations. The `praudit` command hides these variations. So, the terminal ID is handled the same way for any token that contains a terminal ID. The terminal ID is either an IP address and port number, or a device ID. A device ID, such as the serial port that is connected to a modem, can be zero. The terminal ID is specified in one of several formats.

The terminal ID for device numbers is specified as follows:

- **32-bit applications –** 4-byte device number, 4-bytes unused
- **64-bit applications –** 8-byte device number, 4-bytes unused

The terminal ID for port numbers in releases that are earlier than the Solaris 8 release is specified as follows:

- **32-bit applications –** 4-byte port number, 4-byte IP address
- **64-bit applications –** 8-byte port number, 4-byte IP address

The terminal ID for port numbers in the Solaris 8 release or the Solaris 9 release is specified as follows:

- **32-bit with IPv4 –** 4-byte port number, 4-byte IP type, 4-byte IP address
- **32-bit with IPv6 –** 4-byte port number, 4-byte IP type, 16-byte IP address
- **64-bit with IPv4 –** 8-byte port number, 4-byte IP type, 4-byte IP address
- **64-bit with IPv6 –** 8-byte port number, 4-byte IP type, 16-byte IP address

The `praudit` command displays the `process` token as follows:

```
process,root,root,wheel,root,wheel,0,0,0,0.0.0.0
```

The following figure shows the format of a `process` token.

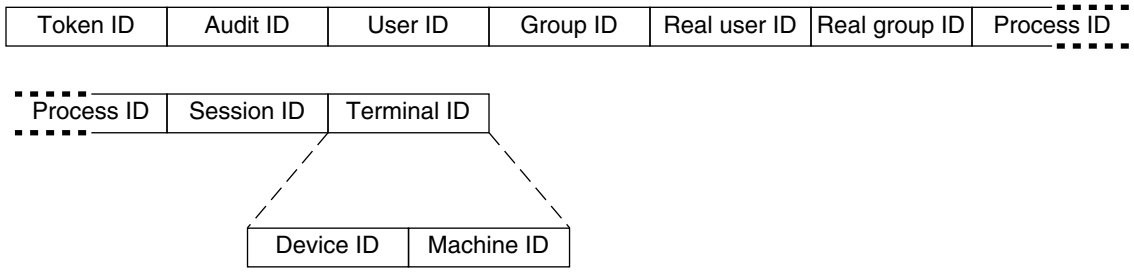


FIGURE 23–22 `process` Token Format

`return` Token

The `return` token contains the return status of the system call (`u_error`) and the process return value (`u_rval1`). This token has three fields:

- A token ID that identifies this token as a `return` token
- The error status of the system call
- The return value of the system call

The `return` token is always returned as part of kernel-generated audit records for system calls. This token indicates exit status and other return values in application auditing.

The `praudit` command displays the `return` token as follows:

```
return,success,0
```

The following figures shows the format of a `return` token.

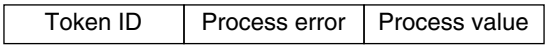


FIGURE 23–23 `return` Token Format

`seq` Token

The sequence token, `seq`, is an optional token that contains a sequence number. Used for debugging, this token is added to each audit record when the `seq` policy is active. The `seq` token has two fields:

- A token ID that identifies this token as a `seq` token
- A 32-bit unsigned long field that contains the sequence number

The sequence number is incremented every time an audit record is generated and added to the audit trail. The `praudit` command displays the `seq` token as follows:

```
sequence,1292
```

The following figure shows the format of a `seq` token.

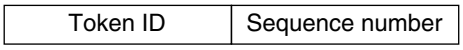


FIGURE 23–24 `seq` Token Format

Note – The `seq` token is output only when the `seq` audit policy is active.

`socket` Token

The `socket` token contains information that describes an Internet socket. This token has six fields:

- A token ID that identifies this token as a `socket` token
- A socket type field that indicates the type of socket referenced, one of TCP, UDP, or UNIX
- The local port address
- The local Internet address
- The remote port address
- The remote Internet address

The `praudit` command displays the `socket` token as follows:

```
socket,0x0000,0x0000,0.0.0.0,0x0000,0.0.0.0
```

Since the Solaris 8 release, the Internet address can be displayed in IPv4 format or IPv6 format. The IPv4 address uses 4 bytes. The IPv6 address uses 16 bytes to describe the type, and 16 bytes to describe the address. The following figure shows the format of a `socket` token.

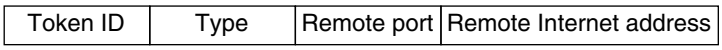


FIGURE 23–25 `socket` Token Format

`subject` Token

The `subject` token describes a user who performs or attempts to perform an operation. The format is the same as the `process` token. The `subject` token has nine fields:

- An ID that identifies this token as a `subject` token
- The invariant audit ID
- The effective user ID
- The effective group ID
- The real user ID
- The real group ID
- The process ID
- The audit session ID
- A terminal ID that consists of a device ID and a machine ID

The audit ID, user ID, group ID, process ID, and session ID are long instead of short.

Note – The `subject` token fields for the session ID, the real user ID, or the real group ID might be unavailable. The value is then set to -1.

Any token that contains a terminal ID has several variations. The `praudit` command hides these variations. So, the terminal ID is handled the same way for any token that contains a terminal ID. The terminal ID is either an IP address and port number, or a device ID. A device ID, such as the serial port that is connected to a modem, can be zero. The terminal ID is specified in one of several formats.

The terminal ID for device numbers is specified as follows:

- **32-bit applications –** 4-byte device number, 4-bytes unused
- **64-bit applications –** 8-byte device number, 4-bytes unused

The terminal ID for port numbers in releases that are earlier than the Solaris 8 release is specified as follows:

- **32-bit applications –** 4-byte port number, 4-byte IP address
- **64-bit applications –** 8-byte port number, 4-byte IP address

The terminal ID for port numbers in the Solaris 8 release or the Solaris 9 release is specified as follows:

- **32-bit with IPv4 –** 4-byte port number, 4-byte IP type, 4-byte IP address
- **32-bit with IPv6 –** 4-byte port number, 4-byte IP type, 16-byte IP address
- **64-bit with IPv4 –** 8-byte port number, 4-byte IP type, 4-byte IP address
- **64-bit with IPv6 –** 8-byte port number, 4-byte IP type, 16-byte IP address

The `subject` token is always returned as part of kernel-generated audit records for system calls. The `praudit` command displays the `subject` token as follows:

```
subject,cjc,cjc,staff,cjc,staff,424,223,0 0 quisp
```

The following figure shows the format of the `subject` token.

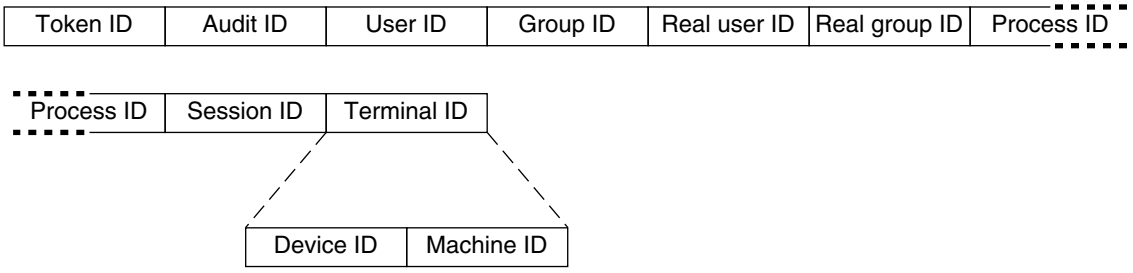


FIGURE 23–26 `subject` Token Format

`text` Token

The `text` token contains a text string. This token has three fields:

- A token ID that identifies this token as a `text` token
- The length of the text string
- The text string itself

The `praudit` command displays the `text` token as follows:

```
text,aw_test_token
```

The following figure shows the format of a `text` token.

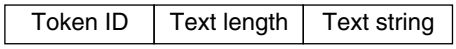


FIGURE 23–27 `text` Token Format

`trailer` Token

The two tokens, `header` and `trailer`, are special in that they distinguish the end points of an audit record and bracket all the other tokens. A `header` token begins an audit record. A `trailer` token ends an audit record. The `trailer` token is an optional token. The `trailer` token is added as the last token of each record only when the `trail` audit policy has been set.

If an audit record was generated with trailers turned on, the `auditreduce` command verifies that the trailer points back to the record header correctly. The `trailer` token supports backward seeks of the audit trail.

The `trailer` token has three fields:

- A token ID that identifies this token as a `trailer` token
- A pad number to aid in marking the end of the record
- The total number of characters in the audit record, including both the `header` and `trailer` tokens

The `praudit` command displays the `trailer` token as follows:

```
trailer,136
```

The following figure shows the format of a `trailer` token.

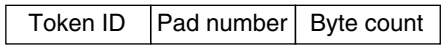


FIGURE 23–28 `trailer` Token Format

Device Allocation Reference

Device allocation protects removable media from unauthorized use. You can require that a user allocate a device. You can deny a user permission to use a device. Such allocation measures can protect your site from loss of data, computer viruses, and other security breaches. The following section provides information about device allocation.

Components of the Device-Allocation Mechanism

The components of the device-allocation mechanism are as follows:

- The `allocate`, `deallocate`, `dminfo`, and `list_devices` commands. For more information, see "Using the Device Allocation Commands" on page 410.
- The `/etc/security/device_allocate` file. See the `device_allocate`(4) man page.
- The `/etc/security/device_maps` file. See the `device_maps`(4) man page.
- A lock file in the `/etc/security/dev` directory for each allocatable device.
- The changed attributes of the *device-special files* that are associated with each allocatable device.
- Device-clean scripts for each allocatable device.

The `device_allocate` file, the `device_maps` file, and the lock files are local configuration files. These files are not administered as name service databases because tape drives, diskette drives, and printers connect to specific machines.

Using the Device Allocation Commands

This section describes some of the options to the `allocate`, `deallocate`, and `list_devices` commands that are for use by administrators. Only `root` or a role of equivalent power can access these options. The commands are detailed on their respective man pages.

TABLE 23–8 Administrative Options to the Device Allocation Commands

Command With Option	Description
`allocate -F` *device_special_filename*	Reallocates the specified device. This option is often used with the `-U` option to reallocate the specified device to the specified user. Without the `-U` option, the device is allocated to `root`.
`allocate -U` *username*	Causes the device to be allocated to the user who is specified rather than to the current user. This option allows you to allocate a device for another user, without having to assume that user's identity.
`deallocate -F` *device_special_filename*	Forces the deallocation of a device. Devices that a user has allocated are not automatically deallocated when the process terminates or when the user logs out. When a user forgets to deallocate a tape drive, you can force deallocation by using the `-F` option.

TABLE 23–8 Administrative Options to the Device Allocation Commands *(Continued)*

Command With Option	Description
`deallocate -I`	Forces the deallocation of all allocatable devices. This option should be used only at system initialization.
`list_devices`	Lists all the device-special files that are associated with any device that is listed in the `device_maps` file.
`list_devices -U` *username*	Lists the devices that are allocatable or allocated to the user ID that is associated with the specified user name. This option allows you to check which devices are allocatable or allocated to another user.

The Allocate Error State

An allocatable device is in the *allocate error state* if it is owned by user `bin` and group `bin` with a device-special file mode of 0100. If a user wants to allocate a device that is in the allocate error state, you can try to force the deallocation of the device. The `deallocate` command with the `-F` option forces deallocation. Or, you can use `allocate -U` to assign the device to the user. Once the device is allocated, you can investigate any error messages that appear. After any problems with the device are corrected, you must use the force option, `-F` to clear the allocate error state from the device.

The `device_maps` File

You can examine the `/etc/security/device_maps` file to determine the device names, device types, and device-special files that are associated with each allocatable device. See the `device_maps`(4) man page. Device maps are created when you set up device allocation. A rudimentary `device_maps` file is created by `bsmconv` when the BSM is enabled. This initial `device_maps` file should be used only as a starting point. You can then augment and customize the `device_maps` file for your site.

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

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

device-name:*device-type*:*device-list*

Lines in the `device_maps` file can end with a backslash (\) to continue an entry on the next line. Comments can also be included. A "#" makes a comment out of all subsequent text until the next newline not immediately preceded by a backslash. Leading and trailing blanks are allowed in any field.

TABLE 23–9 Description of Fields in a `device_maps` Entry

Field	Description
device-name	Specifies the name of the device, for example `st0`, `fd0`, or `audio`. The device name that is specified here must correspond to the name of the lock file that is used in the `/etc/security/dev` directory.
device-type	Specifies the generic device type. The generic name is the name for the class of devices, such as `st`, `fd`, and `audio`. The *device-type* field logically groups related devices.
device-list	Lists of the device-special files that are associated with the physical device. The *device-list* must contain *all* of the special files that allow access to a particular device. If the list is incomplete, a malevolent user can still obtain or modify private information. Valid entries for the *device-list* field are either the real device files located under `/devices` or the symbolic links that are in `/dev`. The symbolic links in the `/dev` directory are provided for binary compatibility.

The following is an example of entries in a `device_maps` file for SCSI tape `st0` and diskette `fd0`.

```
fd0:\
    fd:\
    /dev/fd0 /dev/fd0a /dev/fd0b /dev/rfd0 /dev/rfd0a /dev/rfd0b:\
                      .
                      .
                      .
st0:\
    st:\
    /dev/rst0 /dev/rst8 /dev/rst16 /dev/nrst0 /dev/nrst8 /dev/nrst16:\
```

The `device_allocate` File

You can modify the `device_allocate` file to change devices from allocatable to nonallocatable, or to add new devices. A sample `device_allocate` file follows.

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

You define which devices should be allocatable during initial BSM configuration. You can decide to accept the default devices and their defined characteristics, as shown in the preceding sample `device_allocate` file. Whenever you add a device to any machine after the system is up and running, you must decide whether to make the new device allocatable.

After installation, you can modify the entries for devices in the `device_allocate` file. Any device that needs to be allocated before use must be defined in the `device_allocate` file on each machine. Currently, cartridge tape drives, diskette drives, CD-ROM devices, and audio chips are considered allocatable. These device types have device-clean scripts.

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

An entry in the `device_allocate` file does not mean that the device is allocatable, unless the entry specifically states that the device is allocatable. In the sample `device_allocate` file, note the asterisk (*) in the fifth field of the audio device entry. An asterisk in the fifth field indicates to the system that the device is not allocatable. That is, the system administrator does not require a user to allocate the device before it is used nor to deallocate it afterward. Any other string placed in this field indicates that the device is allocatable.

In the `device_allocate` file, you represent each device by a one-line entry of the form:

device-name`;`*device-type*`;reserved;reserved;`*alloc*`;`*device-clean*

For example, the following line shows the entry for device name `st0`:

```
st0;st;;;;;/etc/security/lib/st_clean
```

Lines in the `device_allocate` file can end with a "\" to continue an entry on the next line. Comments can also be included. A "#" makes a comment out of all subsequent text until the next newline not immediately preceded by a "\". Leading and trailing blanks are allowed in any field.

The following table describes each field in the `device_allocate` file.

TABLE 23–10 Description of Fields in a `device_allocate` Entry

Field	Description
device-name	Specifies the name of the device, for example, `st0`, `fd0`, or `sr0`. When you make a device allocatable, retrieve the *device-name* from the *device-name* field in the `device_maps` file. You can also use the `dminfo` command. Note that the name is also the DAC file name for the device.
device-type	Specifies the generic device type. The generic name is the name for the class of devices, such as `st`, `fd`, and `sr`. This field groups related devices. When you make an allocatable device, retrieve the *device-type* from the *device-type* field in the `device_maps` file, or use the `dminfo` command.
`reserved`	Sun reserves the two fields that are marked `reserved` for future use.
alloc	Specifies whether the device is allocatable. An asterisk (*) in this field indicates that the device is *not* allocatable. Any other string, or an empty field, indicates that the device is allocatable.
device-clean	Supplies the path name of a script to be invoked for special handling, such as cleanup and object-reuse protection during the allocation process. The *device-clean* script is run any time that the device is acted on by the `deallocate` command, such as when a device is forcibly deallocated with `deallocate -F`.

Device-Clean Scripts

The *device-clean* scripts address the security requirement that all usable data be purged from a physical device before reuse. By default, cartridge tape drives, diskette drives, CD-ROM devices, and audio devices require device-clean scripts, which are provided. This section describes what device-clean scripts do.

Object Reuse

Device allocation satisfies part of the object-reuse requirement. The device-clean scripts make sure that data that is left on a device by one user is cleared. The data is cleared before the device is allocatable by another user.

Device-Clean Script for Tapes

The `st_clean` device-clean script supports three tape devices. The supported tape devices are as follows:

- SCSI 1/4-inch tape
- Archive 1/4-inch tape
- Open-reel 1/2-inch tape

The `st_clean` script uses the `rewoffl` option to the `mt` command to affect the device cleanup. For more information, see the `mt`(1) man page. If the script runs during system boot, the script queries the device. The script determines if the device is online.

If the device is online, the script determines if the device has media in it. The 1/4-inch tape devices that have media in them are placed in the allocate error state. The allocate error state forces the administrator to clean up the device manually.

During normal system operation, when the `allocate` or `deallocate` command is executed in interactive mode, the user is prompted to remove the media. The script pauses until the media is removed from the device.

Device-Clean Scripts for Diskettes and CD-ROM Devices

The following table shows the device-clean scripts for diskettes and CD-ROM devices.

TABLE 23–11 Device-Clean Scripts for Diskettes and CD-ROM Devices

Disk Device Type	Device-Clean Script
Diskette	`fd_clean`
CD-ROM	`sr_clean`

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

Device-Clean Script for Audio

Audio devices are cleaned up with an `audio-clean` script. The script performs an `AUDIO_DRAIN` ioctl system call to flush the device. The script then performs an `AUDIO_SETINFO` ioctl system call to reset the device configuration to the default. In addition, the script retrieves the audio chip registers by using the `AUDIOGETREG` ioctl system call. Any registers that deviate from the default settings are reset by using the `AUDIOSETREG` ioctl system call.

Writing New Device-Clean Scripts

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

`st_clean -[I|F|S]` *device-name*

Device-clean scripts must return “0” for success and greater than “0” for failure. The options `-I`, `-F`, and `-S` help the script determine its running mode. The following table describes the options.

TABLE 23–12 Options for Device-Clean Scripts

Option	Description
`-I`	The `-I` option is needed during system boot only. All output must go to the system console. Failure or inability to forcibly eject the media must put the device in the allocate error state.
`-F`	The `-F` option is for forced cleanup. The option is interactive. The option assumes that the user is available to respond to prompts. A script with this option must attempt to complete the cleanup if one part of the cleanup fails.
`-S`	The `-S` option is for standard cleanup. The option is interactive. The option assumes that the user is available to respond to prompts.

How the Device Allocation Mechanism Works

This section gives an example of how the device-allocate mechanism works.

The `allocate` command first checks for the presence of a lock file under the device name for the specified device in the `/etc/security/dev` directory. If the file is owned by `allocate`, then the ownership of the lock file is changed to the name of the user who initiated the `allocate` command.

The `allocate` command then checks for an entry for the device in the `device_allocate` file. The command further checks that the entry shows that the device as allocatable.

The first listing in the following example shows that a lock file exists with owner `bin`, group `bin`, and mode `600` for the `st0` device in `/etc/security/dev`. The second listing shows that the associated device-special files are set up properly, with owner `bin`, group `bin`, and mode `000`.

```
untouchable% ls -lg /etc/security/dev/st0
-rw------- 1 bin bin                    0 Dec 6 15:21 /etc/security/dev/st0
untouchable% ls -lg /devices/sbus@1,f8000000/esp@0,800000
c--------- 1 bin bin               18,  4 May 12 13:11 st@4,0:
c--------- 1 bin bin           18, 20 May 12 13:11 st@4,0:b
c--------- 1 bin bin           18, 28 May 12 13:11 st@4,0:bn
c--------- 1 bin bin           18, 12 May 12 13:11 st@4,0:c
                     .
                     .
                     .
c--------- 1 bin bin           18,  0 May 12 13:11 st@4,0:u
c--------- 1 bin bin           18, 16 May 12 13:11 st@4,0:ub
c--------- 1 bin bin           18, 24 May 12 13:11 st@4,0:ubn
c--------- 1 bin bin           18,  8 May 12 13:11 st@4,0:un
```

In this example, the user `vanessa` allocates device `st0`.

```
untouchable% whoami
vanessa
```

```
untouchable% allocate st0
```

When the user vanessa runs the allocate command to allocate the tape st0, allocate first checks for the existence of an /etc/security/dev/st0 file. If no lock file exists or if the lock file is owned by a user other than allocate, then user vanessa could not allocate the device.

If the allocate command finds the lock file for the device with the correct ownership and permissions, the command then checks to make sure that the device has an entry in the device_allocate file. The command also checks that the entry specifies that the device is allocatable.

In this example, the default device_allocate entry for the st0 device specifies that the device is allocatable. Because the allocate command finds that all these conditions are met, the device is allocated to user vanessa.

The allocate command changes the ownership and permissions of the device-special files that are associated with the device in the /dev directory. To allocate the st0 device to the user vanessa, the mode on its associated device-special files is changed to 600 and the owner is changed to vanessa.

The allocate command also changes the ownership of the lock file that is associated with the device in the /etc/security/dev directory. To allocate the st0 device to the user vanessa, the owner of /etc/security/dev/st0 is changed to vanessa.

In the following example, after the user vanessa executes the allocate command with the device name st0, the owner of /etc/security/dev/st0 is changed to vanessa and the owner of the associated device-special files is now also vanessa. Lastly, user vanessa now has permission to read and write the files.

```
untouchable% whoami
vanessa
untouchable% allocate st0
untouchable% ls -lg /etc/security/dev/st0
-rw------- 1 vanessa staff              0 Dec 6 15:21 /etc/security/dev/st0
untouchable% ls -la /devices/sbus@1,f8000000/esp@0,800000
.
.
.
crw------- 1 vanessa 18,  4 May 12 13:11 st@4,0:
crw------- 1 vanessa 18, 12 May 12 13:11 st@4,0:b
crw------- 1 vanessa 18, 12 May 12 13:11 st@4,0:bn
crw------- 1 vanessa 18, 12 May 12 13:11 st@4,0:c
.
.
.
crw------- 1 vanessa 18,  4 May 12 13:11 st@4,0:u
crw------- 1 vanessa 18, 12 May 12 13:11 st@4,0:ub
crw------- 1 vanessa 18, 12 May 12 13:11 st@4,0:ubn
crw------- 1 vanessa 18, 12 May 12 13:11 st@4,0:un
```

APPENDIX **A**

System Administration Guide: Security Services Updates

The following sections list the updates to Solaris 9 operating environment features that are described in this book.

Solaris 9 12/02 Updates

- New password encryption modules are described in "Maintaining Login Control" on page 30.

  How to specify a module for encrypting passwords is described in "Changing the Default Algorithm for Password Encryption" on page 53.
- Password policy is enforced on the Sun ONE Directory Server. The change is described in "Managing Password Information" on page 31.
- A new binding control flag in PAM is described in "PAM Control Flags" on page 181.

Solaris 9 8/03 Updates

- Public files are no longer audited for read-only events. The `-public` option to the `auditconfig` command controls whether public files are audited. By not auditing public files, the audit trail is greatly reduced. Attempts to read sensitive files are therefore easier to monitor. See "BSM Terminology" on page 337.
- The `praudit` command has an additional output format, XML. The XML format enables the output to be read in a browser, and provides source for XML scripting for reports. The `-x` option to the `praudit` command is described in "The `praudit` Command" on page 377.

- The default set of audit classes has been restructured. Audit meta-classes provide an umbrella for finer-grained audit classes. See "Definitions of Audit Flags" on page 385.
- The `bsmconv` command no longer disables the use of the Stop-A key. The Stop-A event is now audited to maintain security.

Glossary

admin principal A user principal with a name of the form *username*/`admin` (as in `joe/admin`). An admin principal can have more privileges (for example, to change policies) than a regular user principal. See also principal name, user principal.

application server See network application server.

authentication The process of verifying the claimed identity of a principal.

authenticator Authenticators are passed by clients when requesting tickets (from a KDC) and services (from a server). They contain information that is generated by using a session key known only by the client and server, that can be verified as of recent origin, thus indicating that the transaction is secure. When used with a ticket, an authenticator can be used to authenticate a user principal. An authenticator includes the principal name of the user, the IP address of the user's host, and a timestamp. Unlike a ticket, an authenticator can be used only once, usually when access to a service is requested. An authenticator is encrypted by using the session key for that client and that server.

authorization 1. The process of determining if a principal can use a service, which objects the principal is allowed to access, and the type of access that is allowed for each object.

2. In role-based access control (RBAC), a permission that can be assigned to a role or user (or embedded in a rights profile) for performing a class of actions that are otherwise prohibited by security policy.

client Narrowly, a process that makes use of a network service on behalf of a user; for example, an application that uses `rlogin`. In some cases, a server can itself be a client of some other server or service.

More broadly, a host that a) receives a Kerberos credential, and b) makes use of a service that is provided by a server.

	Informally, a principal that makes use of a service.
client principal	(RPCSEC_GSS API) A client (a user or an application) that uses RPCSEC_GSS-secured network services. Client principal names are stored in the form of `rpc_gss_principal_t` structures.
clock skew	The maximum amount of time that the internal system clocks on all hosts that are participating in the Kerberos authentication system can differ. If the clock skew is exceeded between any of the participating hosts, requests are rejected. Clock skew can be specified in the `krb5.conf` file.
confidentiality	See privacy.
credential	An information package that includes a ticket and a matching session key. Used to authenticate the identity of a principal. See also ticket, session key.
credential cache	A storage space (usually a file) that contains credentials that are received from the KDC.
flavor	Historically, *security flavor* and *authentication flavor* had the same meaning, as a flavor that indicated a type of authentication (AUTH_UNIX, AUTH_DES, AUTH_KERB). RPCSEC_GSS is also a security flavor, even though it provides integrity and privacy services in addition to authentication.
forwardable ticket	A ticket that a client can use to request a ticket on a remote host without requiring the client to go through the full authentication process on that host. For example, if the user `david` obtains a forwardable ticket while on user `jennifer`'s machine, he can log in to his own machine without being required to get a new ticket (and thus authenticate himself again). See also proxiable ticket.
FQDN	Fully qualified domain name. For example, `denver.mtn.example.com` (as opposed to simply `denver`).
GSS-API	The Generic Security Service Application Programming Interface. A network layer that provides support for various modular security services (including SEAM). GSS-API provides for security authentication, integrity, and privacy services. See also authentication, integrity, privacy.
host	A machine that is accessible over a network.
host principal	A particular instance of a service principal in which the principal (signified by the primary name `host`) is set up to provide a range of network services, such as `ftp`, `rcp`, or `rlogin`. An example of a host principal is `host/boston.eng.example.com@ENG.EXAMPLE.COM`. See also server principal.
initial ticket	A ticket that is issued directly (that is, not based on an existing ticket-granting ticket). Some services, such as applications that change

passwords, might require tickets to be marked *initial* so as to assure themselves that the client can demonstrate a knowledge of its secret key. This assurance is important because an initial ticket indicates that the client has recently authenticated itself (instead of relying on a ticket-granting ticket, which might existed for a long time).

instance The second part of a principal name, an instance qualifies the principal's primary. In the case of a service principal, the instance is required and is the host's fully qualified domain name, as in `host/boston.eng.example.com`. For user principals, an instance is optional. Note, however, that `joe` and `joe/admin` are unique principals. See also primary, principal name, service principal, user principal.

integrity A security service that, in addition to user authentication, provides for the validity of transmitted data through cryptographic checksumming. See also authentication, privacy.

invalid ticket A postdated ticket that has not yet become usable. An invalid ticket is rejected by an application server until it becomes validated. To be validated, an invalid ticket must be presented to the KDC by the client in a TGS request, with the `VALIDATE` flag set, after its start time has passed. See also postdated ticket.

KDC Key Distribution Center. A machine that has three Kerberos V5 components:

- Principal and key database
- Authentication service
- Ticket-granting service

Each realm has a master KDC and should have one or more slave KDCs.

Kerberos An authentication service, the protocol that is used by that service, or the code that is used to implement that service.

SEAM is an authentication implementation that is closely based on Kerberos V5.

While technically different, "SEAM" and "Kerberos" are often used interchangeably in SEAM documentation. The same is true for "Kerberos" and "Kerberos V5."

Kerberos (also spelled Cerberus) was a fierce, three-headed mastiff who guarded the gates of Hades in Greek mythology.

key 1. An entry (principal name) in a keytab file. See also keytab file.

2. An encryption key, of which there are three types:

- A *private key* – An encryption key that is shared by a principal and the KDC, and distributed outside the bounds of the system. See also private key.
- A *service key* – This key serves the same purpose as the private key, but is used by servers and services. See also service key.
- A *session key* – A temporary encryption key that is used between two principals, with a lifetime limited to the duration of a single login session. See also session key.

keytab file A key table file that contains one or more keys (principals). A host or service uses a keytab file in the much the same way that a user uses a password.

kvno Key version number. A sequence number that tracks a particular key in order of generation. The highest kvno is the latest and most current key.

name service scope The scope in which a role is permitted to operate, that is, an individual host or all hosts that are served by a specified name service such as NIS, NIS+, or LDAP. Scopes are applied to Solaris Management Console toolboxes.

master KDC The main KDC in each realm, which includes a Kerberos administration server, `kadmind`, and an authentication and ticket-granting daemon, `krb5kdc`. Each realm must have at least one master KDC, and can have many duplicate, or slave, KDCs that provide authentication services to clients.

mechanism A software package that specifies cryptographic techniques to achieve data authentication or confidentiality. Examples: Kerberos V5, Diffie-Hellman public key.

network application server A server that provides a network application, such as `ftp`. A realm can contain several network application servers.

NTP Network Time Protocol. Software from the University of Delaware that enables you to manage precise time or network clock synchronization, or both, in a network environment. You can use NTP to maintain clock skew in a Kerberos environment. See also clock skew.

PAM Pluggable Authentication Module. A framework that allows for multiple authentication mechanisms to be used without having to recompile the services that use them. PAM enables SEAM session initialization at login.

policy A set of rules, initiated when SEAM is installed or administered, that govern ticket usage. Policies can regulate principals' accesses, or ticket parameters, such as lifetime.

postdated ticket A postdated ticket does not become valid until some specified time after its creation. Such a ticket is useful, for example, for batch jobs that

are intended to run late at night, since the ticket, if stolen, cannot be used until the batch job is run. When a postdated ticket is issued, it is issued as *invalid* and remains that way until a) its start time has passed, and b) the client requests validation by the KDC. A postdated ticket is normally valid until the expiration time of the ticket-granting ticket. However, if the postdated ticket is marked *renewable*, its lifetime is normally set to be equal to the duration of the full life time of the ticket-granting ticket. See also invalid ticket, renewable ticket.

primary The first part of a principal name. See also instance, principal name, realm.

principal 1. A uniquely named client/user or server/service instance that participates in a network communication. Kerberos transactions involve interactions between principals (service principals and user principals) or between principals and KDCs. In other words, a principal is a unique entity to which Kerberos can assign tickets. See also principal name, service principal, user principal.

2. (RPCSEC_GSS API) See client principal, server principal.

principal name 1. The name of a principal, in the format *primary/instance@REALM*. See also instance, primary, realm.

2. (RPCSEC_GSS API) See client principal, server principal.

privacy A security service, in which transmitted data is encrypted before being sent. Privacy also includes data integrity and user authentication. See also authentication, integrity, service.

private key A key that is given to each user principal, and known only to the user of the principal and to the KDC. For user principals, the key is based on the user's password. See also key.

private-key encryption In private-key encryption, the sender and receiver use the same key for encryption. See also public-key encryption.

privileged application An application that can override system controls and that checks for specific UIDs, GIDs, or authorizations.

profile shell A shell used in RBAC that enables a role (or user) to run any privileged applications that are assigned to the role's rights profiles from the command line. The profile shells are `pfsh`, `pfcsh`, and `pfksh`. They correspond to the Bourne shell (`sh`), C shell (`csh`), and Korn shell (`ksh`), respectively.

proxiable ticket A ticket that can be used by a service on behalf of a client to perform an operation for the client. Thus, the service is said to act as the client's proxy. With the ticket, the service can take on the identity of the client. The service can use a proxiable ticket to obtain a service ticket to another service, but it cannot obtain a ticket-granting ticket. The

	difference between a proxiable ticket and a forwardable ticket is that a proxiable ticket is only valid for a single operation. See also forwardable ticket.
public-key encryption	An encryption scheme in which each user has two keys, one public key and one private key. In public-key encryption, the sender uses the receiver's public key to encrypt the message, and the receiver uses a private key to decrypt it. SEAM is a private-key system. See also private-key encryption.
QOP	Quality of Protection. A parameter that is used to select the cryptographic algorithms that are used in conjunction with the integrity service or privacy service.
RBAC	Role-Based Access Control. An alternative to the all-or-nothing superuser model. RBAC lets an organization separate superuser's capabilities and assign them to special user accounts called roles. Roles can be assigned to specific individuals, according to their job needs.
realm	1. The logical network that is served by a single SEAM database and a set of Key Distribution Centers (KDCs). 2. The third part of a principal name. For the principal name `joe/admin@ENG.EXAMPLE.COM`, the realm is `ENG.EXAMPLE.COM`. See also principal name.
relation	A configuration variable or relationship that is defined in the `kdc.conf` or `krb5.conf` files.
renewable ticket	Because having tickets with very long lives is a security risk, tickets can be designated as *renewable*. A renewable ticket has two expiration times: a) the time at which the current instance of the ticket expires, and b) maximum lifetime for any ticket. If a client wants to continue to use a ticket, the client renews the ticket before the first expiration occurs. For example, a ticket can be valid for one hour, with all tickets having a maximum lifetime of ten hours. If the client that holds the ticket wants to keep it for more than an hour, the client must renew the ticket. When a ticket reaches the maximum ticket lifetime, it automatically expires and cannot be renewed.
rights profile	Also referred to as right or profile. A collection of overrides used in RBAC that can be assigned to a role or user. A rights profile can consist of authorizations, commands with set UIDs or GIDs, which are referred to as security attributes, and other rights profiles.
role	A special identity for running privileged applications that only assigned users can assume.

SEAM Sun Enterprise Authentication Mechanism. A system for authenticating users over a network, based on the Kerberos V5 technology that was developed at the Massachusetts Institute of Technology.

"SEAM" and "Kerberos" are often used interchangeably in the SEAM documentation.

secret key See private key.

Secure Shell A special protocol for secure remote login and other secure network services over an insecure network.

security flavor See flavor.

security mechanism See mechanism.

security service See service.

server A principal that provides a resource to network clients. For example, if you `rlogin` to the machine `boston.eng.acme.com`, then that machine is the server that provides the `rlogin` service. See also service principal.

server principal (RPCSEC_GSS API) A principal that provides a service. The server principal is stored as an ASCII string in the form *service@host*. See also client principal.

service 1. A resource that is provided to network clients, often by more than one server. For example, if you `rlogin` to the machine `boston.eng.example.com`, then that machine is the server that provides the `rlogin` service.

2. A security service (either integrity or privacy) that provides a level of protection beyond authentication. See also integrity and privacy.

service key An encryption key that is shared by a service principal and the KDC, and is distributed outside the bounds of the system. See also key.

service principal A principal that provides Kerberos authentication for a service or services. For service principals, the primary name is a name of a service, such as `ftp`, and its instance is the fully qualified host name of the system that provides the service. See also host principal, user principal.

session key A key that is generated by the authentication service or the ticket-granting service. A session key is generated to provide secure transactions between a client and a service. The lifetime of a session key is limited to a single login session. See also key.

slave KDC A copy of a master KDC, which is capable of performing most functions of the master. Each realm usually has several slave KDCs (and only one master KDC). See also KDC, master KDC.

stash file A stash file contains an encrypted copy of the master key for the KDC. This master key is used when a server is rebooted to automatically authenticate the KDC before it starts the `kadmind` and `krb5kdc` processes. Because the stash file includes the master key, the stash file and any backups of it should be kept secure. If the encryption is compromised, then the key could be used to access or modify the KDC database.

ticket An information packet that is used to securely pass the identity of a user to a server or service. A ticket is valid for only a single client and a particular service on a specific server. A ticket contains the principal name of the service, the principal name of the user, the IP address of the user's host, a timestamp, and a value that defines the lifetime of the ticket. A ticket is created with a random session key to be used by the client and the service. Once a ticket has been created, it can be reused until the ticket expires. A ticket only serves to authenticate a client when it is presented along with a fresh authenticator. See also authenticator, credential, service, session key.

ticket file See credential cache.

TGS Ticket-Granting Service. That portion of the KDC that is responsible for issuing tickets.

TGT Ticket-Granting Ticket. A ticket that is issued by the KDC that enables a client to request tickets for other services.

user principal A principal that is attributed to a particular user. A user principal's primary name is a user name, and its optional instance is a name that is used to described the intended use of the corresponding credentials (for example, `joe` or `joe/admin`). Also known as a user instance. See also service principal.

VPN Virtual Private Network. A network that provides secure communication by using encryption and tunneling to connect users over a public network.

Index

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y