AIX 5L Version 5.2

Security Guide

AIX 5L Version 5.2

Security Guide

Note
Before using this information and the product it supports, read the information in Appendix E, “Notices” on page 233.

First Edition (October 2002)

This edition applies to AIX 5L Version 5.2 and to all subsequent releases of this product until otherwise indicated in new editions.

A reader’s comment form is provided at the back of this publication. If the form has been removed, address comments to Information Development, Department H6DS-905-6C006, 11501 Burnet Road, Austin, Texas 78758-3493. To send comments electronically, use this commercial Internet address: aix6kpub@austin.ibm.com. Any information that you supply may be used without incurring any obligation to you.

Contents

About This Book

This book provides system administrators with information about user and group, file, system, and network security for the AIX operating system. This guide contains information about how to perform such tasks as changing permissions, setting up authentication methods, and configuring the Trusted Computing Base environment and Controlled Access Protection Profile (CAPP) with Evaluation Assurance Level 4+ (EAL4+) features.

The *AIX 5L Version 5.2 Security Guide* contains the following parts: Standalone System Security, Network and Internet Security, and the Appendixes.

- Part 1, ″Standalone System Security,″ provides a baseline of AIX security for standalone systems. The scope of this part includes installing a standalone system with the Trusted Computing Base environment, installing the CAPP/EAL4+ features, controlling login, enforcing adequate password rules, implementing proper user-security mechanisms, enabling system auditing, and monitoring file and directory access. Also covered in this part is security information regarding the X11, Common Desktop Environment (CDE), Lightweight Directory Access Protocol (LDAP), and more.
- Part 2, ″Network and Internet Security,″ provides information about network and Internet security. This part addresses concerns about configuring TCP/IP security, controlling network services, auditing and monitoring network security, configuring IP Security, configuring Virtual Private Networks, E-mail Security, NFS security, name services, and Kerberos.
- Part 3 contains the appendixes, which include security checklists, information about security tools, online security resources, and reference information about network services and communications ports.

Who Should Use This Book

This book is intended for system administrators and IT security managers.

Highlighting

The following highlighting conventions are used in this book:

Bold	Identifies commands, subroutines, keywords, files, structures, directories, and other items whose names are predefined by the system. Also identifies graphical objects such as buttons, labels, and icons that the user selects.
Italics	Identifies parameters whose actual names or values are to be supplied by the user.
`Monospace`	Identifies examples of specific data values, examples of text similar to what you might see displayed, examples of portions of program code similar to what you might write as a programmer, messages from the system, or information you should actually type.

Case-Sensitivity in AIX

Everything in the AIX operating system is case-sensitive, which means that it distinguishes between uppercase and lowercase letters. For example, you can use the **ls** command to list files. If you type `LS`, the system responds that the command is ″not found.″ Likewise, **FILEA**, **FiLea**, and **filea** are three distinct file names, even if they reside in the same directory. To avoid causing undesirable actions to be performed, always ensure that you use the correct case.

ISO 9000

ISO 9000 registered quality systems were used in the development and manufacturing of this product.

Related Publications

The following publications contain related information:

- *AIX 5L Version 5.2 System Management Guide: Operating System and Devices*
- *AIX 5L Version 5.2 System Management Concepts: Operating System and Devices*
- *AIX 5L Version 5.2 System Management Guide: Communications and Networks*
- *AIX 5L Version 5.2 Operating System Installation: Getting Started*
- *AIX 5L Version 5.2 Installation Guide and Reference*
- *AIX 5L Version 5.2 Commands Reference*
- *AIX 5L Version 5.2 Files Reference*
- *AIX 5L Version 5.2 General Programming Concepts: Writing and Debugging Programs*
- *AIX 5L Version 5.2 System User's Guide: Operating System and Devices*
- *AIX 5L Version 5.2 System User's Guide: Communications and Networks*
- *AIX 5L Version 5.2 Network Information Services (NIS and NIS+) Guide*
- *AIX 5L Version 5.2 Guide to Printers and Printing*

Part 1. Standalone System Security

Part 1 of this guide provides information on how to protect the standalone system regardless of network connectivity. These chapters describe how to install your system with security options turned on, and how to secure AIX against nonprivileged users gaining access to the system.

Chapter 1. Installing and Configuring a Secure System

This chapter provides information about installing and configuring a secure system.

Topics in this chapter include:

The Trusted Computing Base

The system administrator must determine how much trust can be given to a particular program. This determination includes considering the value of the information resources on the system in deciding how much trust is required for a program to be installed with privilege.

The Trusted Computing Base (TCB) is the part of the system that is responsible for enforcing systemwide information security policies. By installing and using the TCB, you can define user access to the trusted communication path, which allows for secure communication between users and the TCB. TCB features can only be enabled when the operating system is installed. To install TCB on an already installed machine, you will have to perform a Preservation installation. Enabling TCB allows you to access the trusted shell, trusted processes, and the Secure Attention Key (SAK).

This section discusses the following topics:

Installing a System with the Trusted Computing Base

The TCB is the part of the system that is responsible for enforcing the information security policies of the system. All of the computer's hardware is included in the TCB, but a person administering the system should be concerned primarily with the software components of the TCB.

If you install a system with the Trusted Computing Base option, you enable the trusted path, trusted shell, and system-integrity checking (**tcbck** command). These features can *only* be enabled during a base operating system (BOS) installation. If the TCB option is not selected during the initial installation, the **tcbck** command is disabled. The command can be correctly enabled only by reinstalling the system with the TCB option turned on.

To set the TCB option during a BOS installation, select **More Options** from the Installation and Settings screen. In the Installation Options screen, the default for the **Install Trusted Computing Base** selection is **no**. To enable the TCB, type 2 and press Enter.

Because every device is part of the TCB, every file in the **/dev** directory is monitored by the TCB. In addition, the TCB automatically monitors over 600 additional files, storing critical information about these files in the **/etc/security/sysck.cfg** file. If you are installing the TCB, immediately after installing, back up this file to removable media, such as tape, CD, or disk, and store the media in a secure place.

Checking the Trusted Computing Base

The **tcbck** command audits the security state of the Trusted Computing Base. The security of the operating system is jeopardized when the TCB files are not correctly protected or when configuration files have unsafe values. The **tcbck** command audits this information by reading the **/etc/security/sysck.cfg** file. This file includes a description of all TCB files, configuration files, and trusted commands.

The **/etc/security/sysck.cfg** file is not offline and, could therefore be altered by a hacker. Make sure you create an offline read-only copy after each TCB update. Also, copy this file from the archival media to disk before doing any checks.

Installing the TCB and using the tcbck command do not guarantee that a system is operating in a Controlled Access Protection Profile (CAPP) and Evaluation Assurance Level 4+ (EAL4+) compliant mode. For information on the CAPP/EAL4+ option, see “Controlled Access Protection Profile and Evaluation Assurance Level 4+” on page 8.

Structure of the sysck.cfg file

The **tcbck** command reads the **/etc/security/sysck.cfg** file to determine which files to check. Each trusted program on the system is described by a stanza in the **/etc/security/sysck.cfg** file.

Each stanza has the following attributes:

class Name of a group of files. This attribute allows several files with the same class name to be checked by specifying a single argument to the **tcbck** command. More than one class can be specified, with each class being separated by a comma.

owner User ID or name of the file owner. If this does not match the file owner, the **tcbck** command sets the owner ID of the file to this value.

group Group ID or name of the file group. If this does not match the file owner, the **tcbck** command sets the owner ID of the file to this value.

mode Comma-separated list of values. The allowed values are SUID, SGID, SVTX, and TCB. The file permissions must be the last value and can be specified either as an octal value or as a 9-character string. For example, either **755** or **rwxr-xr-x** are valid file permissions. If this does not match the actual file mode, the **tcbck** command applies the correct value.

links Comma-separated list of path names linked to this file. If any path name in this list is not linked to the file, the **tcbck** command creates the link. If used without the *tree* parameter, the **tcbck** command prints a message that there are extra links but does not determine their names. If used with the *tree* parameter, the **tcbck** command also prints any additional path names linked to this file.

symlinks Comma-separated list of path names symbolically linked to this file. If any path name in this list is not a symbolic link to the file, the **tcbck** command creates the symbolic link. If used with the *tree* argument, the **tcbck** command also prints any additional path names that are symbolic links to this file.

program Comma-separated list of values. The first value is the path name of a checking program. Additional values are passed as arguments to the program when it is executed.

> **Note:** The first argument is always one of **-y**, **-n**, **-p**, or **-t**, depending on which flag the **tcbck** command was used with.

acl Text string representing the access control list for the file. It must be of the same format as the output of the **aclget** command. If this does not match the actual file ACL, the **sysck** command applies this value using the **aclput** command.

> **Note:** The SUID, SGID, and SVTX attributes must match those specified for the mode, if present.

source	Name of a file this source file is to be copied from prior to checking. If the value is blank, and this is either a regular file, directory, or a named pipe, a new empty version of this file is created if it does not already exist. For device files, a new special file is created for the same type device.

If a stanza in the **/etc/security/sysck.cfg** file does not specify an attribute, the corresponding check is not performed.

Using the tcbck Command

The **tcbck** command is normally used to do the following:

- Ensure the proper installation of security-relevant files
- Ensure that the file system tree contains no files that clearly violate system security
- Update, add, or delete trusted files

The **tcbck** command can be used in the following ways:

- Normal use
  - Noninteractive at system initialization
  - With the **cron** command
- Interactive use
  - Useful for checking out individual files and classes of files
- Paranoid use
  - Store the **sysck.cfg** file offline and restore it periodically to check out the machine

Although not cryptographically secure, the TCB uses the UNIX **sum** command for checksums. The TCB database can be set up manually with a different checksum command, for example, the **md5sum** command (shipped in the **textutils** RPM package with *AIX Toolbox for Linux Applications CD*).

Checking Trusted Files

To check all the files in the **tcbck** database, and fix and report all errors, type:

```
tcbck -y ALL
```

This causes the **tcbck** command to check the installation of each file in the **tcbck** database described by the **/etc/security/sysck.cfg** file.

To perform this automatically during system initialization, and produce a log of what was in error, add the previous command string to the **/etc/rc** file.

Checking the File System Tree

Whenever you suspect the integrity of the system might have been compromised, run the **tcbck** command to check the file system tree any time. This is done by running the following command:

```
tcbck -t tree
```

When the **tcbck** command is used with the *tree* value, all files on the system are checked for correct installation (this could take a long time). If the **tcbck** command discovers any files that are potential threats to system security, you can alter the suspected file to remove the offending attributes. In addition, the following checks are performed on all other files in the file system:

- If the file owner is root and the file has the **SetUID** bit set, the **SetUID** bit is cleared.
- If the file group is an administrative group, the file is executable, and the file has the **SetGID** bit set, the **SetGID** bit is cleared.

- If the file has the **tcb** attribute set, this attribute is cleared.
- If the file is a device (character or block special file), it is removed.
- If the file is an additional link to a path name described in **/etc/security/sysck.cfg** file, the link is removed.
- If the file is an additional symbolic link to a path name described in **/etc/security/sysck.cfg** file, the symbolic link is removed.

  Note: All device entries must have been added to the **/etc/security/sysck.cfg** file prior to execution of the **tcbck** command or the system is rendered unusable. To add trusted devices to the **/etc/security/sysck.cfg** file, use the **-l** flag.

Attention: *Do not* run the **tcbck -y tree** command option. This option deletes and disables devices that are not properly listed in the TCB, and might disable your system.

Adding a Trusted Program

To add a specific program to the **/etc/security/sysck.cfg** file, type:

```
tcbck -a PathName [attribute=value]
```

Only attributes whose values are not deduced from the current state of the file need be specified on the command line. All attribute names are contained in the **/etc/security/sysck.cfg** file.

For example, the following command registers a new SetUID root program named **/usr/bin/setgroups**, which has a link named **/usr/bin/getgroups**:

```
tcbck -a /usr/bin/setgroups links=/usr/bin/getgroups
```

To add `jfh` and `jsl` as administrative users and `developers` as an administrative group to be verified during a security audit of the file **/usr/bin/abc**, type:

```
tcbck -a /usr/bin/abc setuids=jfh,jsl setgids=developers
```

After installing a program, you might not know which new files are registered in the **/etc/security/sysck.cfg** file. These can be found and added with the following command:

```
tcbck -t tree
```

This command string displays the name of any file that is to be registered in the **/etc/security/sysck.cfg** file.

Deleting a Trusted Program

If you remove a file from the system that is described in the **/etc/security/sysck.cfg** file, you need to also remove the description of this file from the **/etc/security/sysck.cfg** file. For example, if you have deleted the **/etc/cvid** program, the following command string will cause an error message to be shown:

```
tcbck -t ALL
```

The error message shown is:

```
3001-020 The file /etc/cvid was not found.
```

The description for this program is still in the **/etc/security/sysck.cfg** file. To remove the description of this program type the following command:

```
tcbck -d /etc/cvid
```

Configuring Additional Trusted Options

The following sections provide information about how to configure additional options for the TCB.

Restricting Access to a Terminal

The **getty** and **shell** commands change the owner and mode of a terminal to prevent untrusted programs from accessing the terminal. The operating system provides a way to configure exclusive terminal access.

Using the Secure Attention Key

A trusted communication path is established by pressing the Secure Attention Key (SAK) reserved key sequence (Ctrl-X, and then Ctrl-R). A trusted communication path is established under the following conditions:

- When logging in to the system

  After you press the SAK:

  – If a new login screen displays, you have a secure path.

  – If the trusted shell prompt displays, the initial login screen was an unauthorized program that might have been trying to steal your password. Determine who is currently using this terminal by using the **who** command and then log off.
- When you want the command you enter to result in a trusted program running. Some examples of this include:

  – Running as root user. Run as root user only after establishing a trusted communication path. This ensures that no untrusted programs are run with root user authority.

  – Running the **su -**, **passwd**, and **newgrp** commands. Run these commands only after establishing a trusted communication path.

  Attention: Use caution when using SAK because it kills all processes that attempt to access the terminal and any links to it (for example, **/dev/console** can be linked to **/dev/tty0**).

Configuring the Secure Attention Key

Each terminal can be independently configured so that pressing SAK at that terminal creates a trusted communication path. This is specified by the **sak_enabled** attribute in **/etc/security/login.cfg** file. If the value of this attribute is True, the SAK is enabled.

If a port is to be used for communications, (for example, by the **uucp** command), the specific port used has the following line in its stanza of the **/etc/security/login.cfg** file:

```
sak_enabled = false
```

This line (or no entry in that stanza) disables the SAK for that terminal.

To enable SAK on a terminal, add the following line to the stanza for that terminal:

```
sak_enabled = true
```

Controlled Access Protection Profile and Evaluation Assurance Level 4+

In AIX 5.2, system administrators can install a system with the Controlled Access Protection Profile (CAPP) and Evaluation Assurance Level 4+ (EAL4+) option during a CD-ROM base operating system (BOS) installation. A system with this option has restrictions on the software that is installed during BOS installation, plus network access is restricted.

This section discusses the following topics:

CAPP/EAL4+ Compliant System Overview

A CAPP system is a system that has been designed and configured to meet the Controlled Access Protection Profile (CAPP) for security evaluation according to the Common Criteria. The CAPP specifies the functional requirements for the system, similar to the old TCSEC C2 standard (also known as the *Orange Book*).

A Common Criteria (CC) Evaluated System is a system that has been evaluated according to the Common Criteria, an ISO standard (ISO 15408) for the assurance evaluation of IT products. AIX 5.2 contains the technology to meet the requirements of the CAPP and CC assurance level EAL4+. The system configuration that meets these requirements is referred to as a *CAPP/EAL4+ system* in this guide.

If a system is evaluated according to the CC, the CC evaluation is valid only for a specific system configuration (hardware and software). Changing the relevant security configuration results in a nonevaluated system. This does not necessarily mean that the security of the system will be reduced, but only indicates that the system is no longer in a certified configuration. This chapter will explain the constraints of a system that must meet the CAPP and the requirements of a CC evaluation. Neither the CAPP nor the CC cover all possible security configuration options of AIX 5.2. Some features, such as IPsec or custom-password checking modules, are not included, but can be used to enhance the security of the system.

The AIX 5.2 CAPP/EAL4+ system includes the base operating system on 64-bit POWER3 and POWER4 Processors with the following:

- Logical Volume Manager (LVM) and the enhanced journaled file system (JFS2)
- The X-Windows system with the CDE GUI
- Basic Internet Protocol version 4 (IPv4) network functions (Telnet, FTP, rlogin, rsh/rcp)
- Network File System (NFS)

A CAPP/EAL4+ system is considered to be in a secured state, if the following conditions apply:

- If auditing is configured and the system is in multi-user mode, then auditing must be operational.
- The system accepts user logins and services network requests.
- For a distributed system, the administrative databases are NFS-mounted from the master server.

For the latest information on CAPP/EAL4+, refer to the *AIX 5.2 Release Notes*.

Installing a CAPP/EAL4+ System

To set the CAPP/EAL4+ option during a BOS installation, do the following:

1. In the Installation and Settings screen, select **More Options**.
2. In the More Options screen, type the number corresponding to the Yes or No choice for **Enable CAPP and EAL4+ Technology**. The default is **no**.

The **Enable CAPP and EAL4+ Technology** option is available only under the following conditions:

- The installation method is set to new and complete overwrite installation.
- The English language is selected.
- The 64-bit kernel is enabled.
- The enhanced journaled file system (JFS2) is enabled.

When the **Enable CAPP and EAL4+ Technology** option is set to **yes**, the **Trusted Computing Base** option is also set to **yes**, and the only valid **Desktop** choices are **NONE** or **CDE**.

If you are performing a nonprompted installation using a customized **bosinst.data** file, the INSTALL_TYPE field must be set to `CC_EVAL` and the following fields must be set as follows:

```
INSTALL_TYPE = CC_EVAL
INSTALL_METHOD = overwrite
TCB = yes
DESKTOP = NONE or CDE
```

A CAPP/EAL4+ system can also be installed using the Network Installation Management (NIM) environment. To install a CAPP/EAL4+ client, the NIM master must be a CAPP/EAL4+ system. Although both systems can be on the building network, CAPP/EAL4+ systems can only communicate with other CAPP/EAL4+ systems. To install a CAPP/EAL4+ client, a **bosinst_data** resource must be defined and the fields edited, as shown above.

CAPP/EAL4+ Software Bundle

When the CAPP/EAL4+ option is selected, the contents of the **/usr/sys/inst.data/sys_bundles/CC_EVAL.BOS.autoi** installation bundle are installed.

You can optionally select to install the graphics software bundle and the documentation services software bundle with the CAPP/EAL4+ option selected. If you select the Graphics Software option with the CAPP/EAL4+ option, the contents of the **/usr/sys/inst.data/sys_bundles/CC_EVAL.Graphics.bnd** software bundle are installed. If you select the Documentation Services Software option with the CAPP/EAL4+ option, the contents of the **/usr/sys/inst.data/sys_bundles/CC_EVAL.DocServices.bnd** software bundle are installed.

After the Licensed Program Products (LPPs) have been installed, the system changes the default configuration to comply with the CAPP/EAL4+ requirements. The following changes are made to the default configuration:

- Remove **/dev/echo** from the **/etc/pse.conf** file.
- Instantiate streams devices.
- Allow only root to access removable media.
- Remove non-CC entries from the **inetd.conf** file.
- Change various file permissions.
- Register symlinks in the **sysck.cfg** file.
- Register devices in the **sysck.cfg** file.
- Set default user and port attributes.
- Configure the **doc_search** application for browser use.
- Remove **httpdlite** from the **inittab** file.

- Remove **writesrv** from the **inittab** file.
- Remove **mkatmpvc** from the **inittab** file.
- Remove **atmsvcd** from the **inittab** file.
- Disable **snmpd** in the **/etc/rc.tcpip** file.
- Disable **hostmibd** in the **/etc/rc.tcpip** file.
- Disable **snmpmibd** in the **/etc/rc.tcpip** file.
- Disable **aixmibd** in the **/etc/rc.tcpip** file.
- Disable **muxatmd** in the **/etc/rc.tcpip** file.
- NFS port (2049) is a privileged port.
- Add missing events to the **/etc/security/audit/events** file.
- Ensure that the loopback interface is running.
- Create synonyms for **/dev/console**.
- Enforce default X-server connection permissions.
- Change the **/var/docsearch** directory so that all files are world-readable.
- Add ODM stanzas to set the console permissions.
- Set permissions on BSD-style ptys to 000.
- Disable **.netrc** files.
- Add patch directory processing.

Physical Environment for a CAPP/EAL4+ System

The CAPP/EAL4+ system has specific requirements for the environment in which it is run. The requirements are as follows:

- Physical access to the systems must be restricted so that only authorized administrators can use the system consoles.
- The Service Processor is not connected to a modem.
- Physical access to the terminals is restricted to authorized users.
- The physical network is secure against eavesdropping and spoofing. When communicating over insecure lines, additional security measures, such as encryption are needed.
- Communication with other systems that are not AIX 5.2 CAPP/EAL4+ systems, or are not under the same management control, is not permitted.
- Only IPv4 is to be used when communicating with other CAPP/EAL4+ systems, IPv6 has not been evaluated.
- Users should not be allowed to change the system time.

Organizational Environment for a CAPP/EAL4+ System

The following procedural and organizational requirements must be met for a CAPP/EAL4+ system:

- Only users authorized to work with the information on the systems are granted user IDs on the system.
- Users must use high-quality passwords (as random as possible and not affiliated with the user or the organization). For information about setting up password rules, see “Passwords” on page 38.
- Users must not disclose their passwords to others.
- Administrators must have sufficient knowledge to manage security critical systems.
- Administrators must work in accordance with the guidance provided by the system documentation.
- Administrators must log in with their personal ID and use **su -** to switch to superuser mode for administration.
- Passwords generated for system users by administrators must be transmitted securely to the users.
- Those who are responsible for the system must establish and implement the necessary procedures for the secure operation of the systems.

- Administrators must ensure that the access to security-critical system resources is protected by appropriate settings of permission bits and ACLs.
- The physical network must be approved by the organization to carry the most sensitive data held by the systems.
- Maintenance procedures must include regular diagnostics of the systems.
- The administrators must have procedures in place that ensure a secure operation and recovery after a system failure.
- The LIBPATH environment variable should not be changed, because this might result in a trusted process loading an untrusted library.
- Wiretapping and trace software (**tcpdump**, **trace**) must not be used on an operational system.
- Anonymous protocols such as HTTP may only be used for public information (for example, the online documentation).
- Only TCP-based NFS can be used on a CAPP/EAL4+ system.
- Access to removable media is not to be given to users. The device files are to be protected by appropriate permission bits or ACLs.
- Only root authority is used when administering AIX. All the role-based and group-based administration-delegation features, as well as the privilege mechanism of AIX, are not included in the CAPP/EAL4+ compliance.

System Configuration for a CAPP/EAL4+ System

This section provides information about the configuration of the subsystems involved in a CAPP/EAL4+ system.

Administration

Administrators must log in with their personal user account and use the **su** command to become the root user for the administration of the system. To effectively prevent guessing the root account's password, only authorized administrators should be allowed to use the **su** command on the root account. To ensure this, do the following:

1. Add an entry to the **root** stanza of the **/etc/security/user** file as follows:

```
root:
  admin = true
  .
  .
  .
  sugroups = SUADMIN
```

2. A group must be defined in the **/etc/group** file containing only the user IDs of authorized administrators as follows:

```
system:!:0:root,paul
staff:!:1:invscout,julie
bin:!:2:root,bin
.
.
.
SUADMIN:!:13:paul
```

Administrators must also adhere to the following procedures:

- Establish and implement procedures to ensure that the hardware, software and firmware components that comprise the distributed system are distributed, installed, and configured in a secure manner.
- Ensure that the system is configured so that only an administrator can introduce new trusted software into the system.
- Implement procedures to ensure that users clear the screen before logging off of serial login devices (for example, IBM 3151 terminals).

User and Port Configuration

AIX configuration options for users and ports must be set to satisfy the requirements of the evaluation. The actual requirement is that the probability of correctly guessing a password should be at least 1 in 1,000,000, and the probability of correctly guessing a password with repeated attempts in one minute should be at least 1 in 100,000.

The recommended values for the **/etc/security/user** file are the following:

```
default:
  admin = false
  login = true
  su = true
  daemon = true
  rlogin = true
  sugroups = ALL
  admgroups =
  ttys = ALL
  auth1 = SYSTEM
  auth2 = NONE
  tpath = nosak
  umask = 077
  expires = 0
  SYSTEM = "compat"
  logintimes =
  pwdwarntime = 5
  account_locked = false
  loginretries = 3
  histexpire = 52
  histsize = 20
  minage = 0
  maxage = 8
  maxexpired = 1
  minalpha = 2
  minother = 2
  minlen = 8
  mindiff = 4
  maxrepeats = 2
  dictionlist = /usr/share/dict/words
  pwdchecks =
  dce_export = false

root:
  rlogin = false
  login = false
```

The default settings in the **/etc/security/user** file should not be overwritten by specific settings for single users.

Note: Setting `login = false` in the root stanza prevents direct root login. Only user accounts that have **su** privileges for the root account will be able to log in as the root account. If a Denial of Service attack is launched against the system that sends incorrect passwords to the user accounts, it could lock all the user accounts. This attack might prevent any user (including administrative users) from logging into the system. Once a user's account is locked, the user will not be able to log in until the system administrator resets the user's `unsuccessful_login_count` attribute in the **/etc/security/lastlog** file to be less than the value of the `loginretries` user attribute. If all the administrative accounts become locked, you might need to reboot the system into maintenance mode and run the **chsec** command. For more information on using the **chsec** command, see "User Account Control" on page 27.

The recommended values for the **/etc/security/login.cfg** file are the following:

```
default:
  sak_enabled = false
  logintimes =
```

```
logindisable = 4
logininterval = 60
loginreenable = 30
logindelay = 5
```

Resource Limits

When setting resource limits in the **/etc/security/limits** file, make sure that the limits correspond to the needs of the processes on the system. In particular, the `stack` and `rss` sizes should *never* be set to `unlimited`. An unlimited stack might overwrite other segments of the running process, and an unlimited `rss` size allows a process to use all real memory, therefore creating resource problems for other processes. The `stack_hard` and `rss_hard` sizes should be limited as well.

Audit Subsystem

The following procedures help protect the audit subsystem:

- Configure the audit subsystem to record all the relevent security activities of the users. To ensure that the file space needed for auditing is available and is not impaired by other consumers of file system space, set up a dedicated file system for audit data.
- Protect audit records (such as audit trails, bin files, and all other data stored in **/audit**) from non-root users.
- For the CAPP/EAL4+ system, **bin** mode auditing must be set up when the audit subsystem is used. For information about how to set up the audit subsystem, refer to “Setting Up Auditing” on page 53.
- At least 20 percent of the available disk space in a system should be dedicated to the audit trail.
- If auditing is enabled, the **binmode** parameter in the **start** stanza in **/etc/security/audit/config** should be set to **panic**. The **freespace** parameter in the **bin** stanza should be configured at minimum to a value that equals 25 percent of the disk space dedicated to the storage of the audit trails. The **bytethreshold** and **binsize** parameters should each be set to 65536 bytes.
- Copy audit records from the system to permanent storage for archival.

Network Configuration

Network configuration must use Discretionary Access Control for Internet Ports (DACinet) to make sure that the X protocol (X11) and NFS cannot be used anonymously. For more information on the **dacinet** command, see “User Based TCP Port Access Control with Discretionary Access Control for Internet Ports” on page 126.

The **dacinet** command prevents the following conditions:

- A user from taking over another user’s desktop with X11.
- A user on a client from forging requests to an NFS server that would permit the user to become root. Normally, a user accesses a remote NFS server by making requests to the Logical File System on the local host, which then makes the request (as root) to the remote server. Setting an ACL for root only and not permitting this port to be bypassed ensures that the user cannot send direct protocol requests to an NFS server.

System Services

The following table shows the standard system services running on a CAPP/EAL4+ system (if there is no graphics card).

Table 1. Standard System Services

UID	Command	Description
root	/init	The Init process
root	/usr/sbin/syncd 60	File system sync daemon
root	/usr/sbin/srcmstr	SRC master daemon
root	/usr/sbin/cron	CRON facility with AT support
root	/usr/ccs/bin/shlap64	Shared Library Support Daemon
root	/usr/sbin/syslogd	Syslog daemon
root	/usr/lib/errdemon	AIX error log daemon
root	/usr/sbin/getty /dev/console	getty / TSM
root	/usr/sbin/portmap	Portmapper for NFS and CDE
root	/usr/sbin/biod 6	NFS Client
root	/usr/sbin/rpc.lockd	NFS lock daemon
daemon	/usr/sbin/rpc.statd	NFS stat daemon
root	/usr/sbin/rpc.mountd	NFS mount daemon
root	/usr/sbin/nfsd	NFS server daemon
root	/usr/sbin/inetd	Inetd master daemon
root	/usr/sbin/uprintfd	Kernel print daemon
root	/usr/sbin/qdaemon	Queuing daemon
root	/usr/lpp/diagnostics/bin/diagd	Diagnostics

Running a CAPP/EAL4+ Distributed System

To run a distributed system that is CAPP/EAL4+ compliant, all users must have identical user IDs on all systems. Although this can be achieved with NIS, the result is not secure enough for a CAPP/EAL4+ system. This section describes a distributed setup that ensures that the user IDs are identical on all systems that are CAPP/EAL4+ compliant.

The master system stores the identification and authentication data (user and group configuration) for the whole distributed system. All other systems use NFS to mount this data. NFS is protected by DACinet so that only the administrators can access the NFS ports on the master.

Authentication data can be changed by any administrator by using tools, such as SMIT, on any system. Authentication data is physically changed on the master.

All shared identification and authentication data comes from the **/etc/data.shared** directory. The regular identification and authentication files are replaced by symbolic links into the **/etc/data.shared** directory.

Shared Files in the Distributed System: The following files are shared in the distributed system. Typically, they come from the **/etc/security** directory.

Table 2. Shared Files in the Distributed System

File	Description
/etc/security/.ids	The next available user and group ID
/etc/security/.profile	The default **.profile** file for new users
/etc/security/audit/bincmds	Bin-mode auditing commands for this host
/etc/security/audit/config	Local audit configuration
/etc/security/audit/events	List of audit events and formats
/etc/security/audit/objects	List of audited objects on this host
/etc/security/audit/streamcmds	Stream-mode auditing commands for this host
/etc/security/environ	Per-user environmental variables
/etc/group	The **/etc/group** file
/etc/passwd	The **/etc/passwd** file
/etc/security/group	Extended group information from the **/etc/security/group** file
/etc/hosts	The **/etc/hosts** file
/etc/security/limits	Per-user resource limits
/etc/security/passwd	Per-user passwords
/etc/security/user	Per-user and default user attributes
/etc/security/priv	Ports that are to be designated as privileged when the system starts are listed in the **/etc/security/priv** file
/etc/security/services	Ports listed in the **/etc/security/services** file are considered exempt from ACL checks
/etc/security/acl	The **/etc/security/acl** file stores system-wide ACL definitions for protected services that will be reactivated at the next system boot by the **/etc/rc.tcpip** file.

Nonshared Files in the Distributed System: The following files in the **/etc/security** directory are not to be shared in the distributed system, but are to remain host-specific:

Table 3. Nonshared files in the Distributed System

File	Description
/etc/security/failedlogin	Log file for failed logins per host
/etc/security/lastlog	Per-user information about the last successful and unsuccessful logins on this host
/etc/security/portlog	Per-port information for locked ports on this host
/etc/security/login.cfg	Host-specific login characteristics for trusted path, login shells, and other login-related information

The automatically generated backup files of the shared files are also nonshared. Backup files have the same name as the original file, but have a lowercase letter `o` prepended.

Setting up the Distributed System (The Master System): On the master, a new logical volume is created that holds the file system for the identification and authentication data. The logical volume is named **/dev/hd10sec** and it is mounted on the master system as **/etc/data.master**. To generate the necessary changes on the master system, run the **mkCCadmin** command with the IP address and host name of the master, as follows:

```
mkCCadmin -m -a ipaddress hostname
```

Setting up the Distributed System (All Systems): All data that is to be shared is moved to the **/etc/data.shared** directory. At startup all systems will mount the master's **/etc/data.master** directory over the **/etc/data.shared** directory. The master itself uses a loopback mount.

Client systems are set up by running the following:

```
mkCCadmin -a ipaddress hostname
```

To change the client to use a different master, use the **chCCadmin** command.

After a system has been integrated into the distributed identification and authentication system, the following additional **inittab** entries are generated:

isCChost
Initializes the system to CAPP/EAL4+ mode.

rcCC This clears all DACinet ACLs and opens only the ports needed for the portmapper and NFS. It then mounts the shared directory.

rcdacinet
This loads additional DACinet ACLs that the administrator might have defined.

Consider the following: When running the distributed system,

- Administrators must make sure that the shared data is mounted before changing shared configuration files to ensure that the shared data is seen on all systems.
- Changing the root password is the only administrative action that is permitted while the shared directory is not mounted.

Using the DACinet Feature for User and Port Based Network Access Control

The DACinet feature can be used to restrict the access of users to TCP ports. For more information about DACinet, see “User Based TCP Port Access Control with Discretionary Access Control for Internet Ports” on page 126. For example, when using DACinet to restrict access to port TCP/25 inbound to root only with the DACinet feature, only root users from CAPP/EAL4+ compliant hosts can access this port. This situation limits the possibilities of regular users spoofing e-mail by using **telnet** to connect to port TCP/25 on the victim.

To activate the ACLs for TCP connections at boot time, the **/etc/rc.dacinet** script is run from **/etc/inittab**. It will read the definitions in the **/etc/security/acl** file and load ACLs into the kernel. Ports which should not be protected by ACLs should be listed in **/etc/security/services**. This file uses the same format as the **/etc/services** file.

Assuming a subnet of 10.1.1.0/24 for all the connected systems, the ACL entries to restrict access to the root user only for X (TCP/6000) in **/etc/security/acl** would be as follows:

```
6000    10.1.1.0/0xFFFFFF00 u:root
```

Installing Additional Software on a CAPP/EAL4+ Compliant System

The administrator can install additional software on the CAPP/EAL4+ compliant system. If the software is not run by the root user or with root user privileges, this will not invalidate the CAPP/EAL4+ compliance. Typical examples include office applications that are run only by regular users and have no SUID components.

Additionally, installed software that runs with root user privileges, invalidates the CAPP/EAL4+ compliance. This means, for example, drivers for the older JFS should not be installed, as they are running in kernel mode. Additional daemons that are run as root (for example, an SNMP daemon) also invalidates the CAPP/EAL4+ compliance.

A CAPP/EAL4+ compliant system is rarely used in the evaluated configuration, especially in a commercial environment. Typically, additional services are needed, so that the production system is based on an evaluated system, but does not comply with the exact specification of the evaluated system.

Login Control

Potential hackers can get valuable information from the default AIX login screen, such as the host name and the version of the operating system. This information would allow them to determine which exploitation methods to attempt. For security reasons you may want to change the login screen defaults as soon as possible after a system installation. This section discusses the following topics:

The KDE and GNOME desktops share some of the same security issues. For more information about KDE and GNOME, refer to the *AIX 5L Version 5.2 Installation Guide and Reference*.

For information about users, groups, and passwords, refer to the Chapter 2, "Users, Roles, and Passwords" on page 21 chapter.

Setting Up Login Controls

Set up login controls in the **/etc/security/login.cfg** file to make it harder to attack a system with password guessing as follows:

Table 4. Attributes and Recommended Values for the **/etc/security/login.cfg** *file.*

Attribute	Applies to PtYs (Network)	Applies to TTYs	Recommended Value	Comments
sak_enabled	Y	Y	false	The Secure Attention key is rarely needed. See "Using the Secure Attention Key" on page 7.
logintimes	N	Y		Specify allowed login times here.
logindisable	N	Y	4	Disable login on this terminal after 4 consecutive failed attempts.
logininterval	N	Y	60	Terminal will be disabled when the specified invalid attempts have been made within 60 seconds.
loginreenable	N	Y	30	Re-enable the terminal when it was autmatically disabled after 30 minutes.
logindelay	Y	Y	5	The time in seconds between login prompts. This will be multiplied with the number of failed attempts, for example 5,10,15,20 seconds when 5 is the initial value.

Be aware, that these port restrictions work mostly on attached serial terminals, not on pseudo-terminals used by network logins. You can specify explicit terminals in this file, for example:

```
/dev/tty0:
        logintimes = 0600-2200
        logindisable = 5
        logininterval = 80
        loginreenable = 20
```

Changing the Login Screen Welcome Message

To prevent displaying certain information on login screens, edit the *herald* parameter in the **/etc/security/login.cfg** file. The default *herald* contains the welcome message that displays with your login prompt. To change this parameter, you can either use the **chsec** command or edit the file directly.

The following example uses the **chsec** command to change the default *herald* parameter:

```
# chsec -f /etc/security/login.cfg -a default -herald
"Unauthorized use of this system is prohibited.\n\nlogin: "
```

For more information about the **chsec** command, see the *AIX 5L Version 5.2 Commands Reference, Volume 1*.

To edit the file directly, open the **/etc/security/login.cfg** file and update the *herald* parameter as follows:

```
default:
herald ="Unauthorized use of this system is prohibited\n\nlogin:"
 sak_enable = false
 logintimes =
 logindisable = 0
 logininterval = 0
 loginreenable = 0
 logindelay = 0
```

Note: Set the `logindisable` and `logindelay` variables to a number greater than `0` (`# > 0`) to make the system more secure.

Changing the Common Desktop Environment Login Screen

This security issue also affects the Common Desktop Environment (CDE) users. The CDE login screen also displays, by default, the host name and the operating system version. To prevent this information from being displayed, edit the **/usr/dt/config/$LANG/Xresources** file, where **$LANG** refers to the local language installed on your machine.

In our example, assuming that **$LANG** is set to **C**, copy this file into **/etc/dt/config/C/Xresources**. Next, open the **/usr/dt/config/C/Xresources** file and edit it to remove welcome messages that include the host name and operating system version.

For more information about CDE security issues, see “Managing X11 and CDE Concerns” on page 20.

Tightening System Default Login Parameters

Edit the **/etc/security/login.cfg** file to set up base defaults for many login parameters, such as those you might set up for a new user (number of login retries, login re-enable, and login internal).

Securing Unattended Terminals

All systems are vulnerable if terminals are left logged in and unattended. The most serious problem occurs when a system manager leaves a terminal unattended that has been enabled with root authority. In general, users should log out any time they leave their terminals. Leaving system terminals unsecure poses a potential security hazard. To lock your terminal, use the **lock** command. If your interface is AIXwindows, use the **xlock** command.

Enforcing Automatic Logoff

Another valid security concern results from users leaving their accounts unattended for a lengthy period of time. This situation allows an intruder to take control of the user’s terminal, potentially compromising the security of the system.

To prevent this type of potential security hazard, you can enable automatic logoff on the system. To do this, edit the **/etc/security/.profile** file to include an automatic logoff value for *all* users, as in the following example:

```
TMOUT=600 ; TIMEOUT=600 ; export readonly TMOUT TIMEOUT
```

The number 600, in this example, is in seconds, which is equal to 10 minutes. However, this method will only work from the shell. If the user is in a application, for example **vi**, this will not work.

While the previous action allows you to enforce an automatic logoff policy for all users, system users can bypass some restrictions by editing their individual **.profile** files. To completely implement an automatic logoff policy, take authoritative action by providing users with appropriate **.profile** files, preventing write-access rights to these files.

Managing X11 and CDE Concerns

This section discusses potential security vulnerabilities involved with the XII X server and the Common Desktop Environment (CDE).

Removing the /etc/rc.dt File

Although running the CDE graphical user interface (GUI) is convenient for users, security issues are associated with it. For this reason, do not run CDE on servers that require a high level of security. The best solution is to avoid installing CDE (dt) file sets. If you have installed these file sets on your system, consider uninstalling them, especially **/etc/rc.dt** script, which starts CDE.

For more information about CDE, see the *AIX 5L Version 5.2 System Management Guide: Operating System and Devices*.

Preventing Unauthorized Monitoring of Remote X Server

An important security issue associated with the X11 server is unauthorized silent monitoring of a remote server. The **xwd** and **xwud** commands can be used to monitor X server activity because they have the ability to capture keystrokes, which can expose passwords and other sensitive data. To solve this problem, remove these executable files unless they are necessary under your configuration, or, as an alternative, change access to these commands to be root only.

The **xwd** and **xwud** commands can be found in the **X11.apps.clients** file set.

If you do need to retain the **xwd** and **xwud** commands, consider using OpenSSH or MIT Magic Cookies. These third-party applications help prevent the risks that are created by running the **xwd** and **xwud** commands.

For more information on OpenSSH and MIT Magic Cookies, refer to each application's respective documentation.

Enabling and Disabling Access Control

The X server allows remote hosts to use the **xhost +** command to connect to your system. Ensure that you specify a host name with the **xhost +** command, because it disables access control for the X server. This allows you to grant access to specific hosts, which eases monitoring for potential attacks to the X server. To grant access to a specific host, run the **xhost** command as follows:

```
# xhost + hostname
```

For more information about the **xhost** command, see the *AIX Commands Reference, Volume 6.*

Disabling User Permissions to Run the xhost Command

Another way to ensure that the **xhost** command is being used appropriately is to restrict execution of this command to super user authority only. To do this, use the **chmod** command to change the permissions of **/usr/bin/X11/xhost** to 744.

```
chmod 744/usr/bin/X11/xhost
```

Ensure that you specify a host name with the **xhost** command because it disables access control for the X server. This allows you to grant access to specific hosts, which eases monitoring for potential attacks to the X server.

If you do not specify a host name, access will be granted to all hosts.

Chapter 2. Users, Roles, and Passwords

This chapter describes aspects of managing AIX users and roles. The following topics are discussed:

The Root Account

The **root** account has virtually unlimited access to all programs, files, and resources on a system. The **root** account is more properly known as the superuser. The superuser is the special user in the **/etc/passwd** file with the userid (UID) of `0`. This user is commonly given the username **root**. So, it is not the username that makes the **root** account so special, but the UID value of `0`. This means that any user that has a UID of `0` also has the same privileges as the superuser. Also, the **root** account is always authenticated by means of the local security files.

The **root** account should always have a password, and that password should never be shared. The **root** account should be given a password immediately after the system is installed. Only the system administrator should know the **root** password. System administrators should only operate as **root** to perform system administration functions that require **root** privileges. For all other operations, they should return to their normal user account. Routinely operating as **root** can result in damage to the system as the **root** account overrides many safeguards in the system.

Disabling Direct root Login

A common attack method of potential hackers is to obtain the super user, or root, password.

To avoid this type of attack, you can disable direct access to your root ID and then require your system administrators to obtain superuser privileges by using the **su -** command. In addition to allowing you to remove the root user as a point of attack, restricting direct root access allows you to monitor which users gained superuser access, as well as the time of their action. You can do this by viewing the **/var/adm/sulog** file. Another alternative is to enable system auditing, which will report this type of activity.

To disable remote login access for your root user, edit the **/etc/security/user** file. Specify **false** as the rlogin value on the entry for root.

Before you disable the remote root login, examine and plan for situations that would prevent a system administrator from logging in under a non-root user ID. For example, if a user's home file system is full, then the user would not be able to log in. If the remote root login were disabled and the user who could **su -** to root had a full home file system, then root could never take control of the system. This issue can be bypassed by system administrators creating home file systems for themselves that are larger than the average user's file system.

For more information on controlling root login, see "Administration" on page 11 and "User and Port Configuration" on page 12.

Administrative Roles

You can assign portions of root user authority to non-root users. Different root user tasks are assigned different authorizations. These authorizations are grouped into roles and assigned to different users.

This section covers the following topics:

- "Roles Overview"
- "Setting Up and Maintaining Roles Using SMIT"
- "Understanding Authorizations" on page 23.

Roles Overview

Roles consist of authorizations that allow a user to run functions that normally would require root user permission.

The following is a list of valid roles:

Add and Remove Users
Allows any user to act as the root user for this role. They are able to add and remove users, change information about a user, modify audit classes, manage groups, and change passwords. Anyone who performs user administration must be in the **security** group.

Change Users Password
Allows a user to change a passwords.

Manage Roles
Allows a user to create, change, remove and list roles. The user must be in the **security** group.

Backup and Restore
Allows a user to back up and restore file systems and directories. This role requires authorizations to enable a system backup and restore.

Backup Only
Allows a user only to back up file systems and directories. The user must have the proper authorization to enable a system backup.

Run Diagnostics
Allows a user or service representative to run diagnostics and diagnostic tasks. The user must have **system** specified as the primary group and also a group set that includes **shutdown**.
Note: Users in the Run Diagnostics role can change the system configuration, update microcode, and so on. Users in this role must understand the responsibility the role requires.

System Shutdown
Allows a user to shut down, reboot, or halt a system.

Setting Up and Maintaining Roles Using SMIT

SMIT fast paths (shown in the following table) are available for implementing and maintaining roles.

Table 5. Setting Up and Maintaining Roles Tasks

Task	SMIT Fast Path
Add a Role	**smit mkrole**
Change Characteristics of a Role	**smit chrole**
Show Characteristics of a Role	**smit lsrole**
Remove a Role	**smit rmrole**
List All Roles	**smit lsrole**

Understanding Authorizations

Authorizations are authority attributes for a user. These authorizations allow a user to do certain tasks. For example, a user with the UserAdmin authorization can create an administrative user by running the **mkuser** command. A user without this authority cannot create an administrative user.

Authorization has two types:

Primary Authorization
Allows a user to run a specific command. For example, RoleAdmin authorization is a primary authorization allowing a user administrator to run the **chrole** command. Without this authorization, the command terminates without modifying the role definitions.

Authorization modifier
Increases the capability of a user. For example, UserAdmin authorization is an authorization modifier that increases the capability of a user administrator belonging to the **security** group. Without this authorization, the **mkuser** command only creates non-administrative users. With this authorization, the **mkuser** command also creates administrative users.

The authorizations perform the following functions:

Backup
Performs a system backup.

The following command uses the Backup authorization:

Backup
Backs up files and file systems. The user administrator must have Backup authorization.

Diagnostics
Allows a user to run diagnostics. This authority is also required to run diagnostic tasks directly from the command line.

The following command uses the Diagnostics authorization:

diag Runs diagnostics on selected resources. If the user administrator does not have Diagnostics authority, the command ends.

GroupAdmin
Performs the functions of the root user on group data.

The following commands use the GroupAdmin authorization:

chgroup
Changes any group information. If the user does not have GroupAdmin authorization, they can only change non-administrative group information.

chgrpmem
Administers all groups. If the group administrator does not have GroupAdmin authorization, they can only change the membership of the group they administer or a user in group security to administer any non-administrative group.

chsec Modifies administrative group data in the **/etc/group** and **/etc/security/group** files. The user can also modify the default **stanza values**. If the user does not have GroupAdmin authorization, they can only modify non-administrative group data in the **/etc/group** and **/etc/security/group** files.

mkgroup
Creates any group. If the user does not have GroupAdmin authorization, the user can only create non-administrative groups.

rmgroup
Removes any group. If the user does not have GroupAdmin authorization, the user can only remove non-administrative groups.

ListAuditClasses
Views the list of valid audit classes. The user administrator who uses this authorization does not have to be the root user or in the **audit** group.

Use the **smit mkuser** or **smit chuser** fast path to list audit classes available to make or change a user. Enter the list of audit classes in the AUDIT classes field.

PasswdAdmin
Performs the functions of the root user on password data.

The following commands use the PasswdAdmin authorization:

chsec Modifies the **lastupdate** and **flags** attributes of all users. Without the PasswdAdmin authorization, the **chsec** command allows the user administrator to only modify the **lastupdate** and **flags** attribute of non-administrative users.

lssec Views the **lastupdate** and **flags** attributes of all users. Without the PasswdAdmin authorization, the **lssec** command allows the user administrator to only view the **lastupdate** and **flags** attribute of non-administrative users.

pwdadm
Changes the password of all users. The user administrator must be in group security.

PasswdManage
Performs password administration functions on non-administrative users.

The following command uses the PasswdManage authorization:

pwdadm
Changes the password of a non-administrative user. The administrator must be in group security or have the PasswdManage authorization.

UserAdmin
Performs the functions of the root user on user data. Only users with UserAdmin authorization can modify the role information of a user. You cannot access or modify user auditing information with this authorization.

The following commands use the UserAdmin authorization:

chfn Changes any user general information (gecos) field. If the user does not have UserAdmin authorization but is in group security, they can change any non-administrative user gecos field. Otherwise, users can only change their own gecos field.

chsec Modifies administrative user data in the **/etc/passwd**, **/etc/security/environ**, **/etc/security/lastlog**, **/etc/security/limits**, and **/etc/security/user** files including the roles attribute. The user administrator can also modify the default stanza values and the **/usr/lib/security/mkuser.default** file, excluding the auditclasses attributes.

chuser
Changes any user's information except for the auditclasses attribute. If the user does not have UserAdmin authorization, they can only change non-administrative user information, except for the auditclasses and roles attributes.

mkuser
Creates any user, except for the auditclasses attribute. If the user does not have UserAdmin authorization, the user can only create non-administrative users, except for the auditclasses and roles attributes.

rmuser
Removes any user. If the user administrator does not have UserAdmin authorization, they can only create non-administrative users.

UserAudit
Allows the user to modify user-auditing information.

The following commands use the UserAudit authorization:

chsec Modifies the auditclasses attribute of the **mkuser.default** file for non-administrative users. If the user has UserAdmin authorization, they can also modify the auditclasses attribute of the **mkuser.default** file for administrative and non-administrative users.

chuser
Modifies the auditclasses attribute of a non-administrative user. If the user administrator has UserAdmin authorization, they can also modify the auditclasses attribute of all users.

lsuser Views the auditclasses attribute of a non-administrative user if the user is root user or in group security. If the user has UserAdmin authorization, they can also view the auditclasses attribute of all users.

mkuser
Creates a new user and allows user administrator to assign the auditclasses attribute of a non-administrative user. If the user has UserAdmin authorization, they can also modify the auditclasses attribute of all users.

RoleAdmin
Performs the functions of the root user on role data.

The following commands use the RoleAdmin authorization:

chrole Modifies a role. If the user administrator does not have the RoleAdmin authorization, the command ends.

lsrole Views a role.

mkrole
Creates a role. If the user administrator does not have the RoleAdmin authorization, the command ends.

rmrole
Removes a role. If the user administrator does not have the RoleAdmin authorization, the command ends.

Restore
Performs a system restoration.

The following command uses the Restore authorization:

Restore
Restores backed-up files. The user administrator must have Restore authorization.

Authorization Commands List

The following table lists the commands and the authorizations they use.

Command	Permissions	Authorizations
chfn	2555 root.security	UserAdmin
chuser	4550 root.security	UserAdmin, UserAudit
diag	0550 root.system	Diagnostics
lsuser	4555 root.security	UserAudit, UserAdmin
mkuser	4550 root.security	UserAdmin, UserAudit
rmuser	4550 root.security	UserAdmin

Command	Permissions	Authorizations
chgroup	4550 root.security	GroupAdmin
lsgroup	0555 root.security	
mkgroup	4550 root.security	GroupAdmin
rmgroup	4550 root.security	GroupAdmin
chgrpmem	2555 root.security	GroupAdmin
pwdadm	4555 root.security	PasswdManage, PasswdAdmin
passwd	4555 root.security	
chsec	4550 root.security	UserAdmin, GroupAdmin, PasswdAdmin, UserAudit
lssec	0550 root.security	PasswdAdmin
chrole	4550 root.security	RoleAdmin
lsrole	0550 root.security	
mkrole	4550 root.security	RoleAdmin
rmrole	4550 root.security	RoleAdmin
backup	4555 root.system	Backup
restore	4555 root.system	Restore

User Accounts

Recommended User Attributes

User administration consists of creating users and groups and defining their attributes. A major attribute of users is how they are authenticated. Users are the primary agents on the system. Their attributes control their access rights, environment, how they are authenticated, and how, when, and where their accounts can be accessed.

Groups are collections of users who can share the same access permissions for protected resources. A group has an ID and is composed of members and administrators. The creator of the group is usually the first administrator.

Many attributes can be set for each user account, including password and login attributes. Refer to “Disk Quota System Overview” on page 44 for a list of configurable attributes. The following attributes are recommended:

- Each user should have a user ID that is not shared with any other user. All of the security safeguards and accountability tools only work if each user has a unique ID.
- Give user names that are meaningful to the users on the system. Actual names are best, since most electronic mail systems use the user ID to label incoming mail.
- Add, change, and delete users using the Web-based System Manager or SMIT interface. Although you can perform all these tasks from the command line, these interfaces help reduce small errors.

- Do not give a user account an initial password until the user is ready to log in to the system. If the password field is defined as an * (asterisk) in the **/etc/passwd** file, account information is kept, but no one can log in to that account.
- Do not change the system-defined user IDs that are needed by the system to function correctly. The system-defined user IDs are listed in the **/etc/passwd** file.
- In general, do not set the **admin** parameter to **true** for any user IDs. Only the root user can change attributes for users with **admin=true** set in the **/etc/security/user** file.

The operating system supports the standard user attributes usually found in the **/etc/passwd** and **/etc/group** files, such as:

Authentication Information	Specifies the password
Credentials	Specifies the user identifier, principal group, and the supplementary group ID
Environment	Specifies the home or shell environment.

User Account Control

Each user account has a set of associated attributes. These attributes are created from default values when a user is created using the **mkuser** command. They can be altered by using the **chuser** command. The following are the user attributes that are not used to control aspects not related to password quality:

account_locked	If an account need to be explicitly locked, this attribute can be set to *true*, the default is *false*
admin	If set to true, then this user can not change her password. Only the administrator can change it.
admgroups	Lists groups for which this user has administrative rights. For those groups the user can add or delete members.
auth1	The authentication method that is used to grant the user access. Typically it is set to SYSTEM which will then use newer methods.
auth2	Method that runs after the user has been authenticated by whatever was specified in *auth1*. It cannot block access to the system. Typically it is set to **NONE**.
daemon	This boolean parameter specifies whether the user is allowed to start daemons or subsystems with the **startsrc** command. It also restricts the use of the **cron** and **at** facilities.
login	Specifies whether this user is allowed to log in at all.
logintimes	Restricts when a user can log in. For example, a user may be restricted to accessing the system only during normal business hours.
registry	Specifies the user registry. It can be used to tell the system about alternate registries for user information, like NIS, LDAP or Kerberos.
rlogin	Specifies whether this user is allowed to log in via **rlogin** or **telnet**.
su	Specifies whether other users can switch to this id with the **su** command.
sugroups	Specifies which groups are allowed to switch to this user id
ttys	Limits certain accounts to physically secure areas
expires	Manages student or guest accounts; also can be used to turn off accounts temporarily
loginretries	Specifies the maximum number of consecutive failed login attempts bevore the userid is locked by the system. The failed attempts are recoreded in **/etc/security/lastlog**.
umask	Specifies the initial **umask** for the user

The complete set of user attributes is defined in the **/etc/security/user**, **/etc/security/limits**, **/etc/security/audit/config** and **/etc/security/lastlog** files. The default for user creation with the **mkuser** command is specified in **/usr/lib/security/mkuser.default** file. Only options that override the general defaults in the **default** stanzas of **/etc/security/user** and **/etc/securtiy/limits** as well audit classes have to be specified in the **mkuser.default** file. Several of these attributes control how a user can log in and they can be configured to lock the user account (prevent further logins) automatically under specified conditions.

Once the user account has been locked by the system, the user is not able to log in until the system administrator resets the user **unsuccessful_login_count** attribute in the **/etc/security/lastlog** file to be less than the value of login retries. This can be done using the following **chsec** command:

```
chsec -f /etc/security/lastlog -s username -a
unsuccessful_login_count=0
```

The defaults can be changed by using the **chsec** command to edit the *default* stanza in the appropriate security file, such as the **/etc/security/user** or **/etc/security/limits** files. Many of the defaults are defined to be the standard behavior. To explicitly specify attributes that are set every time a new user is created, change the *user* entry in **/usr/lib/security/mkuser.default**.

For information on extended user password attributes, refer to “Passwords” on page 38.

Login User IDs

The operating system identifies users by their login user ID. The login user ID allows the system to trace all user actions to their source. After a user logs in to the system but before running the initial user program, the system sets the login ID of the process to the user ID found in the user database. All subsequent processes during the login session are tagged with this ID. These tags provide a trail of all activities performed by the login user ID. The user can reset the effective user ID, real user ID, effective group ID, real group ID, and supplementary group ID during the session, but cannot change the login user ID.

Strengthening User Security with Access Control Lists

To achieve an appropriate level of security in your system, develop a consistent security policy to manage user accounts. The most commonly used security mechanism is the access control list (ACL). For information about ACLs and developing a security policy, see the Access Control List section in this book.

PATH Environment Variable

The **PATH** environment variable is an important security control. It specifies the directories to be searched to find a command. The default systemwide **PATH** value is specified in the **/etc/profile** file, and each user normally has a **PATH** value in the user's **$HOME/.profile** file. The **PATH** value in the **.profile** file either overrides the systemwide **PATH** value or adds extra directories to it.

Unauthorized changes to the **PATH** environment variable can enable a user on the system to ″spoof″ other users (including root users). Spoofing programs (also called Trojan horse programs) replace system commands and then capture information meant for that command, such as user passwords.

For example, suppose a user changes the **PATH** value so that the system searches the **/tmp** directory first when a command is run. Then the user places in the **/tmp** directory a program called **su** that asks for the root password just like the **su** command. Then the **/tmp/su** program mails the root password to the user and calls the real **su** command before exiting. In this scenario, any root user who used the **su** command would reveal the root password and not even be aware of it. This is just one of many scenarios for gaining confidential information by altering **PATH** values.

However, following a few simple steps will prevent any problems with the **PATH** environment variable for system administrators and users:

- When in doubt, specify full path names. If a full path name is specified, the **PATH** environment variable is ignored.
- Never put the current directory (specified by **.** (period)) in the **PATH** value specified for the root user. Never allow the current directory to be specified in **/etc/profile**.
- The root user should have its own **PATH** specification in his private **.profile**, typically the specification in **/etc/profile** lists the minimal standard for all users, whereas the root user might need more or less directories than the default.

- Warn other users not to change their **.profile** files without consulting the system administrator. Otherwise, an unsuspecting user could make changes that allow unintended access. A user **.profile** file should have permissions set to 740.
- System administrators should not use the **su** command to gain root privilege from a user session, because the user's **PATH** value specified in the **.profile** file is in effect. User's can set their **.profile** files to whatever they please. System administrators should log on to the user's machine as root or even better, with their own id and then use the following command:

  ```
  /usr/bin/su - root
  ```

  This ensures that the root environment is used during the session. If a system administrator does operate as root in another user session, then the system administrator should specify full path names throughout the session.
- Protect the input field separator (**IFS**) environment variable from being changed in the **/etc/profile** file. And beware of any user who changes the **IFS** variableb in the **.profile** file. It too can be used to alter the **PATH** value.

Set Up Anonymous FTP with a Secure User Account

This scenario sets up an anonymous **ftp** with a secure user account, using the command line interface and a script.

Note: This scenario cannot be used on a system with the Controlled Access Protection Profile (CAPP) with Evaluation Assurance Level 4+ (EAL4+) feature.

1. Verify that the **bos.net.tcp.client** fileset is installed on your system, by typing the following command:

   ```
   lslpp -L | grep bos.net.tcp.client
   ```

   If you receive no output, the fileset is not installed. For instructions on how to install it, see the *AIX 5L Version 5.2 Installation Guide and Reference*.
2. Verify that you have at least 8 MB of free space available in the system's **/home** directory, by typing the following command:

   ```
   df -k /home
   ```

   The script in step 4 requires at least 8 MB free space in the **/home** directory to install the required files and directories. If you need to increase the amount of available space, see the *AIX 5L Version 5.2 System Management Guide: Operating System and Devices*.
3. With root authority, change to the **/usr/samples/tcpip** directory. For example:

   ```
   cd /usr/samples/tcpip
   ```
4. To set up the account, run the following script:

   ```
   ./anon.ftp
   ```
5. When prompted with `Are you sure you want to modify /home/ftp?`, type **yes**. Output similar to the following displays:

   ```
   Added user anonymous.
   Made /home/ftp/bin directory.
   Made /home/ftp/etc directory.
   Made /home/ftp/pub directory.
   Made /home/ftp/lib directory.
   Made /home/ftp/dev/null entry.
   Made /home/ftp/usr/lpp/msg/en_US directory.
   ```
6. Change to the **/home/ftp** directory. For example:

   ```
   cd /home/ftp
   ```
7. Create a **home** subdirectory, by typing:

   ```
   mkdir home
   ```
8. Change the permissions of the **/home/ftp/home** directory to `drwxr-xr-x`, by typing:

```
chmod 755 home
```

9. Change to the **/home/ftp/etc** directory, by typing:
   ```
   cd /home/ftp/etc
   ```
10. Create the **objrepos** subdirectory, by typing:
    ```
    mkdir objrepos
    ```
11. Change the permissions of the **/home/ftp/etc/objrepos** directory to `drwxrwxr-x`, by typing:
    ```
    chmod 775 objrepos
    ```
12. Change the owner and group of the **/home/ftp/etc/objrepos** directory to the root user and the system group, by typing:
    ```
    chown root:system objrepos
    ```
13. Create a **security** subdirectory, by typing:
    ```
    mkdir security
    ```
14. Change the permissions of the **/home/ftp/etc/security** directory to `drwxr-x---`, by typing:
    ```
    chmod 750 security
    ```
15. Change the owner and group of the **/home/ftp/etc/security** directory to the root user and the security group, by typing:
    ```
    chown root:security security
    ```
16. Change to the **/home/ftp/etc/security** directory, by typing:
    ```
    cd security
    ```
17. Add a user by typing the following SMIT fast path:
    ```
    smit mkuser
    ```

    In this scenario, we are adding a user named `test`.
18. In the SMIT fields, enter the following values:
    ```
    User NAME                                       [test]
    ADMINISTRATIVE USER?                             true
    Primary GROUP                                   [staff]
    Group SET                                       [staff]
    Another user can SU TO USER?                     true
    HOME directory                                  [/home/test]
    ```

    After you enter your changes, press Enter to create the user. After the SMIT process completes, exit SMIT.
19. Create a password for this user with the following command:
    ```
    passwd test
    ```

    When prompted, enter the desired password. You must enter the new password a second time for confirmation.
20. Change to the **/home/ftp/etc** directory, by typing:
    ```
    cd /home/ftp/etc
    ```
21. Copy the **/etc/passwd** file to the **/home/ftp/etc/passwd** file, using the following command:
    ```
    cp /etc/passwd /home/ftp/etc/passwd
    ```
22. Using your favorite editor, edit the **/home/ftp/etc/passwd** file. For example:
    ```
    vi passwd
    ```
23. Remove all lines from the copied content except those for the root, ftp, and test users. After your edit, the content should look similar to the following:
    ```
    root:!:0:0::/:/bin/ksh
    ftp:*:226:1::/home/ftp:/usr/bin/ksh
    test:!:228:1::/home/test:/usr/bin/ksh
    ```
24. Save your changes and exit the editor.

25. Change the permissions of the **/home/ftp/etc/passwd** file to `-rw-r--r--`, by typing:
    ```
    chmod 644 passwd
    ```
26. Change the owner and group of the **/home/ftp/etc/passwd** file to the root user and the security group, by typing:
    ```
    chown root:security passwd
    ```
27. Copy the contents of the **/etc/security/passwd** file to the **/home/ftp/etc/security/passwd** file, using the following command:
    ```
    cp /etc/security/passwd /home/ftp/etc/security/passwd
    ```
28. Using your favorite editor, edit the **/home/ftp/etc/security/passwd** file. For example:
    ```
    vi ./security/passwd
    ```
29. Remove all stanzas from the copied content except the stanza for the test user.
30. Remove the `flags = ADMCHG` line from the test user stanza. After your edits, the content should look similar to the following:
    ```
    test:
            password = 2HaAYgpDZX3Tw
            lastupdate = 990633278
    ```
31. Save your changes and exit the editor.
32. Change the permissions of the **/home/ftp/etc/security/passwd** file to `-rw-------`, by typing:
    ```
    chmod 600 ./security/passwd
    ```
33. Change the owner and group of the **/home/ftp/etc/security/passwd** file to the root user and the security group, by typing:
    ```
    chown root:security ./security/passwd
    ```
34. Using your favorite editor, edit the **/home/ftp/etc/security/group** file. For example:
    ```
    vi ./security/group
    ```
35. Add the following lines to the file:
    ```
    system:*:0:
    staff:*:1:test
    ```
36. Save your changes and exit the editor.
37. Use the following commands to copy the appropriate content into the **/home/ftp/etc/objrepos** directory:
    ```
    cp /etc/objrepos/CuAt ./objrepos
    cp /etc/objrepos/CuAt.vc ./objrepos
    cp /etc/objrepos/CuDep ./objrepos
    cp /etc/objrepos/CuDv ./objrepos
    cp /etc/objrepos/CuDvDr ./objrepos
    cp /etc/objrepos/CuVPD ./objrepos
    cp /etc/objrepos/Pd* ./objrepos
    ```
38. Change to the **/home/ftp/home** directory, by typing:
    ```
    cd ../home
    ```
39. Make a new home directory for your user, by typing:
    ```
    mkdir test
    ```

    This will be the home directory for the new ftp user.
40. Change the owner and group of the **/home/ftp/home/test** directory to the `test` user and the staff group, by typing:
    ```
    chown test:staff test
    ```
41. Change the permissions of the **/home/ftp/home/test** file to `-rwx------`, by typing:
    ```
    chmod 700 test
    ```

At this point, you have ftp sublogin set up on your machine. You can test this with the following procedure:

1. Using ftp, connect to the host on which you created the `test` user. For example:

```
ftp MyHost
```

2. Log in as `anonymous`. When prompted for a password, press Enter.
3. Switch to the newly created `test` user, by using the following command:

   ```
   user test
   ```

   When prompted for a password, use the password you created in step 19 on page 30
4. Use the pwd command to verify the user's home directory exists. For example:

   ```
   ftp> pwd
        /home/test
   ```

   The output shows **/home/test** as an **ftp** subdirectory. The full path name on the host is actually **/home/ftp/home/test**.

System Special User Accounts

AIX provides a default set of system special user accounts that prevents root and system from owning all operating system files and file systems.

Attention: Use caution when removing a system special user account. You can disable a specific account by inserting an asterisk (*) at the beginning of its corresponding line of the **/etc/security/passwd** file. However, be careful not to disable the **root** user account. If you remove system special user accounts or disable the **root** account, the operating system will not function.

The following accounts are predefined in the operating system:

root The root user account, UID 0, sometimes called the superuser account, through which you can perform system maintenance tasks and troubleshoot system problems.

daemon
The daemon user account exists only to own and execute system server processes and their associated files.) This account guarantees that such processes execute with the appropriate file access permissions.

bin The bin user account typically owns the executable files for most user commands. This account's primary purpose is to help distribute the ownership of important system directories and files so everything is not owned solely by the root and sys user accounts.

sys The sys user owns the default mounting point for the Distributed File Service (DFS) cache, which must exist before you can install or configure DFS on a client. The **/usr/sys** directory can also store install images.

adm The adm user account owns two basic system functions:

1. Diagnostics, the tools for which are stored in the **/usr/sbin/perf/diag_tool** directory.
2. Accounting, the tools for which are stored in the following directories:
   - **/usr/sbin/acct**
   - **/usr/lib/acct**
   - **/var/adm**
   - **/var/adm/acct/fiscal**
   - **/var/adm/acct/nite**
   - **/var/adm/acct/sum**

nobody
The nobody user account is used by the Network File System (NFS) product to enable remote printing. This account exists so a program can permit temporary root access to root users. For example, before turning on Secure RPC or Secure NFS, check the **/etc/public** key on the master

NIS server to find a user has not been assigned a public key and a secret key. As root user, you can create an entry in the database for each unassigned user by entering:

```
newkey -u username
```

Or, you can create an entry in the database for the nobody user account, and then any user can run the **chkey** program to create their own entries in the database without logging in as root.

Removing Unnecessary Default User Accounts

During installation of the operating system, a number of default user and group IDs are created. Depending on the applications you are running on your system and where your system is located in the network, some of these user and group IDs can become security weaknesses, vulnerable to exploitation. If these users and group IDs are not needed, you can remove them to minimize security risks associated with them.

The following table lists the most common default user IDs that you might be able to remove:

Table 6. Common default user IDs that you might be able to remove.

User ID	Description
uucp, nuucp	Owner of hidden files used by uucp protocol. The uucp user account is used for the UNIX-to-UNIX Copy Program, which is group of commands, programs, and files, present on most UNIX systems, that allows the user to communicate with another UNIX system over a dedicated line or a telephone line.
lpd	Owner of files used by printing subsystem
imnadm	IMN search engine (used by Documentation Library Search)
guest	Allows access to users who do not have access to accounts

The following table lists common group IDs that might not be needed:

Table 7. Common group IDs that might not be needed.

Group ID	Description
uucp	Group to which uucpand nuucp users belong
printq	Group to which lpd user belongs
imnadm	Group to which imnadm user belongs

Analyze your system to determine which IDs are indeed not needed. There might also be additional user and group IDs that you might not need. Before your system goes into production, perform a thorough evaluation of available IDs.

Access Control Lists

Access control consists of protected information resources that specify who can be granted access to such resources. The operating system allows for need-to-know or discretionary security. The owner of an information resource can grant other users read or write access rights for that resource. A user who is given access to an object may create additional copies of the object and give a third party access rights to the newly created object. However, only the object owner can grant a third party access rights to the original object. The owner of an object and the root user are the only users that may change the access rights to an object.

Users have user-based access only to the objects that they own. Typically, users receive either the group permissions or the default permissions for a resource. The major task in administering access control is to define the group memberships of users, because these memberships determine the users' access rights to the files that they do not own.

Access control lists (ACLs) increase the quality of file access controls by adding extended permissions that modify the base permissions assigned to individuals and groups. With extended permissions, you can permit or deny file access to specific individuals or groups without changing the base permissions.

Access control also involves managing protected resources using the **setuid** and **setgid** programs and hard-copy labeling. The operating system supports several types of information resources, or objects. These objects allow user processes to store or communicate information.

The most important types of objects are:

- Files and directories (used for information storage)
- Named pipes, message queues, shared memory segments, and semaphores (used for information transfer between processes)

Each object has an associated owner, group, and mode. The mode defines access permissions for the owner, group, and other users.

The following are the direct access control attributes for the different types of objects:

Owner The owner of a specific object controls its discretionary access attributes. The owner's attributes are set to the creating process's effective user ID. For file system objects, the direct access control attributes for an owner cannot be changed without root privilege.

For System V Interprocess Communication (SVIPC) objects, either the creator or owner can change the owner. SVIPC objects have an associated creator that has all the rights of the owner (including access authorization). However, the creator cannot be changed, even with root privilege.

Group SVIPC objects are initialized to the effective group ID of the creating process. For file system objects, the direct access control attributes are initialized to either the effective group ID of the creating process or the group ID of the parent directory (this is determined by the group inheritance flag of the parent directory).

The owner of an object can change the group; the new group must be either the effective group ID of the creating process or the group ID of the parent directory. The owner of an object can change the group; the new group must be either the effective group or in the supplementary group ID of the owner's current process. (As above, SVIPC objects have an associated creating group that cannot be changed and share the access authorization of the object group.)

Note: The access control list for a file cannot exceed one memory page (approximately 4096 bytes) in size.

To maintain access control lists, use the **aclget**, **acledit**, and the **aclput** commands.

The **chmod** command in numeric mode (with octal notations) can set base permissions and attributes. The **chmod** subroutine, which the command calls, disables extended permissions. If you use the numeric mode of the **chmod** command on a file that has an ACL, extended permissions are disabled. The symbolic mode of the **chmod** command does not disable extended permissions. For information on numeric and symbolic mode, refer to the **chmod** command.

Using setuid and setgid Programs

The permission bits mechanism allows effective access control for resources in most situations. But for more precise access control, the operating system provides **setuid** and **setgid** programs.

Most programs execute with the user and group access rights of the user who invoked them. Program owners can associate the access rights of the user who invoked them by making the program a **setuid** or **setgid** program; that is, a program with the setuid or setgid bit set in its permissions field. When that program is executed by a process, the process acquires the access rights of the owner of the program. A **setuid** program executes with the access rights of its owner, while a **setgid** program has the access rights of its group and both bits can be set according to the permission mechanism.

Although the process is assigned the additional access rights, these rights are controlled by the program bearing the rights. Thus, the **setuid** and **setgid** programs allow for user-programmed access controls in which access rights are granted indirectly. The program acts as a trusted subsystem, guarding the user's access rights.

Although these programs can be used with great effectiveness, there is a security risk if they are not designed carefully. In particular, the program must never return control to the user while it still has the access rights of its owner, because this would allow a user to make unrestricted use of the owner's rights.

> **Note:** For security reasons, the operating system does not support **setuid** or **setgid** calls within a shell script.

Administrative Access Rights

The operating system provides privileged access rights for system administration. System privilege is based on user and group IDs. Users with effective user or group IDs of 0 are recognized as privileged.

Processes with effective user IDs of 0 are known as root user processes and can:

- Read or write any object
- Call any system function
- Perform certain subsystem control operations by executing **setuid-root** programs.

You can manage the system using two types of privilege: the **su** command privilege and **setuid-root** program privilege. The **su** command allows all programs you invoke to function as root user processes, and **su** is a flexible way to manage the system, but it is not very secure.

Making a program into a **setuid-root** program means the program is a root user-owned program with the setuid bit set. A **setuid-root** program provides administrative functions that ordinary users can perform without compromising security; the privilege is encapsulated in the program rather than granted directly to the user.

It can be difficult to encapsulate all necessary administrative functions in **setuid-root** programs, but it provides more security to system managers.

Base Permissions

Base permissions are the traditional file-access modes assigned to the file owner, file group, and other users. The access modes are: read (r), write (w), and execute/search (x).

In an access control list, base permissions are in the following format, with the *Mode* parameter expressed as rwx (with a hyphen (-) replacing each unspecified permission):

```
base permissions:
   owner(name): Mode
   group(group): Mode
   others: Mode
```

Attributes

Three attributes can be added to an access control list:

setuid (SUID)
Set-user-ID mode bit. This attribute sets the effective and saved user IDs of the process to the owner ID of the file on execution.

setgid (SGID)
Set-group-ID mode bit. This attribute sets the effective and saved group IDs of the process to the group ID of the file on execution.

savetext (SVTX)
For directories, indicates that only file owners can link or unlink files in the specified directory.

These attributes are added in the following format:

```
attributes: SUID, SGID, SVTX
```

Extended Permissions

Extended permissions allow the owner of a file to define access to that file more precisely. Extended permissions modify the base file permissions (owner, group, others) by permitting, denying, or specifying access modes for specific individuals, groups, or user and group combinations. Permissions are modified through the use of keywords.

The **permit**, **deny**, and **specify** keywords are defined as follows:

permit	Grants the user or group the specified access to the file
deny	Restricts the user or group from using the specified access to the file
specify	Precisely defines the file access for the user or group

If a user is denied a particular access by either a **deny** or a **specify** keyword, no other entry can override that access denial.

The **enabled** keyword must be specified in the ACL for the extended permissions to take effect. The default value is the **disabled** keyword.

In an ACL, extended permissions are in the following format:

```
extended permissions:
  enabled | disabled
    permit   Mode  UserInfo...:
    deny     Mode  UserInfo...:
    specify  Mode  UserInfo...:
```

Use a separate line for each **permit**, **deny**, or **specify** entry. The *Mode* parameter is expressed as **rwx** (with a hyphen (-) replacing each unspecified permission). The *UserInfo* parameter is expressed as `u:UserName`, or `g:GroupName`, or a comma-separated combination of `u:UserName` and `g:GroupName`.

> **Note:** If more than one user name is specified in an entry, that entry cannot be used in an access control decision, because a process has only one user ID.

Access Control List Example

The following is an example of an ACL:

```
attributes: SUID
base permissions:
      owner(frank):  rw-
      group(system): r-x
      others: ---
extended permissions:
      enabled
```

```
permit  rw-  u:dhs
deny    r--  u:chas, g:system
specify r--  u:john, g:gateway, g:mail
permit  rw-  g:account, g:finance
```

The parts of the ACL and their meanings are the following:

- The first line indicates that the **setuid** bit is turned on.
- The next line, which introduces the base permissions, is optional.
- The next three lines specify the base permissions. The owner and group names in parentheses are for information only. Changing these names does not alter the file owner or file group. Only the **chown** command and the **chgrp** command can change these file attributes.
- The next line, which introduces the extended permissions, is optional.
- The next line indicates that the extended permissions that follow are enabled.
- The last four lines are the extended entries. The first extended entry grants user `dhs` read (r) and write (w) permission on the file.
- The second extended entry denies read (r) access to user `chas` only when he is a member of the `system` group.
- The third extended entry specifies that as long as user `john` is a member of both the `gateway` group and the `mail` group, has read (r) access. If user `john` is not a member of both groups, this extended permission does not apply.
- The last extended entry grants any user in *both* the `account` group and the `finance` group read (r) and write (w) permission.

> **Note:** More than one extended entry can apply to a process that is requesting access to a controlled object, with restrictive entries taking precedence over permissive modes.

See the **acledit** command in the *AIX 5L Version 5.2 Commands Reference* for the complete syntax.

Access Authorization

The owner of the information resource is responsible for managing access rights. Resources are protected by permission bits, which are included in the mode of the object. The permission bits define the access permissions granted to the owner of the object, the group of the object, and for the `others` default class. The operating system supports three different modes of access (read, write, and execute) that can be granted separately.

When a user logs in to an account (using the **login** or **su** commands), the user IDs and group IDs assigned to that account are associated with the user's processes. These IDs determine the access rights of the process.

For files, directories, named pipes, and devices (special files), access is authorized as follows:

- For each access control entry (ACE) in the access control list (ACL), the identifier list is compared to the identifiers of the process. If there is a match, the process receives the permissions and restrictions defined for that entry. The logical unions for both permissions and restrictions are computed for each matching entry in the ACL. If the requesting process does not match any of the entries in the ACL, it receives the permissions and restrictions of the default entry.
- If the requested access mode is permitted (included in the union of the permissions) and is not restricted (included in the union of the restrictions), access is granted. Otherwise, access is denied.

A process with a user ID of 0 is known as a *root user process*. These processes are generally allowed all access permissions. But if a root user process requests execute permission for a program, access is granted only if execute permission is granted to at least one user.

The identifier list of an ACL matches a process if all identifiers in the list match the corresponding type of effective identifier for the requesting process. A USER-type identifier matched is equal to the effective user ID of the process, and a GROUP-type identifier matches if it is equal to the effective group ID of the process or to one of the supplementary group IDs. For instance, an ACE with an identifier list such as the following:

```
USER:fred, GROUP:philosophers, GROUP:software_programmer
```

would match a process with an effective user ID of `fred` and a group set of:

```
philosophers, philanthropists, software_programmer, doc_design
```

but would not match for a process with an effective user ID of `fred` and a group set of:

```
philosophers, iconoclasts, hardware_developer, graphic_design
```

Note that an ACE with an identifier list of the following would match for both processes:

```
USER:fred, GROUP:philosophers
```

In other words, the identifier list in the ACE functions is a set of conditions that must hold for the specified access to be granted.

All access permission checks for these objects are made at the system call level when the object is first accessed. Because System V Interprocess Communication (SVIPC) objects are accessed statelessly, checks are made for every access. For objects with file system names, it is necessary to be able to resolve the name of the actual object. Names are resolved either relatively (to the process' working directory) or absolutely (to the process' root directory). All name resolution begins by searching one of these.

The discretionary access control mechanism allows for effective access control of information resources and provides for separate protection of the confidentiality and integrity of the information. Owner-controlled access control mechanisms are only as effective as users make them. All users must understand how access permissions are granted and denied, and how these are set.

Passwords

Guessing passwords is one of the most common attack methods that a system experiences. Therefore, controlling and monitoring your password-restriction policy is essential. AIX provides mechanisms to help you enforce a stronger password policy, such as establishing values for the following:

- Minimum and maximum number of weeks that can elapse before and after a password can be changed
- Minimum length of a password
- Minimum number of alphabetic characters that can be used when selecting a password

This section discusses how AIX stores and handles passwords, and how you can establish a strong password policy. Topics in this section include:

What is a Good Password?

Good passwords are effective first lines of defense against unauthorized entry into a system if they are the following:

- A mixture of both uppercase and lowercase letters
- A combination of alphabetic, numeric, or punctuation characters. Also, they may have special characters like ~!@#$%^&*()-_=+[]{}|\;:'",.<>?/<space>
- Are not written down anywhere
- Are at least seven to a maximum of eight characters in length, if using the **/etc/security/passwd** file (authentication implementations that use registries, such as LDAP, can have passwords that exceed this maximum length)
- Are not real words that can be found in any dictionary
- Are not patterns of letters on the keyboard, like *qwerty*
- Are not real words or known patterns spelled backwards
- Do not contain any personal information about yourself, family, or friends
- Do not follow the same pattern as a previous password
- Can be typed relatively quickly so someone nearby cannot determine your password

In addition to these mechanisms, you can further enforce stricter rules by restricting passwords so that they cannot include standard UNIX words, which can be guessed. This feature uses the **dictionlist**, which requires that you first have the **bos.data** and **bos.txt** file sets installed.

To implement the previously defined **dictionlist**, edit the following line in the **/etc/security/users** file:

```
dictionlist = /usr/share/dict/words
```

The **/usr/share/dict/words** file uses the **dictionlist** to prevent standard UNIX words from being used as passwords.

The /etc/passwd File

Traditionally, the **/etc/passwd** file is used to keep track of every registered user that has access to a system. The **/etc/passwd** file is a colon-separated file that contains the following information:

- user name
- encrypted password
- user ID number (UID)
- user's group ID number (GID)
- full name of the user (GECOS)
- user home directory
- login shell

Here is an example of a **/etc/passwd** file:

```
root:!:0:0::/:/usr/bin/ksh
daemon:!:1:1::/etc:
bin:!:2:2::/bin:
sys:!:3:3::/usr/sys:
adm:!:4:4::/var/adm:
uucp:!:5:5::/usr/lib/uucp:
guest:!:100:100::/home/guest:
nobody:!:4294967294:4294967294::/:
lpd:!:9:4294967294::/:
lp:*:11:11::/var/spool/lp:/bin/false
invscout:*:200:1::/var/adm/invscout:/usr/bin/ksh
nuucp:*:6:5:uucp login user:/var/spool/uucppublic:/usr/sbin/uucp/uucico
imnadm:*:188:188::/home/imnadm:/usr/bin/ksh
paul:!:201:1::/home/paul:/usr/bin/ksh
jdoe:*:202:1:John Doe:/home/jdoe:/usr/bin/ksh
```

AIX does not store encrypted passwords in the **/etc/password** file the way UNIX systems do, but in the **/etc/security/password** file by default, which is only readable by the superuser. The password filed in **/etc/passwd** is used by AIX to signify if there is a password or whether the account is blocked.

The **/etc/passwd** file is owned by the root user and must be readable by all the users, but only the root user has writable permissions, which is shown as `-rw-r--r--`. If a userid has a password, then the password field will have an `!` (exclamation point). If the userid does not have a password, then the password field will have an `*` (asterisk). The encrypted passwords are stored in the **/etc/security/passwd** file. The example below contains the last four entries in the **/etc/security/passwd** file based on the entries from the **/etc/passwd** file shown above.

```
guest:
        password = *

nobody:
        password = *

lpd:
        password = *

paul:
        password = eacVScDKri4s6
        lastupdate = 1026394230
        flags = ADMCHG
```

Notice that the userid `jdoe` does not have an entry in the **/etc/security/passwd** file because it does not have a password set in the **/etc/passwd** file.

The consistency of the **/etc/passwd** file can be checked using the **pwdck** command. The **pwdck** command verifies the correctness of the password information in the user database files by checking the definitions for all of the users or for specified users.

The /etc/passwd File and Network Environments

Traditionally, in a networked environment a user must have had an account on each system to gain access to that system. That typically meant that the user would have an entry in each of the **/etc/passwd** files on each system. However, in a distributed environment there is no easy way to ensure that every system had the same **/etc/passwd** file. To solve this problem, several methods have been developed to make the information in the **/etc/passwd** file available over the network, including the following:

- Network Information System (NIS)
- NIS+

Both of these topics are discussed in the NIS chapter.

Hiding User Names and Passwords

To achieve a higher level of security, ensure that user IDs and passwords are not visible within the system. The **.netrc** files contain user IDs and passwords. This file is not protected by encryption or encoding, thus its contents are clearly shown as plain text. To find these files, run the following command:

```
# find `awk -F: '{print $6}' /etc/passwd` -name .netrc -ls
```

After you locate these files, delete them. A more effective way to save passwords is by setting up Kerberos.

Setting Recommended Password Options

Proper password management can only be accomplished through user education. But to provide some additional security, the operating system provides configurable password restrictions. These allow the administrator to constrain the passwords chosen by users and to force passwords to be changed regularly. Password options and extended user attributes are located in the **/etc/security/user** file. This is an ASCII

file that contains attribute stanzas for users. These restrictions are enforced whenever a new password is defined for a user. All password restrictions are defined per user. By keeping restrictions in the default stanza of the **/etc/security/user** file, the same restrictions are enforced on all users. To maintain password security, all passwords must be similarly protected.

The operating system also provides a method for administrators to extend the password restrictions. Using the **pwdchecks** attribute of the **/etc/security/user** file, an administrator can add new subroutines (known as methods) to the password restrictions code. Thus, local site policies can be added to and enforced by the operating system. See “Extending Password Restrictions” on page 43 for more information.

Apply password restrictions sensibly. Attempts to be too restrictive, such as limiting the password space, which makes guessing the password easier, or forcing the user to select passwords that are difficult to remember, which might then be written down, can jeopardize password security. Ultimately, password security rests with the user. Simple password restrictions, coupled with sensible guidelines and an occasional audit to see if current passwords are unique, are the best policy.

The following table lists recommended values for some security attributes related to user passwords in the **/etc/security/user** file.

Table 8. Recommended security attribute values for user passwords.

Attribute	Description	Recommended Value	Default Value	Maximum Value
dictionlist	Verifies passwords do not include standard UNIX words.	**/usr/share/dict/words**	NA[Note 1]	NA
histexpire	Number of weeks before password can be reused.	26	0	260[Note 2]
histsize	Number of password iterations allowed.	20	0	50
maxage	Maximum number of weeks before password must be changed.	8	0	52
maxexpired	Maximum number of weeks beyond *maxage* that an expired password can be changed by the user. (Root is exempt.)	2	-1	52
maxrepeats	Maximum number of characters that can be repeated in passwords.	2	8	8

Table 8. Recommended security attribute values for user passwords. (continued)

Attribute	Description	Recommended Value	Default Value	Maximum Value
minage	Minimum number of weeks before a password can be changed. This should not be set to a non zero value unless administrators are always easy to reach to reset an accidentally compromised password that was recently changed.	0	0	52
minalpha	Minimum number of alphabetic characters required on passwords.	2	0	8
mindiff	Minimum number of unique characters that passwords must contain.	4	0	8
minlen	Minimum length of password.	6 (8 for root user)	0	8
minother	Minimum number of non-alphabetic characters required on passwords.	2	0	8
pwdwarntime	Number of days before the system issues a warning that a password change is required.	5	NA	NA
pwdchecks	This entry can be used to augment the **passwd** command with a custom code that checks the password quality.	For more information. see “Extending Password Restrictions” on page 43.	NA	NA

Notes:

1. NA means *Not Applicable*.
2. A maximum of 50 passwords are retained.

For a Controlled Access Protection Profile and Evaluation Assurance Level 4+ (CAPP/EAL4+) system, use the values recommended in “User and Port Configuration” on page 12.

If text processing is installed on the system, the administrator can use the **/usr/share/dict/words** file as a **dictionlist** dictionary file. In such a case, the administrator can set the **minother** attribute to `0`. Because most words in the dictionary file do not contain characters that fall into the **minother** attribute category, setting the **minother** attribute to `1` or more eliminates the need for the vast majority of words in this dictionary file.

The minimum length of a password on the system is set by the value of the **minlen** attribute or the value of the **minalpha** attribute plus the value of the **minother** attribute, whichever is greater. The maximum length of a password is eight characters. The value of the **minalpha** attribute plus the value of the **minother** attribute must never be greater than eight. If the value of the **minalpha** plus the value of the **minother** attribute is greater than eight, then the value of the **minother** attribute is reduced to eight minus the value of the **minalpha** attribute.

If the values of both the **histexpire** attribute and the **histsize** attribute are set, the system retains the number of passwords required to satisfy both conditions up to the system limit of 50 passwords per user. Null passwords are not retained.

You can edit the **/etc/security/user** file to include any defaults you want to use to administer user passwords. Alternatively, you can change attribute values using the **chuser** command.

Other commands that can be used with this file are the **mkuser**, **lsuser**, and **rmuser**. The **mkuser** command creates an entry for each new user in the **/etc/security/user** file and initializes its attributes with the attributes defined in the **/usr/lib/security/mkuser.default** file. To display the attributes and their values, use the **lsuser** command. To remove a user, use the **rmuser** command.

Extending Password Restrictions

The rules used by the password program to accept or reject passwords (the password composition restrictions) can be extended by system administrators to provide site-specific restrictions. Restrictions are extended by adding subroutines, known as *methods*, which are called during a password change. The **pwdchecks** attribute in the **/etc/security/user** file specifies the methods called.

The *AIX 5L Version 5.2 Technical Reference* contains a description of the **pwdrestrict_method**, the subroutine interface to which specified password restriction methods must conform. To correctly extend the password composition restrictions, the system administrator must program this interface when writing a password restriction method. Use caution in extending the password composition restrictions. These extensions directly affect the **login** command, the **passwd** command, the **su** command, and other programs. The security of the system could easily be subverted by malicious or defective code. Only use code that you trust.

User Authentication

Identification and authentication establish a user's identity. Each user is required to log in to the system. The user supplies the user name of an account and a password, if the account has one (in a secure system, all accounts must either have passwords or be invalidated). If the password is correct, the user is logged in to that account; the user acquires the access rights and privileges of the account. The **/etc/passwd** and **/etc/security/passwd** files maintain user passwords.

Alternative methods of authentication are integrated into the system by means of the **SYSTEM** attribute that appears in **/etc/security/user**. For instance, the Distributed Computing Environment (DCE) requires password authentication but validates these passwords in a manner different from the encryption model used in /**etc/passwd** and **/etc/security/passwd**. Users who authenticate by means of DCE can have their stanza in **/etc/security/user** set to `SYSTEM=DCE`**.**

Other **SYSTEM** attribute values are **compat**, **files**, and **NONE**. The **compat** token is used when name resolution (and subsequent authentication) follows the local database, and if no resolution is found, the Network Information Services (NIS) database is tried. The **files** token specifies that only local files are to be used during authentication. Finally, the **NONE** token turns off method authentication. To turn off all authentication, the **NONE** token must appear in the **SYSTEM** and **auth1** lines of the user's stanza.

Other acceptable tokens for the **SYSTEM** attribute can be defined in **/usr/lib/security/methods.cfg**.

Note: The root user is always authenticated by means of the local system security file. The **SYSTEM** attribute entry for the root user is specifically set to **SYSTEM = "compat"** in **/etc/security/user**.

See the *AIX 5L Version 5.2 System User's Guide: Operating System and Devices* for more information on protecting passwords.

Login User IDs

All audit events recorded for this user are labeled with this ID and can be examined when you generate audit records. See the *AIX 5L Version 5.2 System User's Guide: Operating System and Devices* for more information about login user IDs.

Disk Quota System Overview

The disk quota system allows system administrators to control the number of files and data blocks that can be allocated to users or groups. The following sections provide further information about the disk quota system, its implementation, and use:

- "Understanding the Disk Quota System"
- "Recovering from Over-Quota Conditions"
- "Setting Up the Disk Quota System" on page 45

Understanding the Disk Quota System

The disk quota system, based on the Berkeley Disk Quota System, provides an effective way to control the use of disk space. The quota system can be defined for individual users or groups, and is maintained for each journaled file system.

The disk quota system establishes limits based on the following parameters that can be changed with the **edquota** command:

- User's or group's soft limits
- User's or group's hard limits
- Quota grace period

The *soft limit* defines the number of 1 KB disk blocks or files under which the user must remain. The *hard limit* defines the maximum amount of disk blocks or files the user can accumulate under the established disk quotas. The *quota grace period* allows the user to exceed the soft limit for a short period of time (the default value is one week). If the user fails to reduce usage below the soft limit during the specified time, the system will interpret the soft limit as the maximum allocation allowed, and no further storage is allocated to the user. The user can reset this condition by removing enough files to reduce usage below the soft limit.

The disk quota system tracks user and group quotas in the **quota.user** and **quota.group** files that reside in the root directories of file systems enabled with quotas. These files are created with the **quotacheck** and **edquota** commands and are readable with the quota commands.

Recovering from Over-Quota Conditions

To reduce file system usage when you have exceeded quota limits, you can use the following methods:

- Kill the current process that caused the file system to reach its limit, remove surplus files to bring the limit below quota, and retry the failed program.
- If you are running an editor such as vi, use the shell escape sequence to check your file space, remove surplus files, and return without losing your edited file. Alternatively, if you are using the C or Korn shells, you can suspend the editor with the Ctrl-Z key sequence, issue the file system commands, and then return with the **fg** (foreground) command.

- Temporarily write the file to a file system where quota limits have not been exceeded, delete surplus files, and then return the file to the correct file system.

Setting Up the Disk Quota System

Typically, only those file systems that contain user home directories and files require disk quotas. Consider implementing the disk quota system under the following conditions:

- Your system has limited disk space.
- You require more file-system security.
- Your disk-usage levels are large, such as at many universities.

If these conditions do not apply to your environment, you might not want to create disk-usage limits by implementing the disk quota system.

The disk quota system works only with the journaled file system.

Note: Do not establish disk quotas for the **/tmp** file system.

To set up the disk quota system, use the following procedure:

1. Log in with root authority.
2. Determine which file systems require quotas.

   Note: Because many editors and system utilities create temporary files in the **/tmp** file system, it must be free of quotas.
3. Use the **chfs** command to include the **userquota** and **groupquota** quota configuration attributes in the **/etc/filesystems** file. The following example uses the **chfs** command to enable user quotas on the **/home** file system:

   ```
   chfs -a "quota = userquota" /home
   ```

   To enable both user and group quotas on the **/home** file system, type:

   ```
   chfs -a "quota = userquota,groupquota" /home
   ```

   The corresponding entry in the **/etc/filesystems** file is displayed as follows:

   ```
   /home:
   dev         = /dev/hd1
   vfs         = jfs
   log         = /dev/hd8
   mount       = true
   check       = true
   quota       = userquota,groupquota
   options     = rw
   ```
4. Optionally, specify alternate disk quota file names. The **quota.user** and **quota.group** file names are the default names located at the root directories of the file systems enabled with quotas. You can specify alternate names or directories for these quota files with the **userquota** and **groupquota** attributes in the **/etc/filesystems** file.

   The following example uses the **chfs** command to establish user and group quotas for the **/home** file system, and names the **myquota.user** and **myquota.group** quota files:

   ```
   chfs -a "userquota = /home/myquota.user" -a "groupquota = /home
           /myquota.group" /home
   ```

   The corresponding entry in the **/etc/filesystems** the is displayed as follows:

   ```
   /home:
   dev         = /dev/hd1
   vfs         = jfs
   log         = /dev/hd8
   ```

```
mount      = true
check      = true
quota      = userquota,groupquota
userquota  = /home/myquota.user
groupquota = /home/myquota.group
options    = rw
```

5. If they are not previously mounted, mount the specified file systems.
6. Set the desired quota limits for each user or group. Use the **edquota** command to create each user or group's soft and hard limits for allowable disk space and maximum number of files.

   The following example entry shows quota limits for the *davec* user:

   ```
   Quotas for user davec:
   /home: blocks in use: 30, limits (soft = 100, hard = 150)
           inodes in use: 73, limits (soft = 200, hard = 250)
   ```

   This user has used 30 KB of the maximum 100 KB of disk space. Of the maximum 200 files, *davec* has created 73. This user has buffers of 50 KB of disk space and 50 files that can be allocated to temporary storage.

   When establishing disk quotas for multiple users, use the **-p** flag with the **edquota** command to duplicate a user's quotas for another user.

   To duplicate the quotas established for user davec for user nanc, type:

   ```
   edquota -p davec nanc
   ```

7. Enable the quota system with the **quotaon** command. The **quotaon** command enables quotas for a specified file system, or for all file systems with quotas (as indicated in the **/etc/filesystems** file) when used with the **-a** flag.
8. Use the **quotacheck** command to check the consistency of the quota files against actual disk usage.

   > **Note:** It is recommended that you do this each time you first enable quotas on a file system and after you reboot the system.

   To enable this check and to turn on quotas during system startup, add the following lines at the end of the **/etc/rc** file:

   ```
   echo " Enabling filesystem quotas "
   /usr/sbin/quotacheck -a
   /usr/sbin/quotaon -a
   ```

Chapter 3. Auditing

The auditing subsystem enables the system administrator to record security-relevant information, which can be analyzed to detect potential and actual violations of the system security policy.

This section discusses the following topics:

Auditing Subsystem

The auditing subsystem has the following functions:

The system administrator can configure by each of these functions.

Detecting Events

Event detection is distributed throughout the Trusted Computing Base (TCB), both in the kernel (supervisor state code) and the trusted programs (user state code). An auditable event is any security-relevant occurrence in the system. A security-relevant occurrence is any change to the security state of the system, any attempted or actual violation of the system access control or accountability security policies, or both. The programs and kernel modules that detect auditable events are responsible for reporting these events to the system audit logger, that runs as part of the kernel and can be accessed either with a subroutine (for trusted program auditing) or within a kernel procedure call (for supervisor state auditing). The information reported includes the name of the auditable event, the success or failure of the event, and any additional event-specific information that is relevant to security auditing.

Event detection configuration consists of turning event detection on or off, and specifiying which events are to be audited for which users. To activate event detection use the **audit** command to enable or disable the audit subsystem. The **/etc/security/audit/config** file contains the events and users that are processed by the audit subsystem.

Collecting Event Information

Information collection encompasses logging the selected auditable events. This function is performed by the kernel audit logger, which provides both a system call and an intra-kernel procedure call interface that records auditable events.

The audit logger is responsible for constructing the complete audit record, consisting of the audit header, that contains information common to all events (such as the name of the event, the user responsible, the time and return status of the event), and the audit trail, which contains event-specific information. The audit logger appends each successive record to the kernel audit trail, which can be written in either (or both) of two modes:

BIN mode
The trail is written into alternating files, providing for safety and long-term storage.

STREAM mode
The trail is written to a circular buffer that is read synchronously through an audit pseudo-device. STREAM mode offers immediate response.

Information collection can be configured at both the front end (event recording) and at the back end (trail processing). Event recording is selectable on a per-user basis. Each user has a defined set of audit events that are logged in the audit trail when they occur. At the back end, the modes are individually configurable, so that the administrator can employ the back-end processing best suited for a particular environment. In addition, BIN mode auditing can be configured to generate an alert in case the file system space available for the trail is getting too low.

Processing the Audit Trail Information

The operating system provides several options for processing the kernel audit trail. The BIN mode trail can be compressed, filtered, or formatted for output, or any reasonable combination of these before archival storage of the audit trail, if any. Compression is done through Huffman encoding. Filtering is done with standard query language (SQL)-like audit record selection (using the **auditselect** command), which provides for both selective viewing and selective retention of the audit trail. Formatting of audit trail records can be used to examine the audit trail, to generate periodic security reports, and to print a paper audit trail.

The STREAM mode audit trail can be monitored in real time, to provide immediate threat-monitoring capability. Configuration of these options is handled by separate programs that can be invoked as daemon processes to filter either BIN or STREAM mode trails, although some of the filter programs are more naturally suited to one mode or the other.

Event Selection

The set of auditable events on the system defines which occurrences can actually be audited and the granularity of the auditing provided. The auditable events must cover the security-relevant events on the system, as defined previously. The level of detail you use for auditable event definition must maintain a balance between insufficient detail, which makes it difficult for the administrator to understand the selected information, and too much detail, which leads to excessive information collection. The definition of events takes advantage of similarities in detected events. For the purpose of this discussion, a *detected event* is any single instance of an auditable event; for instance, a given event might be detected in various places. The underlying principle is that detected events with similar security properties are selected as the same auditable event. The following list shows a classification of security policy events:

- Subject Events
  - Process creation
  - Process deletion
  - Setting subject security attributes: user IDs, group IDs
  - Process group, control terminal
- Object Events
  - Object creation
  - Object deletion
  - Object open (including processes as objects)
  - Object close (including processes as objects)
  - Setting object security attributes: owner, group, ACL
- Import/Export Events
  - Importing or exporting an object
- Accountability Events
  - Adding a user, changing user attributes in the password database
  - Adding a group, changing group attributes in the group database

  - User login
  - User logoff
  - Changing user authentication information
  - Trusted path terminal configuration
  - Authentication configuration
  - Auditing administration: selecting events and audit trails, switching on or off, defining user auditing classes
- General System Administration Events
  - Use of privilege
  - File system configuration
  - Device definition and configuration
  - System configuration parameter definition
  - Normal system IPL and shutdown
  - RAS configuration
  - Other system configuration
- Security Violations (potential)
  - Access permission refusals
  - Privilege failures
  - Diagnostically detected faults and system errors
  - Attempted alteration of the TCB

Auditing Subsystem Configuration

The auditing subsystem has a global state variable that indicates whether the auditing subsystem is on. In addition, each process has a local state variable that indicates whether the auditing subsystem should record information about this process. Both of these variables determine whether events are detected by the Trusted Computing Base (TCB) modules and programs. Turning TCB auditing off for a specific process allows that process to do its own auditing and not to bypass the system accountability policy. Permitting a trusted program to audit itself allows for more efficient and effective collection of information.

Collecting Auditing Subsystem Information

Information collection addresses **event selection** and **kernel audit trail** modes. It is done by a kernel routine that provides interfaces to log information, used by the TCB components that detect auditable events, and configuration interfaces, used by the auditing subsystem to control the audit logging routine.

Audit Logging

Auditable events are logged by the following interfaces, the user state and supervisor state. The user state portion of the TCB uses the **auditlog** or **auditwrite** subroutine, while the supervisor state portion of the TCB uses a set of kernel procedure calls.

For each record, the audit event logger prefixes an audit header to the event-specific information. This header identifies the user and process for which this event is being audited, as well as the time of the event. The code that detects the event supplies the event type and return code or status and optionally, additional event-specific information (the event tail). Event-specific information consists of object names (for example, files that are refused access or tty used in failed login attempts), subroutine parameters, and other modified information.

Events are defined symbolically, rather than numerically. This lessens the chances of name collisions, without using an event registration scheme. Because subroutines are auditable and the extendable kernel

definition has no fixed switched virtual circuit (SVC) numbers, it is difficult to record events by number. The number mapping would have to be revised and logged every time that the kernel interface was extended or redefined.

Audit Record Format

The audit records consist of a common header, followed by audit trails specific to the audit event of the record. The structures for the headers are defined in the **/usr/include/sys/audit.h** file. The format of the information in the audit trails is specific to each base event and is shown in the **/etc/security/audit/events** file.

The information in the audit header is generally collected by the logging routine to ensure its accuracy, while the information in the audit trails is supplied by the code that detects the event. The audit logger has no knowledge of the structure or semantics of the audit trails. For example, when the **login** command detects a failed login, it records the specific event with the terminal on which it occurred and writes the record into the audit trail using the **auditlog** subroutine. The audit logger kernel component records the subject-specific information (user IDs, process IDs, time) in a header and appends this to the other information. The caller supplies only the event name and result fields in the header.

Audit Logger Configuration

The audit logger is responsible for constructing the complete audit record. You must select the audit events that you want to be logged.

Selecting Audit Events

Audit event selection has the following types:

Per-Process Auditing

To select process events efficiently, the operating system allows the system administrator to define audit classes. An audit class is a subset of the base auditing events in the system. Auditing classes provide for convenient logical groupings of the base auditing events.

For each user on the system, the system administrator defines a set of audit classes that determine the base events that could be recorded for that user. Each process run by the user is tagged with its audit classes.

Per-Object Auditing

The operating system provides for the auditing of object accesses by name; that is, the auditing of specific objects (normally files). By-name object auditing prevents having to cover all object accesses to audit the few pertinent objects. In addition, the auditing mode can be specified, so that only accesses of the specified mode (read/write/execute) are recorded.

Kernel Audit Trail Modes

Kernel logging can be set to BIN or STREAM modes to define where the kernel audit trail is to be written. If the BIN mode is used, the kernel audit logger must be given (before audit startup) at least one file descriptor to which records are to be appended.

BIN mode consists of writing the audit records into alternating files. At auditing startup, the kernel is passed two file descriptors and an advisory maximum bin size. It suspends the calling process and starts writing audit records into the first file descriptor. When the size of the first bin reaches the maximum bin size, and if the second file descriptor is valid, it switches to the second bin and reactivates the calling process. The kernel continues writing into the second bin until it is called again with another valid file descriptor. If at that point the second bin is full, it switches back to the first bin, and the calling process

returns immediately. Otherwise, the calling process is suspended, and the kernel continues writing records into the second bin until it is full. Processing continues this way until auditing is turned off. See the following figure for an illustration of audit BIN mode:

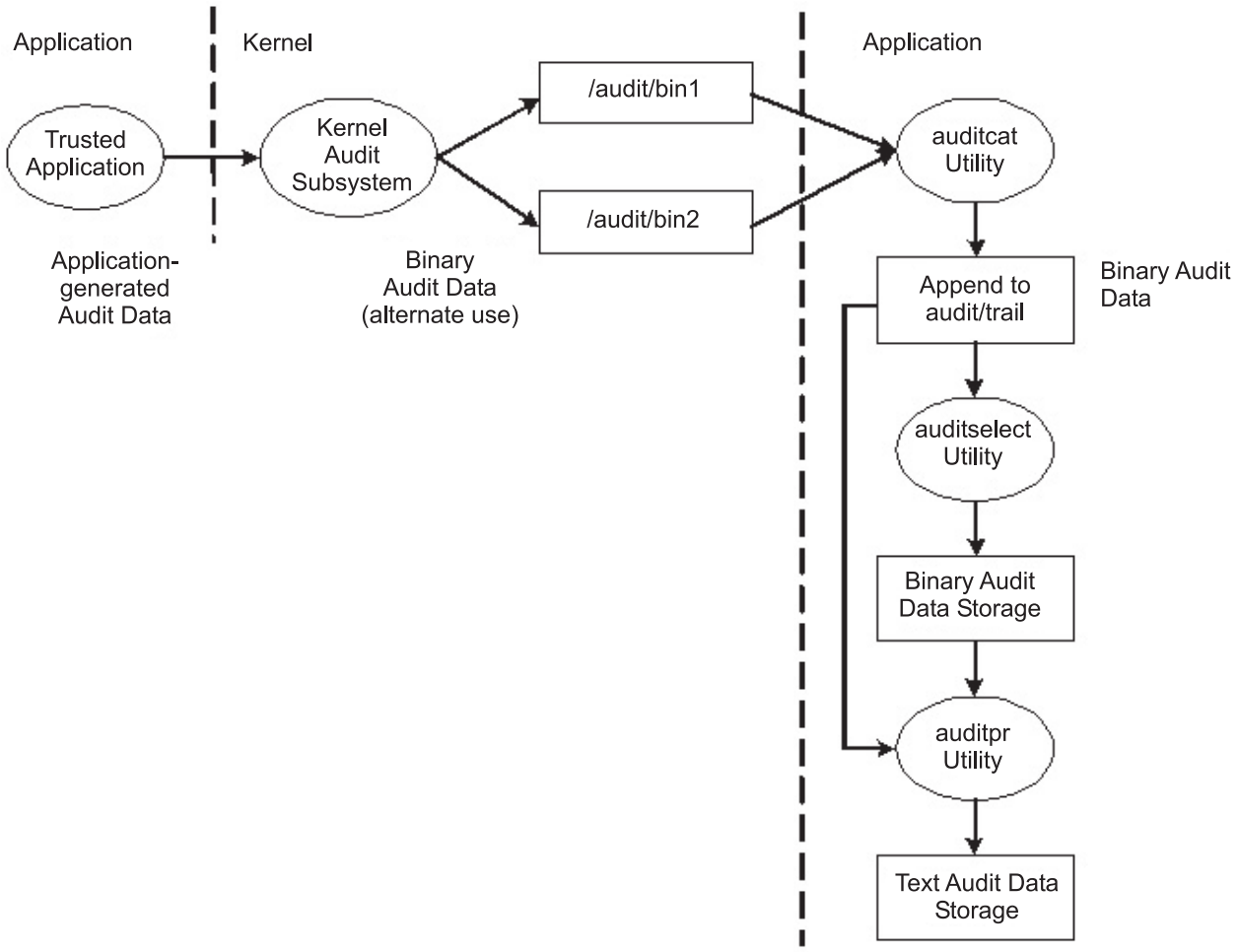


Figure 1. Process of the audit BIN mode.. This illustration shows the process of the audit BIN mode.

The alternating bin mechanism is used to ensure that the audit susbsystem always has something to write to while the audit records are processed. When the audit subsystem switches to the other bin, it empties the first bin content to the **trace** file. When time comes to switch the bin again, the first bin is available. It decouples the storage and analysis of the data from the data generation. Typically, the **auditcat** program is used to read the data from the bin that the kernel is not writing to at the moment. To make sure that the system never runs out of space for the audit trail (the output of the **auditcat** program), the **freespace** parameter can be specified in the **/etc/security/audit/config** file. Should the system have less then the amount of 512-byte blocks specified here, it generates a **syslog** message.

If auditing is enabled, the **binmode** parameter in the **start** stanza in **/etc/security/audit/config** should be set to **panic**. The **freespace** parameter in the **bin** stanza should be configured at minimum to a value that equals 25 percent of the disk space dedicated to the storage of the audit trails. The **bytethreshold** and **binsize** parameters should each be set to 65536 bytes.

In the STREAM mode, the kernel writes records into a circular buffer. When the kernel reaches the end of the buffer, it simply wraps to the beginning. Processes read the information through a pseudo-device called **/dev/audit**. When a process opens this device, a new channel is created for that process.

Optionally, the events to be read on the channel can be specified as a list of audit classes. See the following figure for an illustration of audit STREAM mode:

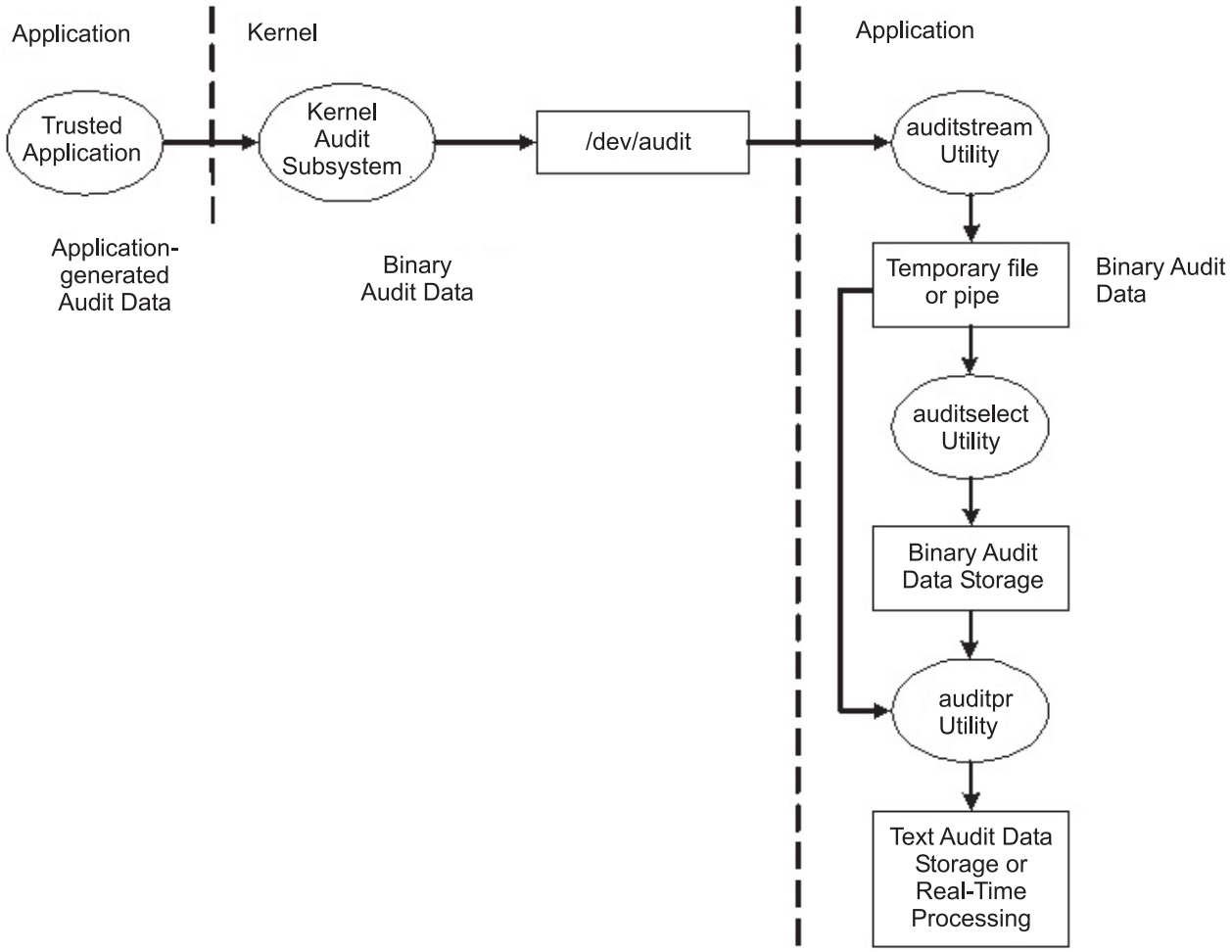


Figure 2. Process of the audit STREAM mode.. This illustration shows the process of the audit STREAM mode.

The main purpose of the STREAM mode is to allow for timely reading of the audit trail, which is desirable for real-time threat monitoring. Another use is to create a trail that is written immediately, preventing any possible tampering with the audit trail, as is possible if the trail is stored on some writable media.

Yet another method to use the STREAM mode is to write the audit stream into a program that stores the audit information on a remote system, which allows central near-time processing, while at the same time protecting the audit information from tampering at the originating host.

Processing Audit Records

The **auditselect**, **auditpr**, and **auditmerge** commands are available to process BIN or STREAM mode audit records. Both utilities operate as filters so that they can be easily used on pipes, which is especially handy for STREAM mode auditing.

auditselect

Can be used to select only specific audit records with SQL like statements. For example, to select only **exec()** events that where generated by user **afx**, type the following:

```
auditselect -e "login==afx && event==PROC_Execute"
```

auditpr

Used to convert the binary audit records into a human readable form. The amount of information displayed depends on the flags specified on the command line. To get all the available information **auditpr** should be run as follows:

```
auditpr -v -hhelrtRpPTc
```

When the **-v** flag is specified, the audit tail which is an event specific string (see the **/etc/security/audit/events** file) is displayed in addition to the standard audit information that the kernel delivers for every event.

auditmerge

Used to merge binary audit trails. This is especially useful if there are audit trails from several systems that need to be combined. The **auditmerge** command takes the names of the trails on the command line and sends the merged binary trail to STDOUT, so you still need to use **auditpr** to make it readable. For example, **auditmerge** and **auditptr** could be run as follows:

```
auditmerge trail.system1 trail.system2 | auditpr -v -hhelrRtpc
```

Using the Audit Susbystem for a Quick Security Check

To monitor a single suspicious program without setting up the audit susbsystm, the **watch** command can be used. It will record either the requested or all events that are generated by the specified program. For example, to see all **FILE_Open** events when running **vi /etc/hosts**, type the following:

```
watch -eFILE_Open -o /tmp/vi.watch vi /etc/hosts
```

The file **/tmp/vi.watch** will show all FILE_Open events for the editor session.

Setting Up Auditing

The following procedure is an overview of the steps you must take to set up an auditing subsystem. For more specific information, refer to the configuration files noted in these steps.

1. Select system activities (events) to audit from the list in the **/etc/security/audit/events** file. If you have added new audit events to applications or kernel extensions, you need to edit the file to add the new events.
   - You add an event to this file if you have included code to log that event in an application program (using the **auditwrite** or **auditlog** subroutine) or in a kernel extension (using the **audit_svcstart**, **audit_svcbcopy**, and **audit_svcfinis** kernel services).
   - Ensure that formatting instructions for any new audit events are included in the **/etc/security/audit/events** file. These specifications enable the **auditpr** command to write an audit trail when it formats audit records.
2. Group your selected audit events into sets of similar items called *audit classes*. Define these audit classes in the classes stanza of the **/etc/security/audit/config** file.
3. Assign the audit classes to the individual users and assign audit events to the files (objects) that you want to audit, as follows:
   - To assign audit classes to an individual user, add a line to the users stanza of the **/etc/security/audit/config** file. To assign audit classes to a user, you can use the **chuser** command.
   - To assign audit events to an object (data or executable file), add a stanza for that file to the **/etc/security/audit/objects** file.
   - You can also specify default audit classes for new users by editing **/usr/lib/security/mkuser.default**. This file holds user attributes that will be used when generating new user IDs. For example, use the **general** audit class for all new user IDs, as follows:

```
user:
    auditclasses = general
    pgrp = staff
    groups = staff
    shell = /usr/bin/ksh
    home = /home/$USER
```

To get all audit events, specify the **ALL** class. When doing so on even a moderately busy systrem, a huge amount of data will be generated. It is typically more practical to limit the number of events that are recorded.

4. In the **/etc/security/audit/config** file, configure the type of data collection that you want using BIN collection, STREAM collection, or both methods. Make sure that audit data does not compete with other data about file space by using a separate file system for audit data. This ensures that there is enough space for the audit data. Configure the type of data collection as follows:
   - To configure BIN collection:
     a. Enable the BIN mode collection by setting `binmode = on` in the **start** stanza.
     b. Edit the **binmode** stanza to configure the bins and trail, and specify the path of the file containing the binmode back-end processing commands. The default file for back-end commands is the **/etc/security/audit/bincmds** file.
     c. Make sure that the audit bins are large enough for your needs and set the **freespace** parameter accordingly to get an alert if the file system is filling up.
     d. Include the shell commands that process the audit bins in an audit pipe in the **/etc/security/audit/bincmds** file.
   - To configure STREAM collection
     a. Enable the STREAM mode collection by setting `streammode = on` in the **start** stanza.
     b. Edit the streammode stanza to specify the path to the file containing the streammode processing commands. The default file containing this information is the **/etc/security/audit/streamcmds** file.
     c. Include the shell commands that process the stream records in an audit pipe in the **/etc/security/audit/streamcmds** file.
5. When you have finished making any necessary changes to the configuration files, you are ready to use the **audit start** command to enable the audit subsystem.
6. Use the **audit query** command to see which events and objects are audited.
7. Use the **audit shutdown** command to deactivate the audit subsystem again.

Selecting Audit Events

The purpose of an audit is to detect activities that might compromise the security of your system. When performed by an unauthorized user, the following activities violate system security and are candidates for an audit:

- Engaging in activities in the Trusted Computing Base
- Authenticating users
- Accessing the system
- Changing the configuration of the system
- Circumventing the auditing system
- Initializing the system
- Installing programs
- Modifying accounts
- Transferring information into or out of the system

The audit system does not have a default set of events to be audited. You have to select events or event classes according to your needs.

To audit an activity, you must identify the command or process that initiates the audit event and ensure that the event is listed in the **/etc/security/audit/events** file for your system. Then you must add the event either to an appropriate class in the **/etc/security/audit/config** file, or to an object stanza in the **/etc/security/audit/objects** file. See the **/etc/security/audit/events** file on your system for the list of audit events and trail formatting instructions. For a description of how audit event formats are written and used, see the **auditpr** command.

After you have selected the events to audit, you must combine similar events into audit classes. Audit classes are then assigned to users.

Selecting Audit Classes

You can facilitate the assignment of audit events to users by combining similar events into audit classes. These audit classes are defined in the classes stanza of the **/etc/security/audit/config** file.

Some typical audit classes might be as follows:

general	Events that alter the state of the system and change user authentication. Audit attempts to circumvent system access controls.
objects	Write access to security configuration files.
kernel	Events in the kernel class are generated by the process management functions of the kernel.

An example of a stanza in the **/etc/security/audit/config** file is as follows:

```
classes:
    general = USER_SU,PASSWORD_Change,FILE_Unlink,FILE_Link,FILE_Rename
    system = USER_Change,GROUP_Change,USER_Create,GROUP_Create
    init = USER_Login,USER_Logout
```

Selecting an Audit Data-Collection Method

Your selection of a data-collection method depends on how you intend to use the audit data. If you need long-term storage of a large amount of data, select BIN collection. If you want to process the data as it is collected, select STREAM collection. If you need both long-term storage and immediate processing, select both methods.

Bin collection	Allows storage of a large audit trail for a long time. Audit records are written to a file that serves as a temporary bin. After the file is filled, the data is processed by the **auditbin** daemon while the audit subsystem writes to the other bin file, and records are written to an audit trail file for storage.
Stream collection	Allows processing of audit data as it is collected. Audit records are written into a circular buffer within the kernel, and are retrieved by reading **/dev/audit**. The audit records can be displayed, printed to provide a paper audit trail, or converted into bin records by the **auditcat** command.

Example of Real-Time File Modification Monitoring

The following example can be used to monitor file access to critical files in real time:

1. Set up a list of critical files to be monitored for changes, for example all files in **/etc** and configure them for **FILE_Write** events in the **objects** file:

   ```
   find /etc -type f | awk '{printf("%s:\n\tw = FILE_Write\n\n",$1)}' >> /etc/security/audit/objects
   ```

2. Set up stream auditing to list all file writes. (This example lists all file writes to the console, but in a production environment you might want to have a backend that sends the events into an Intrusion Detection System.) The **/etc/security/audit/streamcmds** file is similar to the following:

   ```
   /usr/sbin/auditstream | /usr/sbin/auditselect -e "event == FILE_Write" |
   auditpr  -hhelpPRtTc -v > /dev/console &
   ```

3. Set up STREAM mode auditing in **/etc/security/audit/config**, add a class for the file write events and configure all users that should be audited with that class:

```
start:
        binmode = off
        streammode = on

stream:
        cmds = /etc/security/audit/streamcmds

classes:
        filemon = FILE_write

users:
        root = filemon
        afx = filemon
        ...
```

4. Now run **audit start**. All **FILE_Write** events are displayed on the console.

Example of a Generic Audit Log Scenario

In this example, assume that a system administrator wants to use the audit subsystem to monitor a large multi-user server system. No direct integration into an IDS is performed, all audit records will be inspected manually for irregularities. Only a few essential audit events are recorded, to keep the amount of generated data to a manageable size.

The audit events that are considered for event detection are the following:

FILE_Write	We want to know about file writes to configuraion files, so this event will be used with all files in the **/etc** tree.
PROC_SetUserIDs	All changes of user ids
AUD_Bin_Def	Audit bin configuration
USER_SU	The **su** command
PASSWORD_Change	**passwd** command
AUD_Lost_Rec	Notification in case there where lost records
CRON_JobAdd	new **cron** jobs
AT_JobAdd	new **at** jobs
USER_Login	All logins
PORT_Locked	All locks on terminals because of too many invalid attempts

The following is an example of how to generate a generic audit log:

1. Set up a list of critical files to be monitored for changes, such as, all files in **/etc** and configure them for **FILE_Write** events in the **objects**file as follows:

```
find /etc -type f | awk '{printf("%s:\n\tw = FILE_Write\n\n",$1)}' >> /etc/security/audit/objects
```

2. Use the **auditcat** command to set up BIN mode auditing. The **/etc/security/audit/bincmds** file is similar to the following:

```
/usr/sbin/auditcat -p -o $trail $bin
```

3. Edit the **/etc/security/audit/config** file and add a class for the events we have interest. List all existing users and specify the **custom** class for them.

```
start:
        binmode = on
        streammode = off

bin:
        cmds = /etc/security/audit/bincmds
        trail = /audit/trail
        bin1 = /audit/bin1
        bin2 = /audit/bin2
```

```
        binsize = 100000
        freespace = 100000

classes:
        custom = FILE_Write,PROC_SetUser,AUD_Bin_Def,AUD_Lost_Rec,USER_SU, \
                 PASSWORD_Change,CRON_JobAdd,AT_JobAdd,USER_Login,PORT_Locked

users:
        root = custom
        afx = custom
        ...
```

4. Add the **custom** audit class to the **/usr/lib/security/mkuser.default** file, so that new IDs will automatically have the right audit call associated:

```
user:
    auditclasses = custom
    pgrp = staff
    groups = staff
    shell = /usr/bin/ksh
    home = /home/$USER
```

5. Create a new file system named **/audit** using SMIT or the **crfs** command. It should be large enough to hold the two bins and a large audit trail.
6. Now run **audit start**and look at **/audit**: You should see the two bin files and an empty **trail** file initially. After you have used the system for a while, you should have audit records in the **trail** file which can be read with

```
auditpr  -hhelpPRtTc -v | more
```

This example uses only a few events. To see all events, you could specify the classname **ALL** for all users. Ths will generate large amounts of data. You might want to add all events related to user changes and pivilege changes to your **custom** class.

Chapter 4. LDAP Exploitation of the Security Subsystem

The Light Directory Access Protocol (LDAP) defines a standard method for accessing and updating information in a directory (a database) either locally or remotely in a client-server model. The LDAP method is used by a cluster of hosts to allow centralized security authentication as well as access to user and group information. This functionality is intended to be used in a clustering environment to keep authentication, user, and group information common across the cluster.

The LDAP exploitation of the security subsystem is implemented as the LDAP authentication load module. It is conceptually similar to the other load modules such as NIS, DCE, and Kerberos 5. The load modules are defined in the **/usr/lib/security/methods.cfg** file. The implementation of the LDAP authentication load module is at a low level and is handled by the libraries.

After the LDAP authentication load module is enabled to serve user and group information, most high-level APIs, commands, and system-management tools work in their usual manner. An **-R** flag is introduced for most high-level commands to work through different load modules. For example, to create an LDAP user named `joe` from a client machine, use the following command:

```
mkuser -R LDAP joe
```

The client system checks whether the user is an LDAP user through the user's SYSTEM attribute in the **/etc/security/user** file. If the user's SYSTEM attribute is set to LDAP, that user can only authenticate through LDAP. If the SYSTEM attribute in the default stanza is set to LDAP, all users who do not have a SYSTEM attribute set are considered LDAP users. The LDAP keyword can be used with other SYSTEM attribute values as described in "User Authentication" on page 43. The client side communicates to the server through the **secldapclntd** daemon. The daemon accepts requests from applications (through the library APIs), queries the LDAP server, and returns data to the application. The **secldapclntd** daemon is also responsible for caching.

Setting Up an LDAP Security Information Server

To set up a system as an LDAP security information server that serves authentication, user, and group information through LDAP, the LDAP server and client packages must be installed. The LDAP server must be configured as a client as well as a server. A DB2 database is also required by the LDAP server. If the Secure Socket Layer (SSL) is required, then the GSKit must be installed. The system administrator must create a key using the **ikeyman** command. The certificate of the server key must be carried to the clients.

The **mksecldap** command can be used to set up an LDAP security information server. It sets up a database named **ldapdb2**, populates the database with the user and group information from the local host, and sets the LDAP server administrator DN (distinguished name) and password. Optionally, it can set up SSL for client/server communication. Then **mksecldap** loads a server plugin (**libsecldap.a**) and starts the LDAP server process (**slapd**). The **mksecldap** command also adds an entry into the **/etc/inittab** file to start the LDAP server at every reboot. The entire LDAP server setup is done through the **mksecldap** command, which updates the **slapd.conf** file (SecureWay® Directory Version 3.1) or **slapd32.conf** file (SecureWay Directory Version 3.2). There is no need to configure the LDAP Web management interface.

All users and groups from the local system are migrated to LDAP server during LDAP server setup. Select one of the following LDAP schemas for this step:

AIX-specific schema
: Includes aixAccount and aixAccessGroup object class. This schema offers a full set of attributes for AIX users and groups.

NIS schema (RFC 2307)
: Includes posixAccount and posixGroup account and is used by several vendor's directory products. The NIS schema only defines a small subset of attributes that AIX uses.

NIS schema with full AIX support

Includes posixAccount and posixGroup object classes plus the aixAusAccount and aixAusGroup object classes. The aixAusAccount and aixAuxGroup object classes provide the attributes which are used by AIX but not defined by the NIS schema. Setting up the LDAP server using NIS schema with full AIX support is recommended unless setting up an AIX-specific schema LDAP server for compatibility with the existing LDAP servers is necessary.

All the user and group information is stored under a common AIX tree (suffix). The default suffix is "cn=aixsecdb". The **mksecldap** command accepts a user-supplied suffix through the **-d** flag. If the user-supplied suffix does not have "cn=aixsecdb" as its first RDN (Relative Distinguished Name), the **mksecldap** command prefixes the user-supplied suffix with "cn=aixsecdb". This AIX tree is ACL (Access Control List) protected. A client must bind as the LDAP server administrator to be able to access the AIX tree.

The **mksecldap** command works even if an LDAP server has been set up for other purposes such as, for example, for blue page information. In this case, **mksecldap** adds the AIX tree and populates it with the AIX security information to the existing database. This tree is ACL-protected independently from other trees. In this case, the LDAP server works as usual, in addition to serving as an AIX LDAP Security Server.

Note: Backing up the existing database before running the **mdsecldap** command to set up the security server to share the same database is recommended.

After the LDAP security information server is successfully set up, the same host must be set up as a client so that LDAP user and group management can be completed and LDAP users can log in to this server.

If the LDAP security information server setup is not successful, you can undo the setup by running the **mksecldap** command with the **-U** flag. This restores the **slapd.conf** (or **slapd32.conf**) file to its pre-setup state. Run the **mksecldap** command with the **-U** flag after any unsuccessful setup attempt before trying to run the **mksecldap** command again. Otherwise, residual setup information might remain in the configuration file and cause a subsequent setup to fail. As a safety precaution, the undo option does not do anything to the database or to its data, because the database could have existed before the **mksecldap** command was run. Remove any database manually if it was created by the **mksecldap** command. If the **mksecldap** command has added data to a pre-existing database, decide what steps to take to recover from a failed setup attempt.

Beginning with AIX 5.2 , the **mknisldap** command can also be used to set up the LDAP security information server. The **mknisldap** command sets up the server in the same way as the **mksecldap** command, and it migrates other NIS data as well as users and groups to the LDAP server.

For more information on setting up an LDAP security information server, see the **mksecldap** command.

Setting Up an LDAP Client

Each client must have the LDAP client package installed. If the SSL is required, the GSKit must be installed, a key must be created, and the LDAP server SSL key certificate must be added to this key.

The **mksecldap** command can be used to set up the client. To have this client contact the LDAP security information server, the server name must be supplied during setup. The server's administrator domain name and password are also needed for client access to the AIX tree on the server. The **mksecldap** command saves the server administrator domain name, password, server name, AIX tree domain name on the server, and the SSL key path and password to the **/etc/security/ldap/ldap.cfg** file.

Multiple servers can be supplied to the **mksecldap** command during client setup. In this case, the client contacts the servers in the supplied order and establishes connection to the first server that the client can

successfully bind to. If bad connection occurs between the client and the server, a reconnection request is tried using the same logic. The Security LDAP exploitation model does not support referral. It is important that the replicate servers are kept synchronized.

The client communicates to the LDAP security information server through a client side daemon (**secldapclntd**). If the LDAP load module is enabled on this client, high-level commands eventually find this daemon through the library APIs. The daemon queries the server and returns the information back to the caller.

Other fine-tuning options can be supplied to the **mksecldap** command during client setup, such as settings for the number of threads used by the daemon, the cache entry size, and the cache expiration timeout. These options are for experienced users only. For most environments, the default values are sufficient.

A comma-separated user list can be supplied to the **mksecldap** command during client setup. These users' SYSTEM attributes are set to LDAP. Once this is done, these users can only authenticate through the LDAP load module. Note that the **mksecldap** command does not add these users to the LDAP security information server to avoid duplicate user IDs in the LDAP database. Using the **mkuser** command with **-R LDAP** flag to create these users on an LDAP server is recommended.

In the final steps of the client setup, the **mksecldap** command starts the client-side daemon and adds an entry in the **/etc/inittab** file so the daemon starts at every reboot. You can check whether the setup is successful by checking the **secldapclntd** process. Provided that the LDAP security information server is setup and running, this daemon will be running if the setup was successful.

LDAP User Management

You can manage users and groups on an LDAP security information server from any LDAP client by using high-level commands. An **-R** flag added to most of the high-level commands can manage users and groups using LDAP as well as other authentication load modules such as DCE, NIS, and Kerberos 5. For more information concerning the use of the **-R** flag, refer to each of the user or group management commands.

To enable a user to authenticate through LDAP, run the **chuser** command to change the user's SYSTEM attribute value to LDAP. By setting the SYSTEM attribute value according to the defined syntax, a user can be authenticated through more than one load module (for example, compat and LDAP). For more information on setting users' authentication methods, see "User Authentication" on page 43 and the SYSTEM attribute syntax defined in the **/etc/security/user** file.

A user can become an LDAP user at client setup time by running the **mksecldap** command with the **-u** flag in either of the following forms:

1. Run `mksecldap -c -u user1,user2,...` , where **user1,user2,...** is a list of users. The users in this list can be either locally defined or remotely LDAP-defined users. The SYSTEM attribute is set to LDAP in each of the above users' stanzas in the **/etc/security/user** file. Such users are only authenticated through LDAP. The users in this list must exist on the LDAP security information server; otherwise, they can not log in from this host. Run the **chuser** command to modify the SYSTEM attribute and allow authentication through multiple methods (for example, both local and LDAP).
2. Run "mksecldap -c -u ALL" . This command sets the SYSTEM attribute to LDAP in each user's stanza in the **/etc/security/user** file for all locally defined users. All such users only authenticate through LDAP. The locally defined users must exist on the LDAP security information server; otherwise they can not log in from this host. A user that is defined on the LDAP server but not defined locally cannot log in from this host. To allow a remotely LDAP-defined user to log in from this host, run the **chuser** command to set the SYSTEM attribute to LDAP for that user.

Alternatively, you can enable all LDAP users, whether they are defined locally or not, to authenticate through LDAP on a local host by modifying the ″default″ stanza of the **/etc/security/user** file to use ″LDAP″ as its value. All users that do not have a value defined for their SYSTEM attribute must follow what is defined in the default stanza. For example, if the default stanza has "SYSTEM = ″compat″" , changing it to "SYSTEM = ″compat OR LDAP″" allows authentication of these users either through AIX or LDAP. Changing the default stanza to "SYSTEM = ″LDAP″" enables these users to authenticate exclusively through LDAP. Those users who have a SYSTEM attribute value defined are not affected by the default stanza.

LDAP Host Access Control

AIX provides user-level host access (login) control for a system. Administrators can configure LDAP users to log in to an AIX system by setting their SYSTEM attribute to LDAP. The SYSTEM attribute is in the **/etc/security/user** file. The **chuser** command can be used to set its value, similar to the following:

```
# chuser -R LDAP SYSTEM=LDAP registry=LDAP foo
```

Note: With this type of control, do not set the default SYSTEM attribute to LDAP, which allows all LDAP users to login to the system.

This sets the LDAP attribute to allow user `foo` to log in to this system. It also sets the registry to LDAP, which allows the login process to log `foo`'s login attempts to LDAP, and also allows any user management tasks done on LDAP.

The administrator needs to run such setup on each of the client systems to enable login by certain users.

Starting with AIX 5.2, AIX has implemented a feature to limit a LDAP user only to log in to certain LDAP client systems. This feature allows centralized host access control management. Administrators can specify two host access control lists for a user account: an allow list and a deny list. These two user attributes are stored in the LDAP server with the user account. A user is allowed access to systems or networks that are specified in the allow list, while he is denied access to systems or networks in the deny list. If a system is specified in both the allow list and the deny list, the user is denied access to the system. There are two ways to specify the access lists for a user: with the **mkuser** command when the user is created or with the **chuser** command for a existing user. For backward compatibility, if both the allow list and deny list do not exist for a user, the user is allowed to login to any LDAP client systems by default. To exploit this host access control feature, it is strongly recommended that all LDAP client systems are upgraded to AIX 5.2 or later so that users with both allow and deny lists defined can not log in to specific systems.

Examples of setting allow and deny permission lists for users are the following:

```
# mkuser -R LDAP hostsallowedlogin=host1,host2 foo
```

This creates a user `foo`, and user `foo` is only allowed to log in to `host1` and `host2`.

```
# mkuser -R LDAP hostsdeniedlogin=host2 foo
```

This create user `foo`, and user `foo` can log in to any LDAP client systems except `host2`.

```
# chuser -R LDAP hostsallowedlogin=192.9.200.1 foo
```

This sets user `foo` with permission to log in to the client system at address `192.9.200.1`.

```
# chuser -R LDAP hostsallowedlogin=192.9.200/24 \
hostsdeniedlogin=192.9.200.1 foo
```

This sets user `foo` with permission to log in to any client system within the `192.9.200/24` subnet , except the client system at address `192.9.200.1`.

For more information, see the **chuser** command.

LDAP Security Information Server Auditing

SecureWay Directory version 3.2 provides a default server audit logging function. Once enabled, this default audit plugin logs LDAP server activities to a log file. See the LDAP documentation in *Packaging Guide for LPP Installation* for more information on this default audit plugin.

An LDAP security information server auditing function has been implemented in AIX 5.1 and later, called the *LDAP security audit plugin*. It is independent of the SecureWay Directory default auditing service, so that either one or both of these auditing subsystems can be enabled. The AIX audit plugin records only those events that update or query the AIX security information on an LDAP server. It works within the framework of AIX system auditing.

To accommodate LDAP, the following audit events are contained in the **/etc/security/audit/event** file:

- **LDAP_Bind**
- **LDAP_Unbind**
- **LDAP_Add**
- **LDAP_Delete**
- **LDAP_Modify**
- **LDAP_Modifydn**
- **LDAP_Search**

An **ldapserver** audit class definition is also created in the **/etc/security/audit/config** file that contains all of the above events.

To audit the LDAP security information server, add the following line to each user's stanza in the **/etc/security/audit/config** file:

```
ldap = ldapserver
```

Because the LDAP security information server audit plugin is implemented within the frame of the AIX system auditing, it is part of the AIX system auditing subsystem. Enable or disable the LDAP security information server audit using system audit commands, such as **audit start** or **audit shutdown**. All audit records are added to the system audit trails, which can be reviewed with the **auditpr** command. For more information, see Chapter 3, "Auditing" on page 47.

LDAP Commands

The mksecldap Command

The **mksecldap** command can be used to set up IBM SecureWay Directory servers and clients for security authentication and data management. This command must be run on the server and all clients.

Notes:

1. The client (**-c** flag) and the server (**-s** flag) options cannot be run at the same time. When setting up a server, the **mksecldap** command should be run twice on that machine. Once to set up the server, and again to set up the client.
2. The SecureWay Directory server configuration file is **/etc/slapd32.conf** for AIX 3.2 or later. AIX 5.2 only supports SecureWay Directory 3.2 and later.

For server setup, make sure that the **ldap.server** fileset is installed. When installing the **ldap.server** fileset, the **ldap.client** fileset and the backend DB2 software are automatically installed as well. No DB2 preconfiguration is required to run this command for LDAP server setup. When you run the **mksecldap** command to set up the server, the command will:

1. Create a DB2 instance with **ldapdb2** as the default instance name.

2. Create a DB2 database with **ldapdb2** as the default database name. If a database already exists, **mksecldap** will bypass the above two steps. (This is the case when the LDAP server has been set up for other usage.) The **mksecldap** command will use the existing database to store the AIX user/group data.
3. Create the AIX tree DN (suffix). If no baseDN is supplied from the command line, the default suffix is set to **cn=aixdata** and the user/group data is placed to the **cn=aixsecdb,cn=aixdata** DN. This is the recommended case. Otherwise, the **mksecldap** command takes the user-supplied DN and prefixes it with **cn=aixdata** and makes the newly constructed DN the suffix. This behavior is shown in the following table. The value within the brackets represents the optional user supplied DN from command line.

CMD-line DN:	[o=ibm]
suffix:	cn=aixdata[,o=ibm]
sucurrity DN:	cn=aixsecdb,cn=aixdata[,o=ibm]
user DN:	ou=aixuser,cn=aixsecdb,cn=aixdata[,o=ibm]
group DN:	ou=aixgroup,cn=aixsecdb,cn=aixdata[,o=ibm]

In case the LDAP server has already been setup in the local system, the **mksecldap** command searches for the **cn=aixsecdb** keyword from the suffixes defined in the **slapd32.conf** configuration file and from the database. If it finds the keyword, it assumes that **mksecldap** has been run, and bypasses the base DN setup step and the user/group migration step, and exits.

If **cn=aixsecdb** is not found in the suffixes and the database, the **mksecldap** command checks for the **cn=aixdata** keyword. **cn=aixdata** is a common base DN shared by various AIX LDAP components. If the **mksecldap** command find the keyword, it compares it with the user supplied DN. If they are the same, the users/groups will be put under the **cn=aixsecdb,cn=aixdata,[userDN]**. If they are different, the **mksecldap** command prints an error message warning the existence of the **cn=aixdata,...** DN, and it will not migrate the users/groups under the user supplied DN. You can choose to use the existing **cn=aixdata,...** by running the **mksecldap** command again with that existing DN.

4. Migrates the data from the security database files from the local host into the LDAP database. Depending on the **-S** option, the **mksecldap** command migrates users/groups using one of the three LDAP schemas:
   - **AIX** - AIX schema (**aixaccount** and **aixaccessgroup** objectclasses)
   - **RFC2307** - RFC 2307 schema (**posixaccount**, **shadowaccount**, and **posixgroup** objectclasses)
   - **RFC2307AIX** - RFC 2307 schema with full AIX support (**posixaccount**, **shadowaccount**, and **posixgroup objectclasses**, plus the **aixauxaccount** and **aixauxgroup** object classes).

     Attention: Systems running AIX 4.3 and AIX 5.1 which are configured as LDAP clients will only work with servers of AIX schema type. They will not talk to ldap servers of RFC2307 or RFC2307AIX types.
5. Set the LDAP server administrator DN and password. This name/password pair is also used for access control of the AIX tree.
6. Set the SSL (secure socket layer) for secure data transfer between this server and the clients. This setup requires the **GSKIT** to be installed.

   Note: If this option is used, the SSL key must be created before running the **mksecldap** command. Otherwise the server may not be able to start.
7. Installs the **/usr/ccs/lib/libsecldapaudit.a**, a LDAP server plug-in. This plugin supports AIX audit of the LDAP server.
8. Start/restart the LDAP server after all the above is done.
9. Add the LDAP server process (**slapd**) to **/etc/inittab** to have the LDAP server start after reboot.
10. With the **-U** option, undo a previous setup for the server configuration file. The first time you run the **mksecldap** command, it saves two copies of the **slapd32.conf** server configuration file. One is saved

to **/etc/security/ldap/slap32.conf.save.orig** and the other to **/etc/ security/ldap/slapd32.conf.save**. Each subsequent run of **mksecldap**, the **current slapd32.conf** is only saved to **/etc/security/ldap/slapd32.conf.save** file. The undo option restores the **/etc/slapd32.conf** server configuration file with the **/etc/security/ ldap/slapd32.conf.save** copy.

Note: The undo option applies to the server configuration file only. It has no effect on the database.

Note: All the LDAP configuration is saved into the **/etc/slapd32.conf** LDAP server configuration file. For client setup, make sure the LDAP server has been setup and is running. The **mksecldap** command does the follows during for client setup:

1. Saves the LDAP server(s)' host name.
2. Saves the user base DN and group base DN of the server. If no **-d** option is supplied from command line, the **mksecldap** command searches the LDAP server for **aixaccount**, **aixaccessgroup**, **posixaccount**, **posixgroup**, and **aixauxaccount** objectclasses from the LDAP server, and set up the base DNs accordingly. If the server has multiple user/group bases, you must supply the **-d** option with a RDN so that the **mksecldap** command can setup the base DNs to the ones within that RDN.

   If the **posixaccount** objectclass is found during client setup, **mksecldap** will also try to search for base DNs for these entities: hosts, networks, services, netgroups, protocols, and rpc from the server and save any that is found.
3. Determines the schema type used by the LDAP server - **AIX** specific schema, **RFC 2307** schema, or **RFC 2307** schema with full AIX support (see objectclasses listed in step 2). It sets the objectclasses and attribute maps in the **/etc/security/ldap/ ldap.cfg** file accordingly. The **mksecldap** command does not recognize other schema types, so clients must be setup manually.
4. Sets SSL for secure data transfer between this host and the LDAP server. This step requires that the client SSL key and the key password are created in advance, and the server must be setup to use SSL for the client SSL to work.
5. Saves the LDAP server administrator DN and password. The DN/password pair must be the same as the pair specified during server setup.
6. Sets the cache size in terms of the number of entries used by the client side daemon. Valid values are 100-10,000 for users and 10-1,000 for groups. The default value is 1,000 for users and 100 for groups.
7. Sets the cache timeout of the client side daemon. Valid values are 60-3600 seconds. The default is 300 seconds. Set this value to 0 to disable caching.
8. Sets the number of threads used by the client side daemon. Valid values are 1-1,000. The default is 10.
9. Sets the time interval in seconds that the client daemon checks for the LDAP server status. Valid value are 60-3,600 seconds. The default is 300.
10. Optionally sets the list of users or all users to use LDAP by modifying their SYSTEM line in the **/etc/security/user** file. For more information on enabling ldap login, see the following note.
11. Starts the client daemon process (**secldapclntd**).
12. Adds the client side daemon process to **/etc/inittab** to have this daemon start after a reboot.
13. With the **-U** option, undo a previous setup to the **/etc/security/ldap/ldap.cfg** file.

Note: The client configuration data is saved to the **/etc/security/ldap/ldap.cfg** file. Setting the SYSTEM to `LDAP` for the default stanza of **/etc/security/user** only allows LDAP users to login to the system. Setting the SYSTEM to `LDAP` or `compat` allows both LDAP users and local users to login to the system.

Examples

1. To setup a LDAP server of AIX specific schema for users and groups, enter:

```
mksecldap -s -a cn=admin -p adminpwd -S aix
```

This sets up a LDAP server with LDAP server administrator DN being **cn=admin**, password being **adminpwd**. User and group data is migrated from local files to the default **cn=aixdata** suffix.

2. To setup a LDAP server with a baseDN other than the default and with SSL secure communication , enter:

   ```
   mksecldap -s -a cn=admin -p adminpwd -d o=mycompany,c=us -S rfc2307 \ -k /usr/ldap/serverkey.kdb
    -w keypwd
   ```

   This sets up a LDAP server with LDAP server administrator DN being **cn=admin**, password being **adminpwd**. User and group data is migrated from local files to the **cn=aix-data,o=mycompany,c=us** suffix. The LDAP server uses SSL communications by using the key stored at **/usr/ldap/serverkey.kdb**. The password to the key, **keypwd**, must also be supplied. Users and groups are migrated with the RFC 2307 schema.

3. To undo a previous server setup:

   ```
   mksecldap -s -U
   ```

   This undoes the previous setup to the **/etc/slapd32.conf** server configuration file. For safety reasons, this does not remove any database entries or database created my a previous setup. If they are not needed any more, remove the database entries/database manually.

4. To setup a client to use the **server1.ibm.com** and **server2.ibm.com** LDAP servers, enter:

   ```
   mksecldap -c -a cn=admin -p adminpwd -h server1.ibm.com,server2.ibm.com
   ```

   The LDAP server adminstrator DN and password must be supplied for this client to authenticate to the server. The **mksecldap** command contacts the LDAP server for schema type used, and sets up the client accordingly. Without the **-d** option from the command line, the entire server DIT is searched for the user base DN and the group base DN.

5. To setup the client to talk to the **server3.ibm.com** LDAP server using SSL, enter:

   ```
   mksecldap -c -a cn=admin -p adminpwd -h server3.ibm.com -d o=mycompany,c=us -k /usr/ldap/clientkey.kdb -w keypwd -u user1,user2
   ```

   This sets up a LDAP client similar to case 3, but with SSL communication. The **mksecldap** command searches the **o=mycompany,c=us** RDN for user base DN and group base DN. Account user1 and user2 are configured to authenticate through LDAP.

   Note: The **-u ALL** option enables all LDAP users to login to this client.

6. To undo a previous client setup, enter:

   ```
   mksecldap -c -U
   ```

   This undo the previous setup to the **/etc/security/ldap/ldap.cfg** file. This does not remove the **SYSTEM=LDAP** and **registry=LDAP** from the **/etc/security/user** file.

For more information on the **mksecldap** command, see **mksecldap** in the *AIX 5L Version 5.2 Commands Reference*.

The secldapclntd Daemon

The **secldapclntd** daemon accepts requests from the LDAP load module, forwards the request to the LDAP Security Information Server, and passes the result from the server back to the LDAP load module. This daemon reads the configuration information defined in the **/etc/security/ldap/ldap.cfg** file during its startup, and authenticates to the LDAP Security Information Server using the server administrator's distinguished name and password, and establishes a connection between the local host and the server.

If multiple servers are specified in the **/etc/security/ldap/ldap.cfg** file, the **secldapclntd** daemon connects to all of the servers. At a specific time, however, it talks to only one of them. The **secldapclntd** daemon can detect when the server it talks to is down, and automatically talks to another available server. It can also detect when a server becomes available again, and re-establishes connection to that server (but it continues to talk to the server it was talking to). This auto-detect feature is done by the **secldapclntd**

daemon checking on each of the servers periodically. The time interval between subsequent checking is defaulted to 300 seconds, and can be changed at the daemon startup time from command line or by modify the corresponding values of the **/etc/ security/ldap/ldap.cfg** file.

At startup, the **secldapclntd** daemon tries to establish a connection to the LDAP servers. If it cannot connect to any of the servers, it goes to sleep, and tries again in 30 seconds. It repeats this process twice, and if it still cannot establish any connection, the **secldapclntd** daemon process exits.

The **secldapclntd** daemon is a multi-threaded program. The default number of threads used by this daemon is 10. An administrator can fine-tune the system performance by adjusting the number of threads used by this daemon.

The **secldapclntd** daemon caches information retrieved from the LDAP Security Information Server for performance purpose. If the requested data can be found in the cache and the cache entry is not expired, the data in the cache is handed back to the requester. Otherwise, the **secldapclntd** daemon makes a request to the LDAP Security Information Server for the information.

The valid number of cache entries for users is in the range of 100-10,000, and that for groups is in the range of 10-1,000. The default is 1000 entries for users, and 100 entries for groups.

The cache timeout or TTL (time to live) can be from 60 seconds to 1 hour (60*60=3600 seconds). By default, a cache entry expires in 300 seconds. If the cache timeout is set to 0, the caching feature is disabled.

Examples

1. To start the **secldapclntd** daemon, type:

   ```
   /usr/sbin/secldapclntd
   ```

2. To start the **secldapclntd** with using 20 threads and cache timeout value of 600 seconds, type:

   ```
   /usr/sbin/secldapclntd -p 20 -t 600
   ```

It is recommended that you start the **secldapclntd** daemon by running the **start-secldapclntd** command. It is also recommended that you specify these values in the **/etc/security/ldap/ldap.cfg** file, so that these values will be used each time you start the **secldapclntd** process.

For more information on the **secldapclntd** daemon, see **secldapclntd** in the *AIX 5L Version 5.2 Commands Reference*.

LDAP Management Commands

start-secldapclntd Command

The **start-secldapclntd** command starts the **secldapclntd** daemon if it is not running. It does not do anything if the **secldapclntd** daemon is already running. The script also cleans the portmapper registration (if there is any) from previous **secldapclntd** daemon process before it starts the **secldapclntd** daemon. This prevents the startup failure of the new daemon process from portmap-per registration failure.

Examples:

1. To start the **secldapclntd** daemon, type:

   ```
   /usr/sbin/start-secldapclntd
   ```

2. To start the **secldapclntd** with using 20 threads and cache timeout value of 600 seconds, type:

   ```
   /usr/sbin/start-secldapclntd -p 20 -t 600
   ```

   It is recommended that you specify these values in the **/etc/security/ldap/ldap.cfg** file, so that these values will be used each time you start the **secldapclntd** process.

For more information on the **start-secldapclntd** command, see **start-secldapclntd** in the *AIX 5L Version 5.2 Commands Reference*.

stop-secldapclntd Command

The **stop-secldapclntd** command terminates the running **secldapclntd** daemon process. It returns an error if the **secldapclntd** daemon is not running.

Example: To stop the running **secldapclntd** daemon process, type:

```
/usr/sbin/stop-secldapclntd
```

For more information on the **stop-secldapclntd** command, see **stop-secldapclntd** in the *AIX 5L Version 5.2 Commands Reference*.

restart-secldapclntd Command

The **restart-secldapclntd** script stops the **secldapclntd** daemon if it is running, and then restarts it. If the **secldapclntd** daemon is not running, it simply starts it.

Examples:

1. To restart the **secldapclntd** daemon, type:

   ```
   /usr/sbin/restart-secldapclntd
   ```

2. To restart the **secldapclntd** with using 30 threads and cache timeout value of 500 seconds, type:

   ```
   /usr/sbin/restart-secldapclntd -p 30 -t 500
   ```

For more information on the **restart-secldapclntd** command, see **restart-secldapclntd** in the *AIX 5L Version 5.2 Commands Reference*.

ls-secldapclntd Command

The **ls-secldapclntd** command lists the **secldapclntd** daemon status. The information returned includes the following:

- The LDAP server the **secldapclntd** daemon is talking to
- The LDAP server port number
- The version of the LDAP protocol used
- User base DN
- Group base DN
- System (id) base DN
- User cache size
- User cache size used
- Group cache size
- Group cache size used
- Cache time out (time to live) value
- **secldapclntd** to LDAP server heart beat interval
- Number of thread used by **secldapclntd** daemon
- User objectclass used in the LDAP server
- Group objectclass used in the LDAP server

Example:

1. 1.To list the status of the **secldapclntd** daemon, type:

   ```
   /usr/sbin/ls-secldapclntd
   ```

For more information on the **ls-secldapclntd** command, see **ls-secldapclntd** in the *AIX 5L Version 5.2 Commands Reference*.

flush-secldapclntd Command

The **flush-secldapclntd** command clears the cache for the **secldapclntd** daemon process.

Example:

1. To flush the **secldapclntd** daemon cache, type:

   ```
   /usr/sbin/flush-secldapclntd
   ```

For more information on the **flush-secldapclntd** command, see **flush-secldapclntd** in the *AIX 5L Version 5.2 Commands Reference*.

sectoldif Command

The **sectoldif** command reads users and groups defined locally, and prints the result to **stdout** in ldif format. If redirected to a file, the result can be added to a LDAP server with the **ldapadd** command or the **db2ldif** command.

The **-S** option specifies the schema type used for the ldif output. The **sectoldif** command accepts three schema types:

- **AIX** - AIX schema (**aixaccount** and **aixaccessgroup** objectclasses)
- **RFC2307** - RFC 2307 schema (**posixaccount**, **shadowaccount**, and **posixgroup** objectclasses)
- **RFC2307AIX** - RFC 2307 schema with full AIX support (**posixaccount**, **shadowaccount** , and **posixgroup** objectclasses, plus the **aixauxaccount** and **aixauxgroup** objectclasses).

The **sectoldif** command is called by the **mksecldap** command to migrate users and groups during LDAP server setup. Be cautious when migrating additional users and groups from other systems to the LDAP server using the **sectoldif** output. The **ldapadd** and **db2ldif** commands check only for entry name (user name or group name) but not for the numeric id when adding entries, migrating users and groups from multiple systems using **sectoldif** output may result in sharing of a numeric id by multiple accounts, which is a security violation.

Examples:

1. To print all users and groups defined locally, enter the following:

   ```
   sectoldif -d cn=aixsecdb,cn=aixdata -S rfc2307aix
   ```

   This prints all users and groups defined locally to **stdout** in ldif format. User entries and group entries are represented using the rfc2307aix schema type. The base DN is set to `cn=aixsecdb, cn=aixdata`.
2. To print only locally defined user foo, enter the following:

   ```
   sectoldif -d cn=aixsecdb,cn=aixdata -u foo
   ```

   This prints locally defined user foo to **stdout** in ldif format. Without the **-S** option, the default AIX schema type is used to represent foo's ldif output.

For more information on the **sectoldif** command, see **sectoldif** in the *AIX 5L Version 5.2 Commands Reference*.

The ldap.cfg File Format

The **/etc/security/ldap/ldap.cfg** file contains information for the **secldapclntd** daemon to start and function properly as well as information for fine tuning the daemon's performance. The **/etc/security/ldap/ldap.cfg** file is updated by the **mksecldap** command at client setup.

The **/etc/security/ldap/ldap.cfg** file may contain the following fields:

ldapservers	Specifies a comma separated LDAP Security Information Servers. These servers can either be the primary server and/or replica of the primary server.

ldapadmin	Specifies the administrator DN of the LDAP Security Information Server(s).
ldapadmpwd	Specifies the password of the administrator DN.
useSSL	Specifies whether to use SSL communication. Valid values are ON and OFF. The default is OFF. **Note:** You will need the SSL key and the password to the key to enable this feature.
ldapsslkeyf	Specifies the full path to the SSL key.
ldapsslkeypwd	Specifies the password to the SSL key. **Note:** Comment out this line to use stashed password. The password stash file must reside in the same directory as the SSL key itself, and must have the same name as the key file, but with an extension of **.sth** instead of **.kdb**.
userattrmappath	Specifies the full path to the AIX-LDAP attribute map for users.
groupattrmappath	Specifies the full path to the AIX-LDAP attribute map for groups.
idattrmappath	Specifies the full path to the AIX-LDAP attribute map for IDs. These IDs are used by the **mkuser** command when creating LDAP users.
userbasedn	Specifies the user base DN.
groupbasedn	Specifies the group base DN.
idbasedn	Specifies the ID base DN.
hostbasedn	Specifies the host base DN.
servicebasedn	Specifies the service base DN.
protocolbasedn	Specifies the protocol base DN.
networkbasedn	Specifies the network base DN.
netgroupbasedn	Specifies the netgroup base DN.
rpcbasedn	Specifies the RPC base DN.
userclasses	Specifies the objectclasses used for user entry.
groupclasses	Specifies the objectclasses used for group entry.
ldapversion	Specifies the LDAP server protocol version. Default is 3.
ldapport	Specifies the port that the LDAP server listens to. Default is 389.
ldapsslport	Specifies the SSL port that the LDAP server listens to. Default is 636.
followaliase	Specifies whether to follow aliases. Valid values are `NEVER`, `SEARCHING`, `FINDING`, and `ALWAYS`. Default is `NEVER`.
usercachesize	Specifies the user cache size. Valid values are 100 - 10,000 entries. Default is 1,000.
groupcachesize	Specifies the group cache size. Valid values are 10 - 1,000 entries. Default is 100.
cachetimeout	Specifies the cache TTL (time to live). Valid values are 60 - 3,600 seconds. Default is 300. Set to 0 to disable caching.
heartbeatinterval	Specifies the interval in seconds that the client contacts the server for server status. Valid values are 60 - 3,600 seconds. Default is 300.
numberofthread	Specifies the number of threads for the **secldapclntd** daemon. Valid values are 1 - 1,000. Default is 10.

For more information on the **/etc/security/ldap/ldap.cfg** file, see **/etc/security/ldap/ldap.cfg** in the *AIX 5L Version 5.2 Files Reference*.

LDAP Attribute Mapping File Format

These map files are used by the **/usr/lib/security/LDAP** module and the **secldapclntd** daemon for translation between AIX attribute names to LDAP attribute names. Each entry in a mapping file represents a translation for an attribute. A entry has four space separated fields:

```
AIX_Attribute_Name AIX_Attribute_Type LDAP_Attribute_Name LDAP_Value_Type
```

AIX_Attribute_Name	Specifies the AIX attribute name.
AIX_Attribute_Type	Specifies the AIX attribute type. Values are SEC_CHAR, SEC_INT, SEC_LIST, and SEC_BOOL.
LDAP_Attribute_Name	Specifies the LDAP attribute name.
LDAP_Value_Type	Specifies the LDAP value type. Values are **s** for single value and **m** for multi-value.

For more information on the LDAP attribute mapping file format, see **LDAP attribute mapping file format** in the *AIX 5L Version 5.2 Files Reference*.

Related Information

The **mksecldap**, **start-secldapclntd**, **stop-secldapclntd**, **restart-secldapclntd**, **ls-secldapclntd**, **sectoldif**, and **flush-secldapclntd** commands.

The **secldapclntd** daemon.

The **/etc/security/ldap/ldap.cfg** file.

The **LDAP attribute mapping file format**.

Chapter 5. PKCS #11

The PKCS #11 subsystem provides applications a method for accessing hardware devices (tokens) in a device-neutral manner. The content in this chapter conforms to Version 2.01 of the PKCS #11 standard.

This subsystem has been implemented using the following components:

- A slot manager daemon (**pkcsslotd**), which provides the subsystem with information regarding the state of available hardware devices. This daemon is started automatically during installation and when the system is rebooted.
- An API shared object (**/usr/lib/pkcs11/pkcs11_API.so**) is provided as a generic interface to the adapters for which PKCS #11 support has been implemented.
- An adapter-specific library, which provides the PKCS #11 support for the adapter. This tiered design allows the user to use new PKCS #11 devices when they come available without recompiling existing applications.

This chapter includes the following information:

IBM 4758 Model 2 Cryptographic Coprocessor

The IBM 4758 Model 2 Cryptographic Coprocessor provides a secure computing environment. Before attempting to configure the PKCS #11 subsystem, verify that the adapter has been properly configured with a supported microcode.

Verifying the IBM 4758 Model 2 Cryptographic Coprocessor for use with the PKCS #11 Subsystem

The PKCS #11 subsystem is designed to automatically detect adapters capable of supporting PKCS #11 calls during installation and at reboot. For this reason, any IBM 4758 Model 2 Cryptographic Coprocessor which is not properly configured will not be accessible from the PKCS #11 interface and calls sent to the adapter will fail. Complete the following to verify that your adapter is set up correctly:

1. Ensure that the software for the adapter is properly installed using the following command:

   ```
   lsdev -Cc adapter | grep crypt
   ```

   If the IBM 4758 Model 2 Cryptographic Coprocessor does not show in the resulting list, check that the card is seated properly and that the supporting software is correctly installed.
2. Determine that the proper firmware has been loaded onto the card using the **csufclu** utility:

   ```
   csufclu /tmp/l ST device_number_minor
   ```

   Verify that the Segment 3 Image has the PKCS #11 application loaded. If it is not loaded refer to the adapter specific documentation to obtain the latest microcode and installation instructions.

   Note: If this utility is not available, then the supporting software has not been installed.

PKCS #11 Subsystem Configuration

The PKCS #11 subsystem automatically detects devices supporting PKCS #11. However, in order for some applications to use these devices, some initial set up is necessary. These tasks include:

- “Initializing the User PIN”

These tasks can be performed through the API (by writing a PKCS #11 application) or by using the SMIT interface. The PKCS #11 SMIT options are accessed either through **Manage the PKCS11 subsystem** from the main SMIT menu, or by using the **smit pkcs11** fast path.

Initializing the Token

Each adapter or PKCS #11 token must be initialized before it can be used successfully. This initialization procedure involves setting a unique label to the token. This label allows applications to uniquely identify the token. Therefore, the labels should not be repeated. However; the API does not verify that labels are not re-used. This initialization can be done through a PKCS #11 application or by the system administrator using SMIT. If your token has a Security Officer PIN, the default value is set to 87654321. To ensure the security of the PKCS #11 subsystem, this value should be changed after initialization.

To initialize the token:

1. Enter the token management screen by typing `smit pkcs11`.
2. Select **Initialize a Token** .
3. Select a PKCS #11 adapter from the list of supported adapters.
4. Confirm your selection by pressing Enter.

   Note: This will erase all information on the token.
5. Enter the Security Officer PIN (SO PIN) and a unique token label.

If the correct PIN is entered, the adapter will be initialized or reinitialized after the command has finished running.

Setting the Security Officer PIN

If your token has an SO PIN, you can change the PIN from its default value, as follows:

1. Type `smit pkcs11`.
2. Select **Set the Security Officer PIN**.
3. Select the initialized adapter for which you want to set the SO PIN.
4. Enter the current SO PIN and a new PIN.
5. Verify the new PIN.

Initializing the User PIN

After the token has been initialized, it might be necessary to set the user PIN to allow applications to access token objects. Refer to your device specific documentation to determine if the device requires a user to log in before accessing objects.

To initialize the user PIN:

1. Enter the token management screen typing `smit pkcs11`.
2. Select **Initialize the User PIN**.
3. Select a PKCS #11 adapter from the list of supported adapters.
4. Enter the SO PIN and the User PIN.
5. Verify the User PIN.
6. Upon verification, the User PIN must be changed.

Resetting the User PIN

To reset the user PIN, you can either reinitialize the PIN using the SO PIN or set the user PIN by using the existing user PIN. To do this:

1. Enter the token management screen by typing `smit pkcs11`.
2. Select **Set the User PIN**.
3. Select the initialized adapter for which you want to set the user PIN.
4. Enter the current user PIN and a new PIN.
5. Verify the new user PIN.

Setting the PKCS #11 Function Control Vector

Your token might not support strong cryptographic operations without loading a function control vector. Please refer to your device specific documentation to determine if your token needs a function control vector and where to locate it.

If a function control vector is required you should have a key file. To load the function control vector:

1. Enter the token management screen by typing `smit pkcs11`.
2. Select **Set the function control vector**.
3. Select the PKCS #11 slot for the token.
4. Enter the path to the function control vector file.

PKCS #11 Usage

For an application to use the PKCS #11 subsystem, the subsystem's slot manager daemon must be running and the application must load in the API's shared object.

The slot manager is normally started at boot time by **inittab** calling the **/etc/rc.pkcs11** script. This script verifies the adapters in the system before starting the slot manager daemon. As a result, the slot manager daemon is not available before the user logs on to the system. After the daemon starts, the subsystem incorporates any changes to the number and types of supported adapters without intervention from the systems administrator.

The API can be loaded either by linking in the object at runtime or by using deferred symbol resolution. For example, an application can get the PKCS #11 function list in the following manner:

```
d CK_RV (*pf_init)();
void *d;
CK_FUNCTION_LIST *functs;

d = dlopen(e, RTLD_NOW);
if ( d == NULL ) {
   return FALSE;
}

pfoo = (CK_RV (*)())dlsym(d, "C_GetFunctionList");
if (pfoo == NULL) {
   return FALSE;
}

rc = pf_init(&functs);
```

Chapter 6. X.509 Certificate Authentication Service and Public Key Infrastructure

Certificate Authentication Service provides the AIX 5.2 operating system with the ability to authenticate users using X.509 Public Key Infrastructure (PKI) certificates and to associate certificates with processes as proof of a user's identity. It provides this capability through the Loadable Authentication Module Framework (LAMF), the same extensible AIX mechanism used to provide DCE, Kerberos, and other authentication mechanisms.

This section discusses the following topics:

Overview of Certificate Authentication Service

Every user account participating in PKI authentication has a unique PKI certificate. The certificate in conjunction with a password is used to authenticate the user during login. PKI certificates are based on public key/private key technology. This technology uses two asymmetric keys to encrypt and decrypt data. Data encrypted using one key can only be decrypted using the other key. A user keeps one key private, called the private key, storing it in a private keystore while publishing the other key, called the public key, in the form of a certificate. Certificates are commonly maintained on a Lightweight Directory Access Protocol (LDAP) server, either within an organization for intra-company usage or on the Internet for world-wide usage.

For a user named John to send a user named Kathy data that only she can decrypt, John would obtain the public key from Kathy's published certificate, encrypt the data using Kathy's public key, and send the data to her. Kathy would decrypt the data from John using her private key located in her private keystore.

This technology is also used for digital signatures. If Kathy wants to send data to John that is digitally signed by her, Kathy would use her private key to digitally sign the data and send the data and digital signature to John. John would obtain the public key from Kathy's published certificate and use the public key to verify the digital signature before using the data.

In both cases, Kathy's private key is maintained in a private keystore. The many types of private keystores include smart cards and files, but all keystore types protect private keys through the use of passwords or Personal Identification Numbers (PINs). They typically provide storage for multiple private keys along with certificates and other PKI objects. Users typically have their own keystores.

Certificate authentication service uses digital-signature technology to authenticate a user during login. Certificate authentication service locates the user's certificate and keystore based off the user's account name, obtains the certificate's matching private key from the user's keystore using the user's password, signs a data item with the user's private key, and checks the signature using the user's public key from the certificate. After the user authenticates, the system stores the user's certificate in protected memory, associating the certificate with every process created by the user. This in-memory association enables quick access to the user's certificate for any process owned by the user, as well as by the operating system's kernel.

Certificates

Understanding certificate authentication service requires a basic understanding of certificates, certificate formats, and certificate life-cycle management. Certificates are standardized objects that follow the X.509 standard, of which version 3 (X.509v3) is the latest version. Certificates are created, signed, and issued by a Certificate Authority (CA) which is most commonly a software application that accepts and processes certificate requests. Certificates are comprised of several certificate attributes. Some of the attributes are required, but many are optional. Certificate attributes commonly used and discussed in this document are:

- Certificate Version - The X.509 version number (that is, 1, 2, or 3).
- Serial Number - A certificate serial number that uniquely distinguishes the certificate from all other certificates issued by the same CA.
- Issuer Name - A name specifying the certificate's issuing CA.
- Validity Period - The activation and expiration date of the certificate.
- Public Key - The public key.
- Subject Distinguished Name - A name specifying the certificate's owner.
- Subject Alternate Name Email - The owner's e-mail address.
- Subject Alternate Name URI - The owner's Web site URI/URL.

Each certificate has a unique version number that indicates with which version of the X.509 standard it conforms. Each certificate has a serial number which uniquely distinguishes it from all other certificates issued by the same CA. The serial number is unique only to the issuing CA. The certificate's issuer name identifies the issuing CA.

Certificates are valid only between two specified dates: the ″Not Before″ date and the ″Not After″ date. Therefore, certificates may be created prior to their validity date and expire at some date in the future. It's common for certificates to have a life span of 3 months to 5 years.

The subject distinguished name specifies the certificate owner by using a specialized naming format known as a Distinguished Name (DN). A DN allows for the specification of the country, organization, city, state, owner name, and other attributes associated with the requesting entity (usually a person, but not limited to a person). The subject alternate name e-mail allows for the specification of the owner's email address and the subject alternate name URI allows for the specification of the owner's Web site URI/URL.

Certificate Authorities and Certificates

Certificate Authorities issue, store, and typically publish certificates. A common place to publish certificates is on an LDAP server, since LDAP allows for easy access to community oriented data.

CAs also handle the revocation of certificates and the management of certificate revocation lists (CRLs). Revoking a certificate is the act of publishing the fact that a specific certificate is no longer valid due to reasons other than the expiration of the certificate's validity period. Because copies of certificates can be maintained and used outside the control of the issuing CA, CAs publish a list of revoked certificates in a CRL so that outside entities may query the list. This places the responsibility on entities using a copied certificate to compare the copied certificate against the issuing CA's CRL. A CA may only revoke certificates that it creates or issues. It cannot revoke certificates issued by other CAs.

Administrative reasons for revoking a certificate include:

- Compromise of the certificate's private key.
- Certificate owner left the company.
- Compromise of the CA.

CAs also have their own identifying certificate. This allows CAs to identify each other in peer-to-peer communications among other uses (for example, chains of trust).

Many CAs support the Certificate Management Protocol (CMP) for requesting and revoking certificates. The protocol supports multiple methods to establish a secure connection between a client (also known as an End Entity) and the CA, though not all clients and CAs support all methods. One common method requires each certificate creation and revocation request to use a reference number and password recognized by the CA. Other data such as a special certificate recognized by the CA may also be required. Revocation requests may require the matching private key of the certificate being revoked.

Although CMP provides for certificate creation and revocation requests, it does not support CRL query requests. In fact, CRLs are often accessed through out-of-band methods. Since CRLs are often published on LDAP servers, software applications can obtain the CRL from an LDAP server and manually scan the CRL. Another emerging method is the Online Certification Status Protocol (OCSP), but not all CAs support OCSP.

CAs are typically owned and operated by government organizations or trusted private organizations that attempt to provide assurance that certificates issued by them correspond to the person who requested the issuance of the certificate. The phrase *issuing a certificate* means to create a certificate and is not the same as requesting a copy of a published certificate.

Certificate Storage Format

The most common format for storing individual certificates is in Abstract Syntax Notation version 1 (ASN.1) format using the Distinguished Encoding Rules (DER). This format is referred to as *DER format*.

Keystores

A keystore (sometimes called a *keyset*) contains a user's private keys matching the public keys of their certificates. A unique key label is assigned to every private key, usually by the user, for easy identification. Keystores are password-protected requiring a user to enter a password prior to accessing the keys or adding new keys. And typically, users have their own keystores. Keystores come in many different forms, for example: smart cards, LDAP-based, file-based, and so on. Not only do the forms vary, but so do the methods used to access them and the formats used to store the private key data. Certificate authentication service only supports file-based keystores.

Implementation of Certificate Authentication Service

Certificate authentication service functions as a client/server model. The server side contains a Certificate Authority (CA) for creating and maintaining X.509 version 3 certificates and certificate revocation lists (CRLs). (Typically, an organization uses one CA for the entire organization.) The client side contains the software (commands, libraries, load modules, and configuration files) required by every system participating in PKI authentication. The installation package for the server is **cas.server** and the installation package for the client is **cas.client**.

Creating PKI User Accounts

To create a PKI user account, use the AIX **mkuser** command. After it is created, each account has a certificate and a private keystore. (Existing accounts can be converted to PKI accounts too, but other steps are required.) The administrator supplies the keystore passwords to the new users, and new users can then log in to the system and change their keystore password.

User Authentication Data Flow

This section describes how a PKI user is authenticated. Users can have multiple certificates associated with their accounts. Each certificate has a unique, user defined tag value associated with it for easy identification, but only one certificate can be specified as the authentication certificate. Certificate authentication service uses a per-user attribute named **auth_cert** to specify which of the user's certificates is the user's authentication certificate. The value of the **auth_cert** attribute is the certificate's tag value.

The certificates, tags, matching keystore locations, matching key labels, and other related data are maintained under LDAP on a per-user basis. The combination of the user name and tag allows certificate

authentication service to locate the certificate under the LDAP server. For more information on the PKI LDAP layer, see “PKI LDAP Layer (Certificate Storage)” on page 82.

At login, users supply a user name and password. Using the user name, the system retrieves the user's authentication certificate tag from the user's **auth_cert** attribute. Combining the user name and tag, the system retrieves the user's certificate, keystore location, and matching key label from LDAP. It checks the validity period values found in the certificate to determine if the certificate has expired or has not reached its activation date. The system then retrieves the user's private key by using the keystore location, key label, and supplied password. After the private key is retrieved, the system verifies that the private key and certificate match using an internal signing process. If the two match, the user passes the PKI authentication step of the login procedure. (This does not imply that the user is logged in. Several other account checks are performed by the AIX system on a user account before allowing the user access to the system.)

For a certificate to be used as an authentication certificate, the certificate must be signed using a trusted signing key. The signature is stored under LDAP with the certificate for later reference. The implementation requires that a certificate have a signature before the tag can be assigned to **auth_cert**.

The authentication process does not compare a certificate against a CRL. This is due to performance reasons (CRLs take time to acquire and scan and may be temporarily unavailable), but also due to publishing delays of CRLs (CAs may delay an hour or more before publishing a revoked certificate through a CRL, making certificate revocation a poor substitute for disabling a user's account).

It is also worth noting that authentication does not require a CA. The majority of the work is performed locally by certificate authentication service, with the exception of retrieving data stored under LDAP.

Server Implementation

The server side of certificate authentication service implements a CA written in Java and contains a Registration Authority (RA) along with self-auditing features. It publishes certificates and CRLs that it creates to an LDAP server. The CA is configurable through a set of configuration files (Java property files). It contains an administrative application called **runpki** that provides sub-commands to start and stop the server among other functions and supports CMP for creating and revoking certificates. The CA requires Java 1.3.1, the IBM DB2 7.1 database, and the IBM Directory 4.1. Due to DB2 requirements, the CA must run under a user account other than the root user.

The server contains the following commands to help install and manage the **cas.server** component:

mksecpki
: This command is used during installation to configure the AIX PKI server components. As part of its tasks, the command creates a certificate authority user account for the certificate authority.

runpki
: This command allows the system administrator to start the server. If the JavaPKI daemons are running, they must first be stopped. The **runpki** command starts the daemon in the background by using the **lb** flags combination. If the daemons need to be started in interactive mode, the administrator can edit the **runpki** command and use the **l** flag instead of the **lb** flags.

  The **runpki** command must be run after performing an **su -** operation to the user account under which the certificate authority is running. The command is located in the **javapki** directory under the certificate authority user account's home directory. (The **mksecpki** command creates the certificate authority user account.)

  For example, if the certificate authority user account is **pkiinst**, then with root authority, type the following:

  1. `su - pkiinst`
  2. `cd javapki`
  3. `runpki`

Client Implementation

The client side of certificate authentication service implements the user authentication, user administration, and user certificate management functions of certificate authentication service. After it is installed and configured on a system, certificate authentication service integrates into the existing user authentication and administration functions (such as the **mkuser**, **chuser**, **passwd**, and **login** commands) through the use of the AIX Loadable Authentication Module Framework (LAMF). It also adds several commands, libraries, and configuration files to help manage user certificates and keystores.

Certificate authentication service can be used in conjunction with either the AIX LDAP database mechanism or the file-based database mechanism for storing standard AIX attributes. Certificate authentication service always uses LDAP to maintain user certificates, even when the file-based database mechanism is used. For limitations when using the file-based database, see “Planning for Certificate Authentication Service” on page 89.

The client side of certificate authentication service contains the most user oriented software of the two parts. For this reason, the following sections describe how certificate authentication service maintains and uses the data required for PKI authentication.

General Client Features

The following list describes some of the general features of certificate authentication service:

- Provides user authentication via PKI certificates
- Provides commands to manage user certificates and keystores
- Supports multiple certificates per user
- Supports multiple CA’s simultaneously
- Integrates into existing AIX administration commands and authentication (for example, **login**, **passwd**, **mkuser**)
- Generate certificates at user creation time or add certificates after user creation
- Works with either an LDAP user database or the standard AIX file-based user database
- Configurable key sizes and algorithms
- Associates certificates with Process Authentication Groups (PAGs).

General Client Architecture

The client architecture of certificate authentication service takes a layered approach and is divided into the following components:

Java Daemon: At the foundation of the client side is a Java-based daemon using the JCE security package. The daemon manages user keystores, creates key pairs, performs CMP communications, and provides all hashing and encryption functions. Because APIs of PKI service provider packages are not standardized for C applications, a wrapper layer API called the Service Management Layer (SML) provides a normalized API to application programs and daemons.

Service Management Layer: The SML service for the Java daemon is named **/usr/lib/security/pki/JSML.sml**. SML creates certificates, and creates and manages keystores, but it doesn't manage certificate storage. Certificate storage is managed by the PKI LDAP Layer.

Private Key Storage Through SML: The Java daemon uses PKCS#12 formatted keystore files for storing user keys. The keystores are protected by a single password used to encrypt all the keys in the keystore. The location of a keystore is specified as a URI. By default, certificate authentication service maintains keystore files in the **/var/pki/security/keys** directory.

Keystores are typically limited in size, including file keystores. The SML Layer provides the API for managing keystores.

Certificate authentication service supports only file keystores. It does not support smart card or LDAP keystores. You can support roaming users by placing the file keystores on a shared file system under the same mount point on all systems.

PKI LDAP Layer (Certificate Storage): Certificate authentication service stores certificates and other certificate related information on a per user basis in LDAP through the PKI LDAP Layer. Certificate authentication service maintains the certificate associations on a per user basis on an LDAP server. A user account can have multiple certificates associated with it. Each association has a unique, user-specified tag for easy identification and lookup. Certificate authentication service uses the combination of the user's name and the tag to locate a user's certificate association in LDAP.

For performance versus disk space trade-offs, certificate authentication service can save either the entire certificate under LDAP or just a URI reference to the certificate. If a URI reference is used instead of a certificate, certificate authentication service queries the reference to obtain the actual certificate. References are most commonly used in conjunction with a CA which publishes its certificates on an LDAP sever. The types of URI references currently supported by certificate authentication service are LDAP references. Certificate authentication service stores certificates in DER format and expects URI references to refer to DER formatted certificates.

Certificate authentication service also stores the type and location of each certificate's matching keystore and key label in the same record as the certificate association on the LDAP server. This allows users to have more than one keystore and allows certificate authentication service to quickly find a certificate's matching private key. To support roaming users, a user's keystore must reside in the same location on all systems.

Certificate authentication service maintains the **auth_cert** attribute in LDAP on a per-user basis. This attribute specifies the tag of the certificate used for authentication.

All LDAP information is readable by ordinary users, except for the **auth_cert** attribute which is restricted to the LDAP **ldappkiadmin** account. Since the root user has access to the LDAP **ldappkiadmin** password through the **acct.cfg** file, applications running with the effective UID of root can access the **auth_cert** attribute. (This applies to the accessibility of the URI reference value, not to the data referenced by the URI reference value. Typically, the data referenced by the URI reference value is public.) The API for managing the certificate storage is contained in the **libpki.a** library.

The libpki.a Library: In addition to serving as the home of the SML APIs and the PKI LDAP Layer APIs, the **libpki.a** library houses several subroutines. The library includes APIs that do the following:

- Manage the new configuration files
- Access certificate specific attributes
- Combine multiple lower layer functions into higher level functions
- Are expected to be common among SML services

Note: The APIs are not published.

Loadable Authentication Module Framework Layer: On top of the SML API and PKI LDAP API resides the Loadable Authentication Module Framework (LAMF) layer. LAMF supplies AIX authentication and user administration applications with common authentication and user administration APIs regardless of the underlying mechanism (for example, Kerberos, LDAP, DCE, files). LAMF uses the SML API and the PKI LDAP API as building blocks in implementing PKI authentication.

It does this through the use of load modules that map LAMF's API to different authentication/database technologies. Commands like **login**, **telnet**, **passwd**, **mkuser**, and others use the LAMF API to implement their functions; hence, these commands automatically support new authentication and database technologies when new load modules for these technologies are added to the system.

Certificate authentication service adds a new LAMF load module to the system named **/usr/lib/security/PKI**. The module must be added by the system administrator to the **/usr/lib/security/methods.cfg** file before using PKI for authentication. The module must also be paired with a database type (for example, LDAP) in the **methods.cfg** file before it can be used for authentication. An example of the **methods.cfg** file containing the LAMF module and database definition can be found in "The methods.cfg File" on page 101.

Once the definitions are added to **methods.cfg**, the administrator can set the **registry** and **SYSTEM** user attributes (defined in the **/etc/security/user** file) to the new stanza value or values for PKI authentication.

Client Commands: Above all the API layers (LAMF, PKI LDAP, and SML) reside the commands. Besides the standard AIX authentication and user administration commands supporting certificate authentication service (through LAMF), several certificate authentication service specific commands exist. These commands help the user manage certificates and keystores. Below is a list of the commands along with a brief description.

certadd
: Adds a certificate to the user's account in LDAP and checks if the certificate is revoked.

certcreate
: Creates a certificate.

certdelete
: Deletes a certificate from the user's account (i.e., from LDAP).

certget
: Retrieves a certificate from the user's account (i.e., from LDAP).

certlink
: Adds a link to a certificate that exists in a remote repository to the user's account in LDAP and checks if the certificate is revoked.

certlist
: Lists the certificates associated with the user's account contained in LDAP.

certrevoke
: Revokes a certificate.

certverify
: Verifies the private key matches the certificate and performs trusted signing.

keyadd
: Adds a keystore object to a keystore.

keydelete
: Deletes a keystore object from a keystore.

keylist
: Lists the objects in a keystore.

keypasswd
Changes the password on a keystore.

For more information about these commands. see the *AIX 5L Version 5.2 Commands Reference*.

Process Authentication Group Commands: The Process Authentication Group (PAG) commands are new to AIX. PAGs are data items that associate user-authentication data with processes. For certificate authentication service, if the PAG mechanism is enabled, the user's authentication certificate is associated with the user's login shell. As the shell creates child processes, the PAG propagates to each child.

The PAG mechanism requires the **/usr/sbin/certdaemon** daemon to be enabled in order to provide this functionality. By default, the mechanism is not enabled. Certificate authentication service does not require the PAG mechanism to be enabled, but works with the mechanism if it is enabled.

To enable the **certdaemon** daemon, add the following line to the **/etc/inittab** file:

```
certdaemon:2:wait:/usr/sbin/certdaemon
```

A list of PAG commands along with brief descriptions follows:

paginit
Authenticates a user and creates a PAG association.

pagdel
Lists authentication information associated with the current process.

paglist
Removes existing PAG associations within the current process' credentials.

For more information about these commands, see the *AIX 5L Version 5.2 Commands Reference*.

User Administration Commands: Similar to user-authentication, certificate authentication service integrates with the AIX user-administration functions through the AIX LAMF. Commands like **chuser**, **lsuser**, **mkuser**, and **passwd** use the LAMF API to implement their functions. Therefore, these commands automatically support new authentication and database technologies when new load modules for these technologies are added to the system.

The subsections below provide a more in-depth look at how PKI authentication affects the user administration commands.

The following commands are affected by the PKI authentication process:

chuser
This command allows the administrator to modify the **auth_cert** user attribute. This attribute specifies the tag value of the certificate used for authentication. The certificate must be signed by the trusted signing key in order to be used as the authentication certificate. (Certificate attributes, certificate storage attributes, and keystore attributes are not available through this command.)

lsuser This command lists the value of the user's **auth_cert** attribute, as well as, the certificate attributes listed below. The **auth_cert** attribute specifies the tag value of the certificate used for authentication. (Other certificate attributes, certificate storage attributes, and keystore attributes are not available through this command.)

The certificate attributes listed by the **lsuser** command are as follows:

subject-DN
The user's subject distinguished name.

subject-alt-name
The user's subject alternate name email.

valid-after
The date the user's certificate becomes valid.

valid-until
The date the user's certificate becomes invalid.

issuer The distinguished name of the issuer.

mkuser
This command provides an administrator the option of generating a certificate at user creation time. An administrator can use the **mkuser** command to generate a certificate during user creation for users who don't already have an authentication certificate. Optionally, if a user already has an authentication certificate, but no user account, the administrator can create the account without generating a certificate and add the certificate (and keystore) later. The default value for this option is specified in the **/usr/lib/security/pki/policy.cfg** file in the newuser stanza by the **cert** attribute.

Many default values are required when automatically generating an authentication certificate for a user using the **mkuser** command. Many of these values are specified in the **newuser** stanza of the **/usr/lib/security/pki/policy.cfg** file. The **newuser** stanza provides administrative control over these default values. Some of the default values are as follows:

- CA
- Value for the **auth_cert** attribute
- Location for the keystore
- Password for the keystore
- Private key label
- Domain name for the subject alternate name e-mail field

A behavioral difference between creating a PKI user account and a non-PKI user account is that creating a PKI user account requires a password to encrypt the private key if the **mkuser** command generates an authentication certificate for the account. Since the **mkuser** command is a non-interactive command, the command obtains the password from the **policy.cfg** file and sets the keystore password (the private key password) to this value; therefore, the account is immediately accessible after creation. When creating a non-PKI user account, the **mkuser** command sets the password to an invalid value, preventing accessibility.

passwd
This command modifies the user's keystore password when used on a PKI user account. It enforces the password restriction rules found in the **/etc/security/user** file, it enforces the flags attribute found in the **/etc/security/passwd** file, and it enforces any rules required by the PKI service provider.

Because file-based keystores encrypt their private keys using the user's password, the **root** user cannot reset a file-based keystore password without knowing the keystore's current password. If a user forgets their keystore password, the root user will not be able to reset the password unless root knows the keystore's password. If the password is unknown, a new keystore and new certificates may have to be issued to the user.

Configuration Files: Certificate authentication service uses configuration files for configuring the client-side: **acct.cfg**, **ca.cfg**, and **policy.cfg**. The SMIT interface provides support for these configuration files. The following sections provide information about the configuration files.

The acct.cfg File: The **acct.cfg** file consists of CA stanzas and LDAP stanzas. The CA stanzas contain private CA information not suitable for the publicly readable **ca.cfg** file, such as CMP reference numbers and passwords. The LDAP stanzas contain private LDAP information not suitable for public access, such as PKI LDAP administrative names and passwords.

For every CA stanza in the **ca.cfg** file, the **acct.cfg** file should contain an equivalently named CA stanza, and all CA stanzas must be uniquely named. The LDAP stanzas are all named **ldap**, and for this reason, a

CA stanza cannot be named **ldap**. Also, no stanza can be named **default**. An LDAP stanza must exist, and at least one CA stanza, named **local**, must also exist.

CA stanzas contain the following attributes:

capasswd
Specifies the CA's CMP password. The length of the password is specified by the CA.

carefnum
Specifies the CA's CMP reference number.

keylabel
Specifies the label of the private key in the trusted keystore used to sign certificate requests.

keypasswd
Specifies the password for the trusted keystore.

rvpasswd
Specifies the revocation password used for CMP. The length of the password is specified by the CA.

rvrefnum
Specifies the revocation reference number used for CMP.

The LDAP stanza contains the following attributes:

ldappkiadmin
Specifies the account name of the LDAP server listed in **ldapservers**.

ldappkiadmpwd
Specifies the password for the LDAP server's account.

ldapservers
Specifies the LDAP server name.

ldapsuffix
Specifies the DN attributes added to a user's certificate DN by the **mkuser** command.

The following is an example **acct.cfg** file:

```
local:
 carefnum = 12345678
 capasswd = password1234
 rvrefnum = 9478371
 rvpasswd = password4321
 keylabel = "Trusted Key"
 keypasswd = joshua

ldap:
 ldappkiadmin = "cn=admin"
 ldappkiadmpwd = secret
 ldapservers = "ldap.server.austin.ibm.com"
 ldapsuffix = "ou=aix,cn=us"
```

For more information, see the *AIX 5L Version 5.2 Files Reference*.

The ca.cfg File: The **ca.cfg** file consists of CA stanzas. The CA stanzas contain public CA information used by certificate authentication service for generating certificate requests and certificate revocation requests.

For every CA stanza in the **ca.cfg** file, the **acct.cfg** file should contain an equivalently named CA stanza. Each CA stanza name in the **ca.cfg** file must be unique. At least one stanza named **local** must exist. No stanzas should be named **ldap** or **default**.

CA stanzas contain the following attributes:

algorithm
Specifies the public key algorithm (for example, RSA).

crl Specifies the CA's CRL URI.

dn Specifies the base DN used when creating certificates.

keysize
Specifies the minimum key size in bits.

program
Specifies the PKI service module file name.

retries
Specifies the number of retry attempts when contacting the CA.

server Specifies the CA's URI.

signinghash
Specifies the hash algorithm used to sign certificates (for example, MD5).

trustedkey
Specifies the trusted keystore containing the trusted signing key used for signing authentication certificates.

url Specifies the default value for the subject alternate name URI.

The default CA stanza is named local. The following is an example **ca.cfg** file:

```
local:
 program = /usr/lib/security/pki/JSML.sml
 trustedkey = file:/usr/lib/security/pki/trusted.p15
 server = "cmp://9.53.230.186:1077"
 crl = "ldap://dracula.austin.ibm.com/o=aix,c=us"
 dn = "o=aix,c=us"
 url = "http://www.ibm.com/"
 algorithm = RSA
 keysize = 512
 retries = 5
 signinghash = MD5
```

For more information, see the *AIX 5L Version 5.2 Files Reference*.

The policy.cfg File: The **policy.cfg** file consists of four stanzas: **newuser**, **storage**, **crl**, and **comm**. These stanzas modify the behavior of some system administration commands. The **mkuser** command uses the **newuser** stanza. The **certlink** command uses the **storage** stanza. The **certadd** and **certlink** commands use the **comm** and **crl** stanzas.

The **newuser** stanza contains the following attributes:

ca Specifies the CA used by the **mkuser** command when generating a certificate.

cert Specifies whether the **mkuser** command generates a certificate (new) or not (get) by default.

domain
Specifies the domain part of the certificate's subject alternate name e-mail value used by the **mkuser** command when generating a certificate.

keysize
Specifies the minimum encryption key size in bits used by the **mkuser** command when generating a certificate.

keystore
Specifies the keystore URI used by the **mkuser** command when generating a certificate.

keyusage
Specifies the certificate's key usage value used by the **mkuser** command when generating a certificate.

label Specifies the private key label used by the **mkuser** command when generating a certificate.

passwd
Specifies the keystore's password used by the **mkuser** command when generating a certificate.

subalturi
Specifies the certificate's subject alternate name URI value used by the **mkuser** command when generating a certificate.

tag Specifies the **auth_cert** tag value used by the **mkuser** command when creating a user when cert = new.

validity
Specifies the certificate's validity period value used by the **mkuser** command when generating a certificate.

version
Specifies the version number of the certificate to be created. The value 3 is the only supported value.

The **storage** stanza contains the following attributes:

replicate
Specifies whether the **certlink** command saves a copy of the certificate (**yes**) or just the link (**no**).

The **crl** stanza contains the **check** attribute, which specifies whether the **certadd** and **certlink** commands should check the CRL (**yes**) or not (**no**).

The **comm** stanza contains the **timeout** attribute which specifies the timeout period in seconds used by **certadd** and **certlink** when requesting certificate information using HTTP (for example, retrieving CRLs).

The following is an example of the **policy.cfg** file:

```
newuser:
 cert = new
 ca = local
 passwd = pki
 version = "3"
 keysize = 512
 keystore = "file:/var/pki/security/keys"
 validity = 86400

storage:
 replicate = no

crl:
 check = yes

comm:
 timeout = 10
```

For more information, see the *AIX 5L Version 5.2 Files Reference*.

Audit Log Events: The certificate authentication service client generates the following audit-log events:

- CERT_Create
- CERT_Add
- CERT_Link
- CERT_Delete

- CERT_Get
- CERT_List
- CERT_Revoke
- CERT_Verify
- KEY_Password
- KEY_List
- KEY_Add
- KEY_Delete

Trace Events: The certificate authentication service client generates several new trace events in the 3B7 and 3B8 range.

Planning for Certificate Authentication Service

Certificate authentication service is available beginning with AIX 5.2. The minimum software requirements for certificate authentication service are a DB2 server, an IBM Directory server, and a certificate authentication service server. All can be installed on one system or on a combination of systems. Each enterprise must determine the best choice for their environment.

This section provides information on planning for certificate authentication service, as follows:

Certificate Considerations

Certificate authentication service supports X.509 version 3 certificates. It also supports several version 3 certificate attributes, but not all certificate attributes. For a list of supported certificate attributes, see the **certcreate** command and the **ca.cfg** file. Certificate authentication service contains limited support of the Teletex character set. Specifically, only 7-bit (ASCII subset of) Teletex is supported by certificate authentication service.

Keystore Considerations

Certificate authentication service supports keystore files. Smart cards, LDAP keystores, and other types of keystores are not supported.

By default, user keystores are kept in the local file system under the **/var/pki/security/keys** directory. Because the keystores are local to the system, they cannot be accessed by other systems; thus, user authentication will be restricted to the system containing the user's keystore. To allow for roaming users, either copy the user's keystore to the identical location with the same keystore name on other systems or place the keystores on a distributed file system.

Note: Care must be taken to ensure that access permission to the user's keystore remains unchanged. (In AIX, every certificate in LDAP contains the path name to the private keystore containing the certificate's private key. The keystore must exist at the path name specified in LDAP in order to be used for authentication.)

User Registry Considerations

Certificate authentication service supports an LDAP user-registry. LDAP is also the recommended user registry type to use with certificate authentication service.

Certificate authentication service also supports a file-based user registry. Certain restrictions must be enforced by the administrator for file-based PKI to work correctly. Specifically, identically named user accounts on different systems participating in PKI authentication must refer to the same account.

For example, user *Bob* on *system A* and user *Bob* on *system B* must refer to the same user *Bob*. This is because certificate authentication service uses LDAP to store certificate information on a per user basis. The user name is used as the indexing key to access this information. Because file-based registries are local to each system and LDAP is global to all systems, the user names on all systems participating in PKI authentication must map to unique user names in the LDAP namespace. If user *Bob* on *system A* is different from user *Bob* on *system B*, either only one of the *Bob*'s can participate in PKI authentication or each *Bob* account must use a different LDAP namespace/server.

Configuration Considerations

For configuration simplicity, consider maintaining the three configuration files (**acct.cfg**, **ca.cfg**, and **policy.cfg**) on a distributed file system using symbolic links to avoid having to modify configuration files on every system. Maintain proper access-control settings on these files. This situation may increase your security vulnerability because the information in these files will be transferred across your network.

Security Considerations

The acct.cfg File

The **acct.cfg** file contains sensitive CA reference numbers and passwords (see the **carefnum**, **capasswd**, **rvrefnum**, and **rvpasswd** attribute descriptions for **acct.cfg**). These values are used solely for CMP communications with the CA when creating a certificate and revoking a certificate, respectively. If compromised, the compromiser may be able to create certificates at will, and revoke anyone's certificate at will.

To limit the exposure, consider restricting certificate creation or revocation to a small number of systems. The **carefnum** and **capasswd** attribute values are required only on systems where certificates are created (either through the **certcreate** or **mkuser** commands). This may imply limiting user account creation to the same set of systems.

Note: The **mkuser** command can be configured to automatically create a certificate during user creation or it can create an account without a certificate, whereby the administrator must create and add the certificate at a later time.

Similarly, the **rvrefnum** and **rvpasswd** attribute values are required only on systems where certificates are to be revoked (through the **certrevoke** command).

The **acct.cfg** file also contains sensitive trusted signing key information (see the **keylabel** and **keypasswd** attribute descriptions for the **acct.cfg** file). These values are used solely for special certificate verification operations. If compromised, the compromiser may be able to forge verified certificates.

To limit the exposure, consider restricting certificate verification to a small number of systems. The **keylabel** and **keypasswd** attribute values of the **acct.cfg** file and the **trustedkey** attribute value of the **ca.cfg** file are required only on systems where certificate verification is required. Specifically, on systems where the **mkuser** (with automatic certificate creation enabled) and **certverify** commands are required.

Active New Accounts

When creating a PKI user account, if the **cert** attribute of the **newuser** stanza in the **policy.cfg** file is set to **new**, the **mkuser** command creates an active PKI account complete with a working certificate and password. The password on the account is specified by the **passwd** attribute in the **newuser** stanza.

Because keystores require a password in order to store private keys. This differs from other types of user account creations where the administrator must first create the account, then set the password before the account is activated.

The root User and Keystore Passwords

Unlike other account types where the **root** user can change an account's password without knowing the account's password, PKI accounts do not allow this. This is because account passwords are used to encrypt keystores and keystores cannot be decrypted without knowing the password. When users forget their passwords, new certificates must be issued and new keystores created.

Other Certificate Authentication Service Considerations

Other considerations when planning for the certificate authentication service include the following:

- Certificate authentication service contains its own certificate authority (CA). Other CA implementations are not supported by certificate authentication service.
- The larger the key size, the more time required to generate key pairs and to encrypt data. Hardware based encryption is not supported.
- Certificate authentication service uses the IBM Directory for LDAP. Other LDAP implementations are not supported by certificate authentication service.
- Certificate authentication service uses DB2 for database support. Other database implementations are not supported by certificate authentication service.
- Certificate authentication service requires all commands, libraries, and daemons run in a Unicode environment.

Packaging of Certificate Authentication Service

Table 9. Packaging of Certificate Authentication Service

Package Name	Fileset	Contents	Dependencies	Installation
cas.server	cas.server.rte	Certificate Authority (CA)	• AIX 5.2 • Java131 (ships with AIX base media) • Java131 Security Extensions (ships with Expansion Pack) • IBM Directory Server (LDAP) • DB2 7.1	Manual
cas.client	cas.client.rte	• Cert commands • PKI Auth Load Module • libpki.a • SML Module • Config Files • Java Daemon	• AIX 5.2 • Java131 (ships with AIX base media) • Java131 Security Extensions (ships with Expansion Pack) • IBM Directory Client (LDAP) • PAG (assumed)	Manual
cas.msg	cas.msg.[lang].client	Message catalogs	cas.client	Manual
bos	bos.security.rte	PAG commands and daemon	n/a	Installed with kernel

The **cas.server** package contains the CA and installs in the **/usr/cas/server** and **/usr/cas/client** directories. An organization typically uses only one CA, and therefore, this package is installed manually.

This package prerequisites the IBM Directory server side, **db2_07_01.client**, **Java131.rte**, and **Java131.ext.security**. The **Java131.rte package** is installed by default when the AIX 5.2 operating system is installed, but the other packages are manually installed.

In order for the **db2_07_01.client** package to work, the **db2_07_01.server** package must be installed on a system that is on the network.

The **cas.client** package contains the files required for every client system supporting certificate authentication service. Without this package, a system cannot participate in AIX PKI authentication.

Installing and Configuring Certificate Authentication Service

Installation of Certificate Authentication Services consists of performing the following procedures:

Install and Configure the LDAP Server

The following scenarios are possible when installing and configuring LDAP for PKI user certificate data.

1. If the LDAP Server software is not installed, perform the following procedures:
   a. "LDAP Server Installation"
   b. "LDAP Server Configuration" on page 93
   c. "Configuring the LDAP Server for PKI" on page 93
2. If the LDAP Server software is installed and configured, but not configured for PKI, perform "Configuring the LDAP Server for PKI" on page 93.

LDAP Server Installation

Detailed instructions for installing the IBM Directory Server software can be found in the product documentation contained in the **ldap.html.en_US.config** fileset. After installing the **ldap.html.en_US.config** fileset, the documentation can be viewed using a Web browser at the following URL: **file:/usr/ldap/web/C/getting_started.htm**.

The LDAP Server installation procedure is as follows:

1. Login as the **root** user.
2. Place volume 1 of the AIX Base Operating System CDs in the CD-ROM drive.
3. Type **smitty install_latest** at the command line and press Enter
4. Select **Install Software**.
5. Select the input device or software directory containing the IBM Directory Server software and press Enter.
6. Use the **F4** key to list the install packages in the **Software to Install** field.
7. Select the **ldap.server** package and press Enter.
8. Verify that the **AUTOMATICALLY install requisite software** option is set to **YES**, and press Enter. This will install the LDAP server and client filesets and the DB2 backend database filesets.

   The filesets installed include the following:
   - **ldap.client.adt** (Directory Client SDK)
   - **ldap.client.dmt** (Directory Client DMT)
   - **ldap.client.java** (Directory Client Java)
   - **ldap.client.rte** (Directory Client Run-time Environment)

- **ldap.server.rte** (Directory Server Run-time Environment)
- **ldap.server.admin** (Directory Server)
- **ldap.server.cfg** (Directory Server Config)
- **ldap.server.com** (Directory Server Framework)
- **db2_07_01.*** (DB2 Run-time Environment and associated filesets)

The DB2 package, **db2_07_01.jdbc**, must also be installed. The DB2 package, **db2_07_01.jdbc**, is located on the Expansion Pack CD. Use the installation procedure listed above to install the **db2_07_01.jdbc** package.

LDAP Server Configuration

After the LDAP and DB2 filesets have been installed, the LDAP server must be configured. Even though the configuration can be done through the command line and file editing, for ease of administration and configuration, the LDAP web administrator is recommended. This tool requires a web server.

The Apache web server application is located on the AIX Toolbox for LINUX Applications CD. Use either the SMIT interface or the **geninstall** command to install the Apache web server. Other web servers can also be used, see the LDAP documentation for details.

Detailed instructions for configuring LDAP can be found in the product HTML documentation. Below is a concise description of the configuration steps:

1. Use **ldapcfg** to set the admin DN and password for the LDAP database. The administrator is the **root** user of the LDAP database. To configure an administrator DN of **cn=admin** with a password of **secret**, type the following:

   ```
   # ldapcfg -u cn=admin -p secret
   ```

   The DN and password will be required later when configuring each client. Specifically, the DN and password will be used as the **ldappkiadmin** and **ldappkiadmpwd** attributes of an **ldap** stanza in the **acct.cfg** file.
2. Configure the web administrator tool using the location of the web server configuration file, as follows:

   ```
   # ldapcfg -s apache -f /etc/apache/httpd.conf
   ```
3. Restart the web server. For the Apache server, use the command:

   ```
   # /usr/local/bin/apachectl restart
   ```
4. Access the web administrator using the URL **http:// hostname/ldap**. Then login using the LDAP administrator DN and password configured in step 2.
5. Using the web administrator tool, follow the directions to configure the DB2 database backend and restart the LDAP server.

Configuring the LDAP Server for PKI

Certificate authentication service requires two separate LDAP directory information trees. One tree is used by the CA for publishing certificates and CRLs. The other tree is used by each client for storing and retrieving per-user PKI data. The following steps configure the LDAP directory information tree used for storing and retrieving per-user PKI data.

1. **Adding the LDAP Configuration Suffix Entry**. The default suffix for the PKI data is **cn=aixdata**. This places the PKI certificate data below the default suffix for all AIX data. The default data root for the PKI data is **ou=pkidata,cn=aixdata**. All PKI data is placed under this location.

   PKI Data Suffix

   cn=aixdata
   : Common suffix for all AIX data. May already exist if LDAP server is being used for other AIX data.

   The suffix configuration entry can be added through the web administrator tool, or by directly editing the LDAP server configuration file.

To add the suffix configuration entry using the Web administrator, do the following:

a. Select **Settings** from the left side menu.
b. Select **Suffixes**.
c. Enter the necessary suffix for the PKI data, and then click the **Update** button.
d. Restart the LDAP server, after the suffix is successfully added.

To add the suffix configuration entry by editing the LDAP server configuration file, do the following:

a. In the **/usr/ldap/etc/slapd32.conf** file, locate the line containing

   ```
   ibm-slapdSuffix: cn=localhost
   ```

   This is the default system suffix.
b. Add the necessary `ibm-slapdSuffix` entry for the PKI data. For example, you can add a suffix entry similar to the following:

   ```
   ibm-slapdSuffix: cn=aixdata
   ```
c. Save the configuration file changes.
d. Restart the LDAP server.

2. **Adding the PKI Data Suffix, Root, and ACL Database Entries**. The Data Root is the point in the LDAP directory structure under which all the PKI data resides. The ACL is the Access Control List for the Data Root that sets the access rules for all the PKI data. The **pkiconfig.ldif** file is supplied to add the suffix, root, and ACL entries to the database. First, add the suffix and root database entries and the PKI data administrator password. The first part of the file adds the default suffix entries to the database and sets the password as follows:

```
dn: cn=aixdata
objectclass: top
objectclass: container
cn: aixdata

dn: ou=pkidata,cn=aixdata
objectclass: organizationalUnit
ou: cert
userPassword: <<password>>
```

Edit the **pkiconfig.ldif** file and replace the <<password>> character string after the **userPassword** attribute with your password for the PKI data administrator.

The DN and **userPassword** values will be required later when configuring each client. Specifically, the DN (`ou=pkidata,cn=aixdata`) and value for *password* will be used as the **ldappkiadmin** and **ldappkiadmpwd** attributes of an **ldap** stanza in the **acct.cfg** file.

The second part of the file changes the ownership and adds the ACL for the PKI data as follows:

```
dn: ou=pkidata,cn=aixdata
changetype: modify
add: entryOwner
entryOwner: access-id:ou=pkidata,cn=aixdata
ownerPropagate: true

dn: ou=pkidata,cn=aixdata
changetype: modify
add: aclEntry
aclEntry: group:cn=anybody:normal:grant:rsc:normal:deny:w
aclEntry: group:cn=anybody:sensitive:grant:rsc:sensitive:deny:w
aclEntry: group:cn=anybody:critical:grant:rsc:critical:deny:w
aclEntry: group:cn=anybody:object:deny:ad aclPropagate: true
```

Note: DO NOT make any changes to the ACL settings. Doing so may jeopardize the integrity of your PKI implementation.

The **pkiconfig.ldif** file can be edited to use a suffix other than the default, however this is recommended only for experienced LDAP administrators. The **ldif** file can then be applied to the database using the **ldapadd** command below. Replace the values for the **-D** and **-w** options with your local LDAP administrator DN and password, as follows:

```
# ldapadd -c -D cn=admin -w secret -f pkiconfig.ldif
```

3. **Restart the LDAP Server**. Restart the LDAP server using the web administrator tool, or by killing and restarting the **slapd** process.

Install and Configure Certificate Authentication Service Server

To install and configure the certificate authentication service, do the following:

1. Install the Java security filesets (**Java131.ext.security.***) from the Expansion Pack CD. The required packages are as follows:
   - **Java131.ext.security.cmp-us** (Java Certificate Management)
   - **Java131.ext.security.jce-us** (Java Cryptography Extension)
   - **Java131.ext.security.jsse-us** (Java Secure Socket Extension)
   - **Java131.ext.security.pkcs-us** (Java Public Key Cryptography)
2. Move the **ibmjcaprovider.jar** file from **/usr/java131/jre/lib/ext** to another directory. This file conflicts with the Java security filesets and must be moved for correct functioning of the certificate authentication service.
3. Install the certificate authentication service server fileset (**cas.server.rte**) from the Expansion Pack CD.

Configure LDAP For Certificate Authentication Service Server

Configure certificate authentication service server to work with LDAP, by performing the following steps:

1. If not already installed, then install the IBM Directory client package on the system supporting the **cas.server** package.
2. If not already configured, then configure the IBM Directory client, as follows:

   ```
   # ldapcfg -l /home/ldapdb2 -u "cn=admin" -p secret -s apache \
             -f /usr/local/apache/conf/httpd.conf
   ```

   It is assumed that the Web Server is the Apache Web Server in the above configuration command.
3. Add the following suffix to the **slapd.conf** file, as follows:

   ```
   ibm-slapdSuffix: o=aix,c=us
   ```

   You can specify a different distinguished name instead of `o=aix,c=us`.
4. Run the **slapd** command, as follows:

   ```
   # /usr/bin/slapd -f /etc/slapd32.conf
   ```
5. Add the object classes, as follows:

   ```
   # ldapmodify -D cn=admin -w secret -f setup.ldif
   ```

   where **setup.ldif** contains the following:

   ```
   dn: cn=schema
   changetype: modify
   add: objectClasses
   objectClasses: ( 2.5.6.21 NAME 'pkiuser' DESC 'auxiliary class for non-CA certificate owners'
     SUP top AUXILIARY MAY userCertificate )

   dn: cn=schema
   changetype: modify
   add: objectClasses
   objectClasses: ( 2.5.6.22 NAME 'pkiCA' DESC 'class for Cartification Authorities' SUP top
     AUXILIARY MAY ( authorityRevocationList $ caCertificate $ certificateRevocationList $
   ```

```
  crossCertificatePair ) )

dn:cn=schema
changetype: modify
replace: attributetypes
attributetypes: ( 2.5.4.39 NAME ( 'certificateRevocationList'
  'certificateRevocationList;binary' ) DESC '  ' SYNTAX 1.3.6.1.4.1.1466.115.121.1.5
  SINGLE-VALUE )

replace:ibmattributetypes
ibmattributetypes:( 2.5.4.39 DBNAME ( 'certRevocationLst' 'certRevocationLst' )
  ACCESS-CLASS NORMAL)
```

6. Add the entries:

```
# ldapadd -D cn=admin -w secret -f addentries.ldif
```

where **addentries.ldif** contains the following:

```
dn: o=aix,c=us
changetype: add
objectclass: organization
objectclass: top
objectclass: pkiCA
o: aix
```

Note: Sample **addentries.ldif** and **setup.ldif** files are provided in the **cas.server** package.

7. Stop and start the **slapd** daemon.

Create the Certificate Authority

Create the certificate authority as follows:

1. Create a reference file. The reference file contains one or more certificate creation reference number and password pairs. A pair represents the authentication information accepted by the certificate authentication service server when a certificate authentication service client attempts to authenticate to the server during the creation of a certificate (typically using the CMP protocol). The format of the file is a reference number followed by a password, both on separate lines. For example:

```
12345678
password1234
87654321
password4321
```

where `12345678` and `87654321` are reference numbers, and `password1234` and `password4321` are their respective passwords. Blank lines are not allowed. Space characters should not precede or follow reference numbers or passwords. At least one reference number and password must exist in the file. An example file can be found in **/usr/cas/server/iafile**. You will need to reference these values each time you set up a client.

2. Configure the CA using the **mksecpki** command as follows:

```
# mksecpki -u pkiuser -f /usr/cas/server/iafile -p 1077 -H ldap.cert.mydomain.com \
           -D cn=admin -w secret -i o=aix,c=us
```

Information on the **mksecpki** flags follows:

-u Specifies a user account name where the certificate authentication service server will be installed.

-f Specifies the reference file created in the previous step.

-p Specifies a port number for the LDAP server.

-H Specifies the LDAP server host name or IP address.

-D Specifies the LDAP administrator's common name.

-w Specifies the LDAP administration password.

-i Specifies the LDAP branch where the user certificate data will reside.

The **mksecpki** command automatically generates the trusted signing key with a key label of **TrustedKey**, the password of the CA user account, and places it in the **/usr/lib/security/pki/trusted.pkcs12** keystore file. It's not necessary to perform the steps in "Create the Trusted Signing Key" unless you need to generate multiple keys or want a trusted signing key with a different key label and/or password.

Create the Trusted Signing Key

The **mksecpki** command automatically generates a trusted signing key with a key label of **TrustedKey**, the password of the CA user account, and places it in the **/usr/lib/security/pki/trusted.pkcs12** keystore file. If you need to generate a new trusted signing key or multiple trusted signing keys, then this section provides the steps needed to generate a trusted signing key.

All certificate authentication service clients where certificate creation and revocation are allowed require a trusted signing key for signing the user's authentication certificate. The key is saved in a separate keystore and is made available to all systems where certificates can be created. A single key can be used by all systems or, for a more secure approach, multiple keys can be created and distributed.

To create a trusted key, use the **/usr/java131/bin/keytool** command. Use a file name of a non-existing file. The **keytool** command prompts for a keystore password and key password. Both the keystore password and key password must be identical for certificate authentication service to access the key in the keystore. Run the **keytool** command as follows:

```
keytool -genkey -dname `cn=trusted key' -alias `TrustedKey' -keyalg RSA \
        -keystore filename.pkcs12 -storetype pkcs12ks
```

In this example, the trusted key label is **TrustedKey** and the trusted keystore password is user-supplied. Remember these values, because you will need them when configuring the certificate authentication service clients. When configuring a certificate authentication service client, the **keylabel** and **keypasswd** attributes in the **acct.cfg** file will need to be set to the trusted key label and trusted keystore password, respectively.

For security reasons, make sure the keystore file (*filename*.pkcs12) is read and write protected. Only the root user should have access to this file. The trusted key should be the only object in the keystore.

Configure Certificate Authentication Service Client

There are many configuration options on the client side of certificate authentication service. The following sections provide the configuration procedure required for each system participating in PKI authentication.

Install the Trusted Signing Key

Copy the trusted keystore containing the trusted signing key to the local system. For information on creating the trusted signing key, see "Create the Trusted Signing Key". The default location for the trusted keystore is in the **/usr/lib/security/pki** directory.

For security reasons, make sure the keystore file is read and write protected. Only the root user should have access to this file.

Edit the acct.cfg File

Remove any **ldap** stanzas that may exist in the **/usr/lib/security/pki/acct.cfg** file using a text-based editor like the **vi** command.

Configure the Certificate Authority

Minimally, the local CA account must be configured. By default, the local CA account exists, but must be modified to match your environment.

Certificate authentication service supports the use of multiple CA's by a single system through stanza-based configuration files. The default CA stanza name of **local** is used when a CA is not specified by a user or by the software. All systems must have a valid local stanza definition in the appropriate certificate authentication service configuration files. Only one CA may have a stanza name of **local**. All other CA's must have a unique stanza name. CA stanza names cannot be **ldap** or **default**.

The following sections guide you through the SMIT configuration screens for configuring the local CA.

Change / Show a Certificate Authority:

1. Run PKI SMIT, as follows:
   ```
   smitty pki
   ```
2. Select **Change / Show a Certificate Authority**.
3. Type `local` for the Certificate Authority Name field and press enter.
4. Set the **Service Module Name** field to **/usr/lib/security/pki/JSML.sml**. This is the default SML load module. This field maps to the **program** attribute in the **/usr/lib/security/pki/ca.cfg** file.
5. Ignore the **Pathname of CA's Certificate** field. This field maps to the **certfile** attribute in the **/usr/lib/security/pki/ca.cfg** file.
6. Set the **Pathname of CA's Trusted Key** field to a URI that is the location of the trusted keystore on the local system. Only file-based keystores are supported. The typical location for the trusted keystore is in the **/usr/lib/security/pki** directory. (See "Install the Trusted Signing Key" on page 97.) This field maps to the **trustedkey** attribute in the **/usr/lib/security/pki/ca.cfg** file.
7. Set the **URI of the Certificate Authority Server** field to a URI that is the location of the CA (**cmp://myserver:1077**). This field maps to the **server** attribute in the **/usr/lib/security/pki/ca.cfg** file.
8. Ignore the **Certificate Distribution Point** field. This field maps to the **cdp** attribute in the **/usr/lib/security/pki/ca.cfg** file.
9. Set the **Certificate Revocation List (CRL) URI** field. This field specifies the URI that should be set to the location of the certificate revocation list for this CA. This is typically an LDAP URI, for example:
   ```
   ldap://crlserver/o=XYZ,c=us
   ```
   This field maps to the **crl** attribute in the **/usr/lib/security/pki/ca.cfg** file.
10. The **Default Certificate Distinguished Name** field specifies the baseline DN used when creating certificates (for example, `o=XYZ,c=us`). This field is *not* required. This field maps to the **dn** attribute in the **/usr/lib/security/pki/ca.cfg** file.
11. The **Default Certificate Subject Alternate Name URI** field specifies the default subject alternate name URI used when creating certificates if a subject alternate name URI is not provided at creation time. This field is *not* required. This field maps to the **url** attribute in the **/usr/lib/security/pki/ca.cfg** file.
12. The **Public Key Algorithm** field specifies the public key algorithm used when creating a certificate. The choices are **RSA** and **DSA**. If neither are specified, the system defaults to **RSA**. This field maps to the **algorithm** attribute in the **/usr/lib/security/pki/ca.cfg** file.
13. The **Public Key Size (in bits)** field specifies the bit size of the public key algorithm. This field is in bits, not bytes, and this value may be rounded up by the underlying public key mechanism to support the next feasible byte size. (Typically, rounding occurs when the number of bits is not a even multiple of 8). Example values are `512`, `1024`, and `2048`. If this field is not specified, the system defaults to `1024` bits. This field maps to the **keysize** attribute in the **/usr/lib/security/pki/ca.cfg** file.
14. The **MAX. Communications Retries** field specifies the number of times the system attempts to contact the CA (when creating or revoking a certificate) before giving up. The system defaults to 5 attempts. This field maps to the **retries** attribute in the **/usr/lib/security/pki/ca.cfg** file.
15. The **Signing Hash Algorithm** field specifies the hash algorithm used when signing an authentication certificate. The choices are **MD2**, **MD5**, and **SHA1**. The system defaults to **MD5**. This field maps to the **signinghash** attribute in the **/usr/lib/security/pki/ca.cfg** file.
16. Press Enter to commit the changes.

Change / Show CA Accounts:

1. Run PKI SMIT, as follows:

   ```
   smitty pki
   ```

2. Select **Change / Show CA Accounts**.
3. Type `local` for the **Certificate Authority Name** field and press Enter.
4. The **Certificate Creation Reference Number** field specifies the CA's reference number used in creating a certificate. The creation reference number must be composed of all digits and be at least 7 digits in length. The reference number is defined by the CA. (See "Create the Certificate Authority" on page 96.) This field maps to the **carefnum** attribute in the **/usr/lib/security/pki/acct.cfg** file.
5. The **Certificate Creation Password** field specifies the CA's reference password used when creating a certificate. The creation password must be composed of 7-bit ASCII alpha-numerics and be at least 12 characters in length. The creation password is defined at the CA and must be the matching password to the creation reference number above. (See "Create the Certificate Authority" on page 96.) This field maps to the **capasswd** attribute in the **/usr/lib/security/pki/acct.cfg** file.
6. The **Certificate Revocation Reference Number** field specifies the reference number used when revoking a certificate. The revocation reference number must be composed of all digits and be at least 7 digits in length. The revocation reference number is sent to the CA during each certificate creation and is associated with the certificate by the CA. To revoke a certificate, the same revocation reference number (and revocation password) must be sent during revocation as was sent when creating the certificate. This field maps to the **rvrefnum** attribute in the **/usr/lib/security/pki/acct.cfg** file.
7. The **Certificate Revocation Password** field specifies the reference password used when revoking a certificate. The revocation password must be composed of 7-bit ASCII alpha-numerics and be at least 12 characters in length. The revocation password is sent to the CA during each certificate creation and is associated with the certificate by the CA. To revoke a certificate, the same revocation password (and revocation reference number) must be sent during revocation as was sent when creating the certificate. This field maps to the **rvpasswd** attribute in the **/usr/lib/security/pki/acct.cfg** file.
8. The **Trusted Key Label** field specifies the label (sometimes called *alias*) of the trusted signing key located in the trusted keystore. The trusted key label value is the value from "Create the Trusted Signing Key" on page 97. This field maps to the **keylabel** attribute in the **/usr/lib/security/pki/acct.cfg** file.
9. The **Trusted Key Password** field specifies the password of the trusted signing key located in the trusted keystore. The trusted key password value is the value from "Create the Trusted Signing Key" on page 97. This field maps to the **keypasswd** attribute in the **/usr/lib/security/pki/acct.cfg** file.
10. Press Enter to commit the changes.

Add the CA LDAP Account:

1. Run PKI SMIT, as follows:

   ```
   smitty pki
   ```

2. Select **Add an LDAP Account**.
3. The **Administrative User Name** field specifies the LDAP administrative account DN. The administrative user name for the CA LDAP account is the same name used in both "LDAP Server Configuration" on page 93 and "Configure LDAP For Certificate Authentication Service Server" on page 95. The value should be `cn=admin`. It is used by the client side to communicate with the LDAP server when accessing CA LDAP data. This field maps to the **ldappkiadmin** attribute in the **/usr/lib/security/pki/acct.cfg** file. For example:

   ```
   ldappkiadmin = "cn=admin"
   ```

4. The **Administrative Password** field specifies the LDAP administrative account password. The administrative password is the same password used in both "LDAP Server Configuration" on page 93 and "Configure LDAP For Certificate Authentication Service Server" on page 95. This field maps to the **ldappkiadmpwd** attribute in the **/usr/lib/security/pki/acct.cfg** file. For example:

```
ldappkiadmpwd = secret
```

5. The **Server Name** field specifies the name of the LDAP server and must be defined in every LDAP stanza. The value is a single LDAP server name. This field maps to the **ldapservers** attribute in the **/usr/lib/security/pki/acct.cfg** file. For example:

   ```
   ldapservers = ldapserver.mydomain.com
   ```

6. The **Suffix** field specifies the DN suffix for the directory information tree where the data resides. The suffix is the value of the **ibm-slapdSuffix** attribute used in "Configure LDAP For Certificate Authentication Service Server" on page 95. This attribute must be defined in every LDAP stanza. This field maps to the **ldapsuffix** attribute in the **/usr/lib/security/pki/acct.cfg** file. For example:

   ```
   ldapsuffix = "ou=aix,cn=us"
   ```

7. Press Enter to commit the changes.

Add the PKI Per-User LDAP Account: Perform the same steps as in "Add the CA LDAP Account" on page 99, except use the values used in the **Adding the PKI Suffix** and **ACL Database Entries** step in "Configuring the LDAP Server for PKI" on page 93. Use the following values:

- Administrative User Name (`ou=pkidata,cn=aixdata`),
- Administrative Password (*password*),
- Server Name (*site specific*),
- Suffix (`ou=pkidata,cn=aixdata`).

Press Enter to commit the changes.

Change / Show the Policy:

1. Run PKI SMIT, as follows:

   ```
   smitty pki
   ```

2. Select **Change / Show the Policy**.

- The **Create Certificates for New Users** field specifies whether the **mkuser** command generates a certificate and keystore for the new user (**new**), or if the administrator provides a certificate and keystore after the user is created (**get**). This field maps to the **cert** attribute of the **newuser** stanza in the **/usr/lib/security/pki/policy.cfg** file.
- The **Certificate Authority Name** field specifies the CA used by the **mkuser** command when generating a certificate. The field value must be a stanza name found in the **ca.cfg** file; for example, **local**. This field maps to the **ca** attribute of the **newuser** stanza in the **/usr/lib/security/pki/policy.cfg** file.
- The **Initial User Password** field specifies the password used by the **mkuser** command when creating a user's keystore. This field maps to the **passwd** attribute of the **newuser** stanza in the **/usr/lib/security/pki/policy.cfg** file.
- The **Certificate Version** field specifies the certificate version used by the **mkuser** command when generating a certificate. Currently, the only supported value is 3, which represents X.509v3. This field maps to the **version** attribute of the **newuser** stanza in the **/usr/lib/security/pki/policy.cfg** file.
- The **Public Key Size** field specifies the size (in bits) of the public key used by the **mkuser** command when generating a certificate. This field maps to the **keysize** attribute of the **newuser** stanza in the **/usr/lib/security/pki/policy.cfg** file.
- The **Keystore Location** field specifies the keystore directory in URI format used by the **mkuser** command when creating a keystore. This field maps to the **keystore** attribute of the **newuser** stanza in the **/usr/lib/security/pki/policy.cfg** file.
- The **Validity Period** field specifies the certificate's requested validity period used by the **mkuser** command when generating a certificate. The requested validity period may or may not be honored by the CA when creating the certificate. The period can be specified in seconds, days, or years. If just a number is provided, it is assumed to be in seconds. If the letter `d` immediately follows the number, it is interpreted as days. If the letter `y` immediate follows the number, it is interpreted as years. Example values are:

  - 1y (for 1 year)
  - 30d (for 30 days)
  - 2592000 (for 30 days represented in seconds)

  This field maps to the **validity** attribute of the **newuser** stanza in the **/usr/lib/security/pki/policy.cfg** file.
- The **Replicate Non-Local Certificates** field specifies whether the **certlink** command saves a copy of a certificate (**yes**) or just the link to the certificate (**no**). This field maps to the **replicate** attribute of the **storage** stanza in the **/usr/lib/security/pki/policy.cfg** file.
- The **Check Certificate Revocation Lists** field specifies whether the **certadd** and **certlink** commands check the CRL before performing their tasks (**yes**) or not (**no**). This field maps to the **check** attribute of the **crl** stanza in the **/usr/lib/security/pki/policy.cfg** file.
- The **Default Communications Timeout** field specifies the timeout period in seconds used by the **certadd** and **certlink** commands when requesting certificate information using HTTP (for example, retrieving CRLs). This field maps to the **timeout** attribute of the **comm** stanza in the **/usr/lib/security/pki/policy.cfg** file.

The methods.cfg File

The **methods.cfg** file specifies the definitions of the authentication grammar used by the **registry** and **SYSTEM** attributes. Specifically, this is where the authentication grammar for **PKILDAP** (for PKI using LDAP) and **FPKI** (*files* PKI) must be defined and added by the system administrator.

Below is a typical **methods.cfg** definition. The stanza names **PKI**, **LDAP**, and **PKILDAP** are arbitrary names and can be changed by the administrator. This section uses these stanza names throughout for consistency.

```
PKI:
  program = /usr/lib/security/PKI
  options = authonly

LDAP:
  program = /usr/lib/security/LDAP

PKILDAP:
  options = auth=PKI,db=LDAP
```

To support roaming users, use the same **methods.cfg** stanza names and attribute values across all systems that support roaming users.

Administration Configuration Examples

Create a New PKI User Account

To create a new PKI user account, use the **mkuser** command and the appropriate **/usr/lib/security/methods.cfg** stanza name (**PKILDAP**). Depending on the attribute settings in the **/usr/lib/security/pki/policy.cfg** file, the **mkuser** command can automatically create a certificate for the user. Below is a **mkuser** example that creates the user account `bob`:

```
mkuser -R PKILDAP SYSTEM="PKILDAP" registry=PKILDAP bob
```

Convert a Non-PKI User Account to a PKI User Account

There are a couple of different approaches for converting a non-PKI user account into a PKI user account. The first approach allows the system administrator access to the user's private keystore initially, which may or may not be acceptable in a given environment, but is the quickest way to convert a user. The second way requires interaction between the user and system administrator, which may take more time to setup.

Both examples use the following assumptions:

- **cas.server** and **cas.client** are already installed, configured, and working.

- **PKILDAP** is defined in **methods.cfg** as shown in “The methods.cfg File” on page 101.

Example 1:

With root authority, the system administrator can perform the following commands for user account bob:

```
certcreate -f cert1.der -l auth_lbl1 cn=bob bob  # Create & save cert in cert1.der.
certadd -f cert1.der -l auth_lbl1 auth_tag1 bob  # Add cert to LDAP as auth_tag1.
certverify auth_tag1 bob                         # Verify & sign the cert in LDAP.
chuser SYSTEM="PKILDAP" registry=PKILDAP bob     # Change account type to PKILDAP.
chuser -R PKILDAP auth_cert=auth_tag1 bob        # Set the user's auth certificate.
```

Then, have user bob change his password on the keystore using the **keypasswd** command.

Example 2:

Have user bob execute the first 3 commands of example 1 above (**certcreate**, **certadd**, **certverify**), creating his own certificate and keystore. Then have the system administrator perform the last two **chuser** commands of example 1 above.

Create and Add an Authentication Certificate

If a PKI user requires a new authentication certificate, the user can create a new certificate and have the system administrator make it the user’s authentication certificate. Below is an example of user bob creating a certificate and the system administrator making the certificate the authentication certificate.

```
# Logged in as user account bob:
certcreate -f cert1.der -l auth_lbl1 cn=bob  # Create & save cert in cert1.der.
certadd -f cert1.der -l auth_lbl1 auth_tag1  # Add cert to LDAP as auth_tag1.
certverify auth_tag1                         # Verify & sign the cert in LDAP.
# As the system adminstrator:
chuser -R PKILDAP auth_cert=auth_tag1 bob    # Set the user's auth certificate.
```

Change the Default New-Keystore Password

Edit the **passwd** attribute value of the **newuser** stanza in the **/usr/lib/security/pki/policy.cfg** file to modify the password used to create the keystores of new PKI users.

Handling a Compromised Trusted Signing Key

The file that contains the trusted signing key needs to be replaced and the user authentication certificates need to be re-signed.

Handling a Compromised User Private Key

If a user’s private key is compromised, the user or the administrator should revoke the certificate using the appropriate reason code, other users that use the public key should be notified of the compromise and, depending on the purpose of the private/public key, a new certificate should be issued. If the certificate was used as the user’s authentication certificate, then another certificate (either the new certificate or an existing non-promised certificate owned by the user) should be added as the new authentication certificate.

Handling a Compromised Keystore or Keystore Password

Change the password on the keystore. Revoke all the user’s certificates. Create new certificates for the user including a new authentication certificate. The compromised private keys may still be useful to the user for accessing previously encrypted data.

Moving a User’s Keystore or Changing the Name of a User’s Keystore

If a user’s private key is compromised, the user or the administrator should revoke the certificate using the appropriate reason code, other users that use the public key should be notified of the compromise and, depending on the purpose of the private and public key, a new certificate should be issued. If the certificate was used as the user’s authentication certificate, then another certificate (either the new certificate or an existing non-promised certificate owned by the user) should be added as the new authentication certificate.

Moving a User's Keystore or Changing the Name of a User's Keystore

Every user certificate maintained in LDAP contains the keystore location of its matching private key. To move a user's keystore from one directory to another or to change the name of the keystore, requires changing the LDAP keystore location and name associated with the user's certificates. If the user uses multiple keystores, then extra care must be taken to change only the LDAP information of the certificates affected by the keystore change.

To move a keystore from **/var/pki/security/keys/user1.p12** to **/var/pki/security1/keys/user1.p12**:

```
# As root...

cp /var/pki/security/keys/user1.p12 /var/pki/security1/keys/user1.p12

# Retrieve a list of all the certificates associated with the user.
certlist ALL user1

# For each certificate associated with the keystore, do the following:
# A) Retrieve the certificate's private key label and its "verified" status.
# B) Retrieve the certificate from LDAP.
# C) Replace the certificate in LDAP using the same private key label,
# but the new keystore path name.
# D) If the certificate was previously verified, it must verified again.
# (Step D requires the password to the keystore.)

# Example modifying one certificate.
# Assume:

# username: user1

# cert tag: tag1

# key label: label1

# Retrieve the certificate's private key label.
certlist -a label tag1 user1

# Retrieve the certificate from LDAP and place it in file cert.der.
certget -f cert.der tag1 user1

# Replace the certificate in LDAP.
certadd -r -f cert.der -p /var/pki/security1/keys/user1.p12 -l label1 tag1 user1

# Re-verify the certificate if it was previously verified.
# (Need to know the keystore password.)
certverify tag1 user1
```

Chapter 7. Pluggable Authentication Module

The pluggable authentication module (PAM) framework provides system administrators with the ability to incorporate multiple authentication mechanisms into an existing system through the use of pluggable modules. Applications enabled to make use of PAM can be *plugged-in* to new technologies without modifying the existing applications. This flexibility allows administrators to do the following:

- Select any authentication service on the system for an application
- Use multiple authentication mechanisms for a given service
- Add new authentication service modules without modifying existing applications
- Use a previously entered password for authentication with multiple modules

The PAM framework consists of a library, pluggable modules, and a configuration file. The PAM library implements the PAM application programming interface (API) and serves to manage PAM transactions and invoke the PAM service programming interface (SPI) defined in the pluggable modules. Pluggable modules are dynamically loaded by the library based on the invoking service and its entry in the configuration file. Success is determined not only by the pluggable module but also by the behavior defined for the service. Through the concept of *stacking*, a service can be configured to authenticate through multiple authentication methods. If supported, modules can also be configured to use a previously submitted password rather than prompting for additional input.

The following illustration shows the interaction between applications, PAM library, configuration file, and PAM modules. The fictitious PAM applications (pam_login, pam_su, and pam_passwd) invoke the PAM API in the PAM library. The library determines the appropriate module to load based on the application entry in the configuration file and calls the PAM SPI in the module. Communication occurs between the PAM module and the library through the use of a conversation function implemented in the PAM module. Success or failure from the module and the behavior defined in the configuration file then determine if another module needs to be loaded. If so, the process continues; otherwise, the data is passed back to the application.

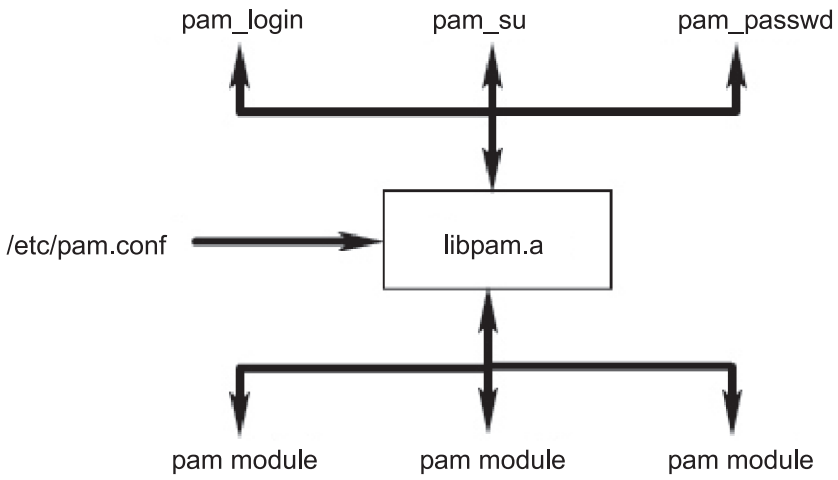


Figure 3. PAM Framework and Entities. This illustration shows how fictitious application commands use the PAM library to access the appropriate PAM module.

PAM Library

The PAM library, **/usr/lib/libpam.a**, contains the PAM-API that serves as a common interface to all PAM applications and also controls module loading. Modules are loaded by the PAM library based on the stacking behavior defined in the **/etc/pam.conf** file.

The following PAM API functions invoke the corresponding PAM SPI provided by a PAM module. For example, the **pam_authenticate** API invokes the **pam_sm_authenticate** SPI in a PAM module.

- **pam_authenticate**
- **pam_setcred**
- **pam_acct_mgmt**
- **pam_open_session**
- **pam_close_session**
- **pam_chauthtok**

Also provided in the PAM library are several functions that enable an application to invoke PAM modules and also to pass information to PAM modules. The following PAM framework APIs are implemented in AIX:

pam_start	Establish a PAM session
pam_end	Terminate a PAM session
pam_get_data	Retrieve module-specific data
pam_set_data	Set module-specific data
pam_get_item	Retrieve common PAM information
pam_set_item	Set common PAM information
pam_get_user	Retrieve user name
pam_strerror	Get PAM standard error message

PAM Modules

PAM modules allow multiple authentication mechanisms to be used collectively or independently on a system. A given PAM module must implement at least one of four module types. The module types are described as follows, along with the corresponding PAM SPIs that are required to conform to the module type.

Authentication Modules

Authenticate users and set, refresh, or destroy credentials. These modules identify user based on their authentication and credentials.

Authentication module functions:

- **pam_sm_authenticate**
- **pam_sm_setcred**

Account Management Modules

Determine validity of the user account and subsequent access after identification from authentication module. Checks performed by these modules typically include account expiration and password restrictions.

Account management module function:

- **pam_sm_acct_mgmt**

Session Management Modules

Initiate and terminate user sessions. Additionally, support for session auditing may be provided.

Session management module functions:

- **pam_sm_open_session**
- **pam_sm_close_session**

Password Management Modules

Perform password modification and related attribute management.

Password management module functions:

- **pam_sm_chauthtok**

PAM Configuration File

The **/etc/pam.conf** configuration file consists of service entries for each PAM module type and serves to route services through a defined module path. Entries in the file are composed of the following whitespace-delimited fields:

service_name module_type control_flag module_path module_options

Where:

service_name	Specifies the name of the service. The keyword OTHER is used to define the default module to use for applications not specified in an entry.
module_type	Specifies the module type for the service. Valid module types are auth, account, session, or password.
control_flag	Specifies the stacking behavior for the module. Supported control flags are required, sufficient, or optional.
module_path	Specifies the path name to a library object that implements the service functionality. Entries for *module_path* should start from the root (**/**) directory. If the entry does not begin with **/**, then **/usr/lib/security** will be prepended to the file name.
module_options	Specifies a list of options that can be passed to the service modules. Values for this field are dependent on the options supported by the module defined in the *module_path* field.

All fields defined above are required for each entry except for the *module_options* field, which is optional. Malformed entries or entries with invalid values for the *module_type* or *control_flag* fields are ignored by the PAM library. Entries beginning with a number sign (#) character at the beginning of the line are also ignored because this denotes a comment.

Stacking is implemented in the configuration file by creating multiple entries with the same *module_type* field. The modules are invoked in the order in which they are listed in the file, with the final result determined by the *control_flag* field specified for each entry. Valid values for the *control_flag* field and the corresponding behavior in the stack are as follows:

required	All required modules in a stack must pass for a successful result. If one or more of the required modules fail, all of the required modules in the stack will be attempted, but the error from the first failed required module is returned.
sufficient	If a module flagged as sufficient succeeds and no previous required or sufficient modules have failed, all remaining modules in the stack are ignored and success is returned.
optional	If none of the modules in the stack are required and no sufficient modules have succeeded, then at least one optional module for the service must succeed. If another module in the stack is successful, a failure in an optional module is ignored.

The following is an example **/etc/pam.conf** file that could be used on a system that has additional PAM modules installed:

```
#
# PAM configuration file /etc/pam.conf
#

# Authentication Management
```

```
login   auth     required       /usr/lib/security/pam_aix
login   auth     required       /usr/lib/security/pam_verify
login   auth     optional       /usr/lib/security/pam_test      use_first_pass
su      auth     sufficient     /usr/lib/security/pam_aix
su      auth     required       /usr/lib/security/pam_verify
OTHER   auth     required       /usr/lib/security/pam_aix

# Account Management
OTHER   account  required       /usr/lib/security/pam_aix

# Session Management
OTHER   session  required       /usr/lib/security/pam_aix

# Password Management
OTHER   password required       /usr/lib/security/pam_aix
```

The example configuration file contains three entries for the login service. Having specified both **pam_aix** and **pam_verify** as required, the user must enter two passwords to be authenticated, and both passwords must succeed for the user to be authenticated. The third entry for the **pam_test** module is optional and its success or failure will not affect whether the user is able to login. The option `use_first_pass` to the **pam_test** module allows a previously entered password to be used instead of prompting for a new one.

The **su** command behaves such that if **pam_aix** succeeds, authentication succeeds. If **pam_aix** fails, then **pam_verify** must pass for successful authentication.

Use of the **OTHER** keyword as a service name enables a default to be set for any other services that are not explicitly declared in the configuration file. Setting up a default ensures that all cases for a given module type will be covered by at least one module.

Adding a PAM Module

To add a PAM module, use the following procedure:

1. Install the module in the **/usr/lib/security** directory.
2. Set file ownership to root and permissions to 555. The PAM library does not load any module not owned by the root user.
3. Update the **/etc/pam.conf** configuration file to include the module in entries for the desired service names.
4. Test the affected services to ensure their functionality. Do not log off the system until a login test has been performed.

Changing the /etc/pam.conf

When changing the **/etc/pam.conf** configuration file, consider the following:

- AIX does not provide a default **/etc/pam.conf** file, so the file must be created before using PAM. When creating the file, set the file ownership to root and base permissions to 644. The file can then be manually edited by the root user to make the desired changes.
- Determine the default module to use for each module type and then make use of the **OTHER** keyword to prevent specifying the module for each service.
- Read any documentation supplied for a chosen module, and determine which control flags and options are supported and what their impact will be.
- Select the ordering of modules and control flags carefully, keeping in mind the behavior of **required**, **sufficient**, and **optional** control flags in stacked modules.

Note: Incorrect configuration of the PAM configuration file can result in a system that cannot be logged in to. After making changes to the file, always test the affected applications before logging out of the

system. A system that cannot be logged in to can be recovered by booting the system in maintenance mode and correcting the **/etc/pam.conf** configuration file.

Enabling PAM Debug

The PAM library can provide debug information during execution. After enabling the system to collect debug output, the information gathered can be used to track PAM-API invocations and determine failure points in the current PAM setup. To enable PAM debug output, follow these steps:

1. Create an empty file at **/etc/pam_debug**. The PAM library checks for existence of the **/etc/pam_debug** file and if found, enables syslog output.
2. Edit the **/etc/syslog.conf** file to contain the appropriate entries for the desired levels of messages.
3. Restart the **syslogd** daemon so that configuration changes are recognized.
4. When the PAM application is restarted, debug messages will be collected in the output file defined in the **/etc/syslog.conf** configuration file.

Integrating PAM in AIX

The integration of PAM in AIX is accomplished through the use of an AIX loadable authentication module, PAM, and a **pam_aix** module. These modules provide the following independent paths of PAM integration:

- Access is provided from AIX security services to PAM by means of the PAM module
- Access is provided from a PAM application to AIX security services through the PAM module (**pam_aix**)

PAM Module

AIX security services can be configured to call PAM modules through the use of the existing AIX loadable authentication module framework. When the **/usr/lib/security/methods.cfg** file is set up correctly, the simple load module PAM will route AIX security services (**passwd**, **login**, and so on) to the PAM library. The PAM library will check the **/etc/pam.conf** file to determine which PAM module to use and then make the corresponding PAM SPI call. Return values from PAM are mapped to AIX error codes and returned to the calling program.

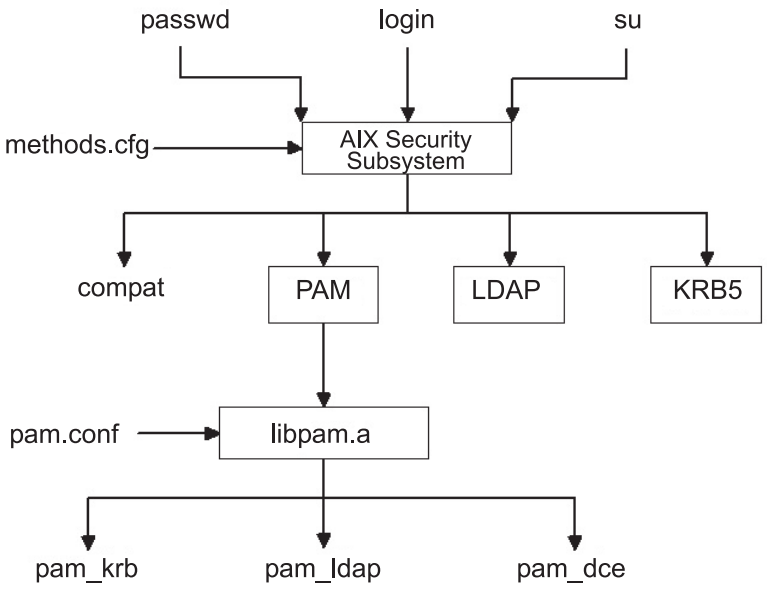


*Figure 4. AIX Security Service to PAM Module Path. This illustration shows the path an AIX security service call will take when PAM is properly configured. The PAM modules shown (***pam_krb***, ***pam_ldap***, and* ***pam_dce****) are listed as examples of third party solutions.*

PAM is a simple load module that is installed in the **/usr/lib/security** directory and is an authentication only module. The PAM module must be combined with a database to form a compound load module. The following example shows the stanzas that could be added to the **methods.cfg** file to form a compound PAM module with a database called **files**. The **BUILTIN** keyword for the **db** attribute will designate the database as UNIX files.

```
PAM:
        program = /usr/lib/security/PAM

PAMfiles:
        options = auth=PAM,db=BUILTIN
```

Creating and modifying users is then performed by using the **-R** option with the administration commands and by setting the **SYSTEM** attribute when a user is created.

```
mkuser -R PAMfiles SYSTEM=PAMfiles registry=PAMfiles pamuser
```

This act will inform further calls to AIX security services (**login**, **passwd**, and so on) to use the PAM load module for authentication. While the **files** database was used for the compound module in this example, other databases, like LDAP, can also be used if they are installed. Creating users as previously described will result in the following mapping of AIX security to PAM API calls:

```
 AIX                     PAM API
=====                   =========
authenticate       -->  pam_authenticate
chpass             -->  pam_chauthtok
passwdexpired      -->  pam_acct_mgmt
passwdrestrictions -->  No comparable mapping exists, success returned
```

Customizing the **/etc/pam.conf** file allows the PAM API calls to be directed to the desired PAM module for authentication. Stacking can be implemented in order to further refine the authentication mechanism.

Data prompted for by an AIX security service is passed to PAM through the **pam_set_item** function because it is not possible to accommodate user dialog from PAM. PAM modules written for integration with the PAM module should retrieve all data with **pam_get_item** calls and should not attempt to prompt the user to input data as this is handled by the security service.

Loop detection is provided to catch possible misconfiguration in which an AIX security service is routed to PAM and then a PAM module in turn attempts to call the AIX security service to perform the operation. Detection of this loop event will result in an immediate failure of the intended operation.

Note: The **/etc/pam.conf** file should NOT be written to make use of the **pam_aix** module when using PAM integration from an AIX security service to a PAM module as this will result in a loop condition.

pam_aix Module

The **pam_aix** module is a PAM module that provides PAM-enabled applications access to AIX security services by providing interfaces that call the equivalent AIX services where they exist. These services are in turn performed by a loadable authentication module or AIX **builtin** function based on the users definition and the corresponding setup in **methods.cfg**. Any error codes generated during execution of an AIX service are mapped to the corresponding PAM error code.

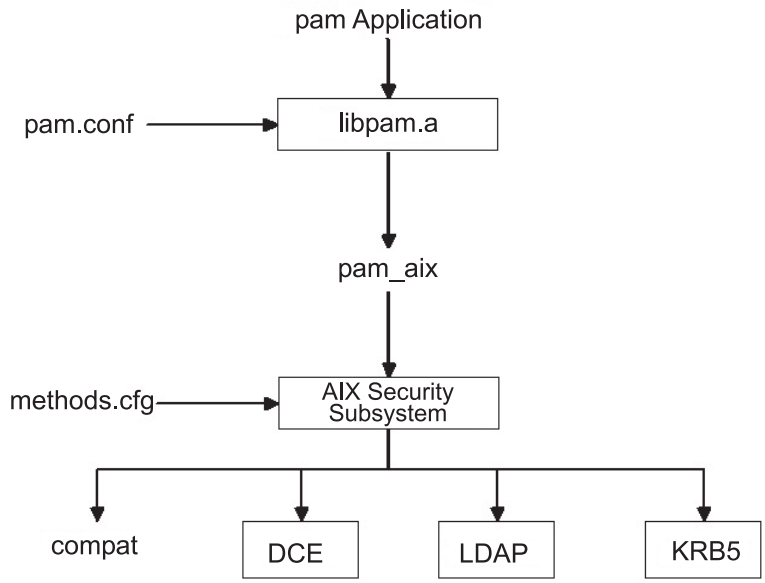


Figure 5. PAM Application to AIX Security Subsystem Path. This illustration shows the path a PAM application API call will follow if the **/etc/pam.conf** *file is configured to make use of the* **pam_aix** *module. As shown in the diagram, the integration allows users to be authenticated by any of the loadable authentication modules (DCE, LDAP, or KRB5) or in UNIX files (compat).*

The **pam_aix** module is installed in the **/usr/lib/security** directory. Integration of the **pam_aix** module requires that the **/etc/pam.conf** file be configured to make use of the module. Note that stacking is still available but chosen not to be shown in the following simple example of the **/etc/pam.conf** file:

```
#
# Authentication management
#
OTHER   auth     required       /usr/lib/security/pam_aix

#
# Account management
#
OTHER   account  required       /usr/lib/security/pam_aix

#
# Session management
#
OTHER   session  required       /usr/lib/security/pam_aix

#
# Password management
#
OTHER   password required       /usr/lib/security/pam_aix
```

The **pam_aix** module has implementations for the **pam_sm_authenticate**, **pam_sm_chauthok** and **pam_sm_acct_mgmt** SPI functions. The **pam_sm_setcred**, **pam_sm_open_session**, and **pam_sm_close_session**SPI are also implemented in the **pam_aix** module, but these SPI simply return PAM_SUCCESS invocations.

The following is a rough mapping of PAM SPI calls to the AIX security subsystem:

```
 PAM SPI                    AIX
=========                  =====
pam_sm_authenticate   -->  authenticate
pam_sm_chauthtok      -->  passwdexpired, chpass
                           Note: passwdexpired is only checked if the
                           PAM_CHANGE_EXPIRED_AUTHTOK flag is passed in.
```

```
pam_sm_acct_mgmt      -->  loginrestrictions, passwdexpired
pam_sm_setcred        -->  No comparable mapping exists, PAM_SUCCESS returned
pam_sm_open_session   -->  No comparable mapping exists, PAM_SUCCESS returned
pam_sm_close_session  -->  No comparable mapping exists, PAM_SUCCESS returned
```

Data intended to be passed to the AIX security subsystem may either be set using the **pam_set_item** function prior to module use or the **pam_aix** module will prompt for data if it does not already exist.

Chapter 8. OpenSSH Software Tools

OpenSSH software tools support the SSH1 and SSH2 protocols. The tools provide shell functions where network traffic is encrypted and authenticated. OpenSSH is based on client and server architecture. OpenSSH runs the **sshd** daemon process on the AIX host and waits for the connection from clients. It supports public-key and private-key pairs for authentication and encryption of channels to ensure secure network connections and host-based authentication. For more information about OpenSSH, see the following Web site:

```
http://www.openssh.org
```

The preceding Web site provides the man page information on the OpenSSH commands.

For more information about OpenSSH on AIX, see the following Web site, which has the latest **installp** format packages for AIX 5L:

```
http://oss.software.ibm.com/developerworks/projects/opensshi
```

This section explains how to install and configure OpenSSH on AIX. The OpenSSH software is shipped on the AIX 5.2 Bonus Pack. This version of OpenSSH is compiled and packaged as **installp** packages using the **openssh-3.4p1** level of source code. The OpenSSH program contained in the Bonus Pack CD-ROM media is licensed under the terms and conditions of the IBM International Program License Agreement (IPLA) for Non-Warranted Programs. OpenSSH is also available for AIX 4.3.3 in several RPM format packages, provided by the AIX toolbox for Linux applications.

Before installing the OpenSSH **installp** format packages, you must install the Open Secure Sockets Layer (OpenSSL) software. The OpenSSL software package contains the encrypted library. OpenSSL is provided in RPM packages in the AIX toolbox for Linux applications. The installation packages include the man pages and the translated message filesets.

1. Install the OpenSSL RPM package using the **geninstall** command, as follows:

   ```
   # geninstall -d/dev/cd0 R:openssl-0.9.6e
   ```

   Output similar to the following displays:

   ```
   SUCCESSES
   ---------
   openssl-0.9.6e-1
   ```

2. Next, install the OpenSSH **installp** packages using the **geninstall** comand as follows:

   ```
   # geninstall -I"Y" -d/dev/cd0 I:openssh.base
   ```

   Use the **Y** flag to accept the OpenSSH license agreement.

   Output similar to the following displays:

   ```
   Installation Summary
   --------------------
   Name                        Level           Part        Event       Result
   -------------------------------------------------------------------------------
   openssh.base.client         3.4.0.5200      USR         APPLY       SUCCESS
   openssh.base.server         3.4.0.5200      USR         APPLY       SUCCESS
   openssh.base.client         3.4.0.5200      ROOT        APPLY       SUCCESS
   openssh.base.server         3.4.0.5200      ROOT        APPLY       SUCCESS
   ```

You can also use the SMIT **install_software** fast path to install OpenSSL and OpenSSH.

The following OpenSSH binary files are installed as a result of the preceding procedure:

ssh	Similar to the **rlogin** and **rsh** client programs
ssh-agent	An agent that can store private keys

ssh-add	Tool that adds keys to **ssh-agent**
sftp	Similar to the **FTP** program that works over SSH1 and SSH2 protocol
scp	File copy program similar to **rcp**
ssh-keygen	Key generation tool
ssh-keyscan	Utility for gathering public host keys from a number of hosts
ssh-keysign	Utility for host-based authentication
sshd	Daemon that permits you to log in
sftp-server	SFTP server subsystem (started automatically by **sshd** daemon)

The following general information covers OpenSSH:

- The **/etc/ssh/ssh_config** directory contains the **sshd** daemon and the configuration files for the **ssh** command.
- The **/usr/openssh** directory contains the readme file and the original OpenSSH open-source license text file.
- The **sshd** daemon is under AIX SRC control. You can start, stop, and view the status of the daemon by issuing the following commands:

  ```
  startsrc -s sshd   OR startsrc -g ssh  (group)
  stopsrc -s sshd    OR stopsrc -g ssh
  lssrc -s sshd      OR lssrc -s ssh
  ```

  You can also start and stop the daemon by issuing the following commands:

  ```
  /etc/rc
  .d/rc2.d/Ksshd start
  ```

  OR

  ```
  /etc/rc.d/rc2.d/Ssshd start
  /etc/rc.d/rc2.d/Ksshd stop
  ```

  OR

  ```
  /etc/rc.d/rc2.d/Ssshd stop
  ```

- When the OpenSSH server fileset is installed, an entry is added to the directory **/etc/rc.d/rc2.d**. An entry is in inittab to execute run level 2 processes (`l2:2:wait:/etc/rc.d/rc 2`), so the **sshd** daemon will start automatically at boot time. To prevent the daemon from starting at boot time, remove the **/etc/rc.d/rc2.d/Ksshd** and **/etc/rc.d/rc2.d/Ssshd** files.
- OpenSSH software logs information to SYSLOG.
- The IBM Redbook, *Managing AIX Server Farms*, provides information about configuring OpenSSH in AIX and is available at the following Web site:

  ```
  http://www.redbooks.ibm.com
  ```

Using OpenSSH with PAM

Beginning with AIX 5.2, OpenSSH is compiled with Pluggable Authentication Module (PAM) support. PAM is an alternate way of authenticating users. It provides an adaptable mechanism for authenticating AIX users by allowing a user-written module to be added to the login process. A user can write his own module or use the **pam_aix** module provided with AIX. The **pam_aix** module provides interfaces to AIX security services.

The following is an example of the **/etc/pam.conf** configuration file using the **pam_aix** PAM module, but other modules may be used if installed on the system. Create the **/etc/pam.conf** file with the following information in that file:

```
sshd   auth            required        /usr/lib/security/pam_aix
OTHER   auth            required        /usr/lib/security/pam_aix
sshd    account         required        /usr/lib/security/pam_aix
OTHER   account         required        /usr/lib/security/pam_aix
sshd    password        required        /usr/lib/security/pam_aix
OTHER   password        required        /usr/lib/security/pam_aix
sshd    session         required        /usr/lib/security/pam_aix
OTHER   session         required        /usr/lib/security/pam_aix
```

Part 2. Network and Internet Security

Part 2 of this guide provides information about network and Internet security measures. These chapters describe how to install and configure IP Security; how to identify necessary and unecessary network services; auditing and monitoring network security, and more.

Chapter 9. TCP/IP Security

If you installed the Transmission Control Protocol/Internet Protocol (TCP/IP) and Network File System (NFS) software, you can configure your system to communicate over a network. This guide does not describe the basic concepts of TCP/IP, but rather describes security-related concerns of TCP/IP. For information on installing and the initial configuration of TCP/IP, refer to the Transmission Control Protocol/Internet Protocol chapter in *AIX 5L Version 5.2 System Management Guide: Communications and Networks*.

For any number of reasons, the person who administers your system might have to meet a certain level of security. For instance, the security level might be a matter of corporate policy. Or a system might need access to government systems and thus be required to communicate at a certain security level. These security standards might be applied to the network, the operating system, application software, even programs written by the person who administers your system.

This chapter describes the security features provided with TCP/IP, both in standard mode and as a secure system, and discusses some security considerations that are appropriate in a network environment.

After you install TCP/IP and NFS software, use the Web-based System Manager or the System Management Interface Tool (SMIT) **tcpip** fast path, to configure your system.

This chapter discusses the following topics:

Operating System-Specific Security

Many of the security features available for TCP/IP are based on those available through the operating system. The following sections outline TCP/IP security.

Network Access Control

The security policy for networking is an extension of the security policy for the operating system, and it consists of the following major components:

- **User authentication** is provided at the remote host by a user name and password in the same way as when a user logs in to the local system. Trusted TCP/IP commands, such as **ftp**, **rexec**, and **telnet**, have the same requirements and undergo the same verification process as trusted commands in the operating system.
- **Connection authentication** is provided to ensure that the remote host has the expected Internet Protocol (IP) address and name. This prevents a remote host from masquerading as another remote host.
- **Data import and export security** permits data at a specified security level to flow to and from network interface adapters at the same security and authority levels. For example, top-secret data can flow only between adapters that are set to the top-secret security level.

Network Auditing

Network auditing is provided by TCP/IP, using the audit subsystem to audit both kernel network routines and application programs. The purpose of auditing is to record those actions that affect the security of the system and the user responsible for those actions.

The following types of events are audited:

Kernel Events

- Change configuration
- Change host ID
- Change route
- Connection
- Create socket
- Export object
- Import object

Application Events

- Access the network
- Change configuration
- Change host ID
- Change static route
- Configure mail
- Connection
- Export data
- Import data
- Write mail to a file

Creation and deletion of objects are audited by the operating system. Application audit records suspend and resume auditing to avoid redundant auditing by the kernel.

Trusted Path, Trusted Shell, and Secure Attention Key (SAK)

The operating system provides the *trusted path* to prevent unauthorized programs from reading data from a user terminal. This path is used when a secure communication path with the system is required, such as when you are changing passwords or logging in to the system. The operating system also provides the *trusted shell* (**tsh**), which executes only trusted programs that have been tested and verified as secure. TCP/IP supports both of these features, along with the *secure attention key* (SAK), which establishes the environment necessary for secure communication between you and the system. The local SAK is available whenever you are using TCP/IP. A remote SAK is available through the **telnet** command.

The local SAK has the same function in **telnet** that it has in other operating system application programs: it ends the **telnet** process and all other processes associated with the terminal in which **telnet** was running. Inside the **telnet** program, however, you can send a request for a trusted path to the remote system using the **telnet send sak** command (while in **telnet** command mode). You can also define a single key to initiate the SAK request using the **telnet set sak** command.

For more information about the Trusted Computing Base, see “The Trusted Computing Base” on page 3.

TCP/IP Command Security

Some commands in TCP/IP provide a secure environment during operation. These commands are **ftp**, **rexec**, and **telnet**. The **ftp** function provides security during file transfer. The **rexec** command provides a secure environment for running commands on a foreign host. The **telnet** function provides security for login to a foreign host.

The **ftp**, **rexec**, and **telnet** commands provide security during their operation only. That is, they do not set up a secure environment for use with other commands. For securing your system for other operations, use the **securetcpip** command. This command gives you the ability to secure your system by disabling the nontrusted daemons and applications, and by giving you the option of securing your IP layer network protocol as well.

The **ftp**, **rexec**, **securetcpip**, and **telnet** commands provide the following forms of system and data security:

ftp

The **ftp** command provides a secure environment for transferring files. When a user invokes the **ftp** command to a foreign host, the user is prompted for a login ID. A default login ID is shown: the user's current login ID on the local host. The user is prompted for a password for the remote host.

The automatic login process searches the local user's **$HOME/.netrc** file for the user's ID and password to use at the foreign host. For security, the permissions on the **$HOME/.netrc** file must be set to 600 (read and write by owner only). Otherwise, automatic login fails.

> **Note:** Because use of the **.netrc** file requires storage of passwords in a nonencrypted file, the automatic login feature of the **ftp** command is not available when your system has been configured with the **securetcpip** command. This feature can be reenabled by removing the **ftp** command from the tcpip stanza in the **/etc/security/config** file.

To use the file transfer function, the **ftp** command requires two TCP/IP connections, one for the File Transfer Protocol (FTP) and one for data transfer. The protocol connection is primary and is secure because it is established on reliable communicating ports. The secondary connection is needed for the actual transfer of data, and both the local and remote host verify that the other end of this connection is established with the same host as the primary connection. If the primary and secondary connections are not established with the same host, the **ftp** command first displays an error message stating that the data connection was not authenticated, and then it exits. This verification of the secondary connection prevents a third host from intercepting data intended for another host.

rexec

The **rexec** command provides a secure environment for executing commands on a foreign host. The user is prompted for both a login ID and a password.

An automatic login feature causes the **rexec** command to search the local user's **$HOME/.netrc** file for the user's ID and password on a foreign host. For security, the permissions on the **$HOME/.netrc** file must be set to 600 (read and write by owner only). Otherwise, automatic login fails.

> **Note:** Because use of the **.netrc** file requires storage of passwords in a nonencrypted file, the automatic login feature of **rexec** is not available when your system is operating in secure. This feature can be reenabled by removing the **rexec** entry from the tcpip stanza in the **/etc/security/config** file.

securetcpip

The **securetcpip** command enables TCP/IP security features. Access to commands that are not trusted is removed from the system when this command is issued. Each of the following commands is removed by running the **securetcpip** command:

- **rlogin** and **rlogind**
- **rcp**, **rsh**, and **rshd**
- **tftp** and **tftpd**
- **trpt**

The **securetcpip** command is used to convert a system from the standard level of security to a higher security level. After your system has been converted, you need not issue the **securetcpip** command again unless you reinstall TCP/IP.

telnet or **tn**

The **telnet** (TELNET) command provides a secure environment for login to a foreign host. The user is prompted for both a login ID and a password. The user's terminal is treated just like a terminal connected directly to the host. That is, access to the terminal is controlled by permission bits. Other users (group and other) do not have read access to the terminal, but they can write messages to it if the owner gives them write permission. The **telnet** command also provides access to a trusted shell on the remote system through the SAK. This key sequence differs from the sequence that invokes the local trusted path and can be defined within the **telnet** command.

Remote Command Execution Access (/etc/hosts.equiv)

Users on the hosts listed in the **/etc/hosts.equiv** file can run certain commands on your system without supplying a password. The following table provides information about how to list, add, and remove remote hosts using Web-based System Manager, SMIT, or command line.

Remote command execution access tasks

Task	SMIT fast path	Command or file	Web-based System Manager Management Environment
List Remote Hosts That Have Command Execution Access	**smit lshostsequiv**	view **/etc/hosts.equiv** file	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **TCPIP Protocol Configuration** —> **TCP/IP** —> **Configure TCP/IP** —> **Advanced Methods** —> **Hosts File** —> **Contents of /etc/hosts file**.
Add a Remote Host for Command Execution Access	**smit mkhostsequiv**	edit **/etc/hosts.equiv** file[Note 1]	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **TCPIP Protocol Configuration** —> **TCP/IP** —> **Configure TCP/IP** —> **Advanced Methods** —> **Hosts File**. In **Add/Change host entry**, complete the following fields: **IP Address**, **Host name**, **Alias(es)**, and **Comment**. Click **Add/Change Entry**, and click **OK**.
Remove a Remote Host from Command Execution Access	**smit rmhostsequiv**	edit **/etc/hosts.equiv** file[Note 1]	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **TCPIP Protocol Configuration** —> **TCP/IP** —> **Configure TCP/IP** —> **Advanced Methods** —> **Hosts File**. Select a host in **Contents of /etc/host file**. Click **Delete Entry** —> **OK**.

Notes:

1. For more information about these file procedures, see the ″hosts.equiv File Format for TCP/IP″ in the *AIX 5L Version 5.2 Files Reference*.

Restricted File Transfer Program Users (/etc/ftpusers)

Users listed in the **/etc/ftpusers** file are protected from remote FTP access. For example, suppose that user A is logged into a remote system, and he knows the password of user B on your system. If user B is listed in the **/etc/ftpusers** file, user A cannot FTP files to or from user B's account, even though user A knows user B's password.

The following table provides information about how to list, add, and remove restricted users using Web-based System Manager, SMIT, or command line.

Remote FTP user tasks

Task	SMIT fast path	Command or file	Web-based System Manager Management Environment
List Restricted FTP Users	**smit lsftpusers**	view **/etc/ftpusers** file	Software —> **Users** —> **All Users**.
Add a Restricted User	**smit mkftpusers**	edit **/etc/ftpusers** file[Note 1]	Software —> **Users** —> **All Users** —> **Selected** —> **Add this User to Group**. Select a group, and click **OK**.
Remove a Restricted User	**smit rmftpusers**	edit **/etc/ftpusers** file[Note 1]	Software —> **Users** —> **All Users** —> **Selected** —> **Delete**.

Notes:

1. For more information about these file procedures, see the ″ftpusers File Format for TCP/IP″ in the *AIX 5L Version 5.2 Files Reference*.

Trusted Processes

A trusted program, or trusted process, is a shell script, a daemon, or a program that meets a particular standard of security. These security standards are set and maintained by the U.S. Department of Defense, which also certifies some trusted programs.

Trusted programs are trusted at different levels. Security levels include A1, B1, B2, B3, C1, C2, and D, with level A1 providing the highest security level. Each security level must meet certain requirements. For example, the C2 level of security incorporates the following standards:

program integrity Ensures that the process performs exactly as intended.

modularity Process source code is separated into modules that cannot be directly affected or accessed by other modules.

principle of least privilege States that at all times a user is operating at the lowest level of privilege authorized. That is, if a user has access only to view a certain file, then the user does not inadvertently also have access to alter that file.

limitation of object reuse Keeps a user from, for example, accidentally finding a section of memory that has been flagged for overwriting but not yet cleared, and which might contain sensitive material.

TCP/IP contains several trusted daemons and many nontrusted daemons.

Examples of trusted daemons are as follows:

- **ftpd**
- **rexecd**
- **telnetd**

Examples of nontrusted daemons are as follows:

- **rshd**
- **rlogind**
- **tftpd**

For a system to be trusted, it must operate with a trusted computing base; that is, for a single host, the machine must be secure. For a network, all file servers, gateways, and other hosts must be secure.

Network Trusted Computing Base

The Network Trusted Computing Base (NTCB) consists of hardware and software for ensuring network security. This section defines the components of the NTCB as they relate to TCP/IP.

The hardware security features for the network are provided by the network adapters used with TCP/IP. These adapters control incoming data by receiving only data destined for the local system and broadcast data receivable by all systems.

The software component of the NTCB consists of only those programs that are considered as trusted. The programs and associated files that are part of a secure system are listed in the following tables on a directory-by-directory basis.

/etc directory

Name	Owner	Group	Mode	Permissions
gated.conf	root	system	0664	rw-rw-r—
gateways	root	system	0664	rw-rw-r—
hosts	root	system	0664	rw-rw-r—
hosts.equiv	root	system	0664	rw-rw-r—
inetd.conf	root	system	0644	rw-r—r—
named.conf	root	system	0644	rw-r—r—
named.data	root	system	0664	rw-rw-r—
networks	root	system	0664	rw-rw-r—
protocols	root	system	0644	rw-r—r—
rc.tcpip	root	system	0774	rwxrwxr—
resolv.conf	root	system	0644	rw-rw-r—
services	root	system	0644	rw-r—r—
3270.keys	root	system	0664	rw-rw-r—
3270keys.rt	root	system	0664	rw-rw-r—

/usr/bin directory

Name	Owner	Group	Mode	Permissions
host	root	system	4555	r-sr-xr-x
hostid	bin	bin	0555	r-xr-xr-x
hostname	bin	bin	0555	r-xr-xr-x
finger	root	system	0755	rwxr-xr-x
ftp	root	system	4555	r-sr-xr-x
netstat	root	bin	4555	r-sr-xr-x
rexec	root	bin	4555	r-sr-xr-x

/usr/bin directory

Name	Owner	Group	Mode	Permissions
ruptime	root	system	4555	r-sr-xr-x
rwho	root	system	4555	r-sr-xr-x
talk	bin	bin	0555	r-xr-xr-x
telnet	root	system	4555	r-sr-xr-x

/usr/sbin directory

Name	Owner	Group	Mode	Permissions
arp	root	system	4555	r-sr-xr-x
fingerd	root	system	0554	r-xr-xr—
ftpd	root	system	4554	r-sr-xr—
gated	root	system	4554	r-sr-xr—
ifconfig	bin	bin	0555	r-xr-xr-x
inetd	root	system	4554	r-sr-xr—
named	root	system	4554	r-sr-x—
ping	root	system	4555	r-sr-xr-x
rexecd	root	system	4554	r-sr-xr—
route	root	system	4554	r-sr-xr—
routed	root	system	0554	r-xr-x—-
rwhod	root	system	4554	r-sr-xr—
securetcpip	root	system	0554	r-xr-xr—
setclock	root	system	4555	r-sr-xr-x
syslogd	root	system	0554	r-xr-xr—
talkd	root	system	4554	r-sr-xr—
telnetd	root	system	4554	r-sr-xr—

/usr/ucb directory

Name	Owner	Group	Mode	Permissions
tn	root	system	4555	r-sr-xr-x

/var/spool/rwho directory

Name	Owner	Group	Mode	Permissions
rwho (directory)	root	system	0755	drwxr-xr-x

Data Security and Information Protection

The security feature for TCP/IP does not encrypt user data transmitted through the network. Therefore, it is suggested that users identify any risk in communication that could result in the disclosure of passwords and other sensitive information, and based on that risk, apply appropriate countermeasures.

Using the TCP/IP security feature in a Department of Defense (DOD) environment might require adherence to DOD 5200.5 and NCSD-11 for communications security.

User Based TCP Port Access Control with Discretionary Access Control for Internet Ports

Discretionary Access Control for Internet Ports (DACinet) features user based access control for TCP ports for communication between AIX 5.2 hosts. AIX 5.2 can use an additional TCP header to transport user and group information between systems. The DACinet feature allows the administrator on the destination system to control access based on the destination port, the originating user id and host.

In addition, the DACinet feature allows the administrator to restrict local ports for root only usage. UNIX systems like AIX treat ports below 1024 as privileged ports which can only be opened by root. AIX 5.2 allows you to specify additional ports above 1024 which can be opened only by root, therefore preventing users from running servers on well known ports.

Depending on the settings a non-DACinet system may or may not be able to connect to a DACinet system. Access is denied in the initial state of the DACinet feature. Once DACinet has been enabled, there is no way to disable DACinet.

The **dacinet** command accepts addresses which are specified as hostnames, dotted decimal host addresses, or network addresses followed by the length of the network prefix.

The following example specifies a single host which is known by the fully qualified host name *host.domain.org*:

```
host.domain.org
```

The following example specifies a single host which is known by the IP address 10.0.0.1:

```
10.0.0.1
```

The following example specifies the entire network which has the first 24 bits (the length of the network prefix) with a value of 10.0.0.0:

```
10.0.0.0/24
```

This network includes all IP addresses between 10.0.0.1 and 10.0.0.254.

Access control for TCP based services

DACinet uses the **/etc/rc.dacinet** startup file and the configuration files it used are **/etc/security/priv**, **/etc/security/services**, and **/etc/security/acl**.

Ports listed in **/etc/security/services** are considered exempt from the ACL checks. The file has the same format as **/etc/services**. The easiest way to initialize it is to copy the file from **/etc** to **/etc/security** and then delete all the ports for which ACLs should be applied. The ACLs are stored in two places. The currently active ACLs are stored in the kernel and can be read by running **dacinet aclls**. ACLs that will be reactivated at the next system boot by **/etc/rc.tcpip** are stored in **/etc/security/acl**. The following format is used:

```
service host/prefix-length [user|group]
```

Where the service can be specified either numerically or as listed in **/etc/services**, the host can be given either as a host name or a network address with a subnet mask specification and the user or group is specified with the **u:** or **g:** prefix. When no user or group is specified, then the ACL takes only the sending host into account. Prefixing the service with a **-** will disable access explicitly. ACLs are evaluated according to the first match. So you could specify access for a group of users, but explicitly deny it for a user in the group by placing the rule for this user in front of the group rule.

The **/etc/services** file includes two entries with port number values which are not supported in AIX 5.2. The system administrator must remove both lines from that file prior to executing the **mkCCadmin** command. Remove the following lines from the **/etc/services** file:

```
sco_printer     70000/tcp     sco_spooler    # For System V print IPC
sco_s5_port     70001/tcp     lpNet_s5_port  # For future use
```

Examples for DACinet Usage

For example, when using DACinet to restrict access to port TCP/25 inbound to root only with the DACinet feature, then only root users from other AIX 5.2 hosts can access this port, therefore limiting the possibilities of regular users to spoof e-mail by just telneting to port TCP/25 on the victim. The following example shows how to configure the X protocol (X11) for root only access. Make sure that the X11 entry in **/etc/security/services** is removed, so that the ACLs will apply for this service.

Assuming a subnet of 10.1.1.0/24 for all the connected systems, the ACL entries to restrict access to the root user only for X (TCP/6000) in **/etc/security/acl** would be as follows:

```
      6000    10.1.1.0/24 u:root
```

When limiting Telnet service to users in the group **friends**, no matter from which system they are coming from, use the following ACL entry after having removed the telnet entry from **/etc/security/services**:

```
telnet    0.0.0.0/0   g:friends
```

Disallow user fred access to the web server, but allow everyone else access:

```
-80    0.0.0.0/0 u:fred
80     0.0.0.0/0
```

Privileged Ports for Running Local Services

Normally any user can open any port above 1024. For example, a user could place a server at port 8080, which is quite often used to run Web proxies or at 1080 where one typically finds a SOCKS server. To prevent regular users from running servers at specific ports, these ports can be designated as privileged. The **dacinet setpriv** command can be used to add privileged ports to the running system. Ports that are to be designated as privileged when the system starts have to be listed in **/etc/security/priv**.

Ports can be listed in this file either with their symbolic name as defined in **/etc/services** or by specifying the port number. The following entries would disallow non-root users to run SOCKS servers or Lotus Notes servers on their usual ports:

```
1080
lotusnote
```

Note: This feature does not prevent the **user** from running the programs. It will only prevent the user from running the services at the well known ports where those services are typically expected.

For more information on the **dacinet** command, refer to the *AIX 5L Version 5.2 Commands Reference*.

Chapter 10. Network Services

This chapter provides information about identifying and securing network services with open communication ports.

Identifying Network Services with Open Communication Ports

Client-server applications open communication ports on the server, allowing the applications to listen to incoming client requests. Because open ports are vulnerable to potential security attacks, identify which applications have open ports and close those ports that are open unnecessarily. This practice is useful because it allows you to understand what systems are being made available to anyone who has access to the Internet.

To determine which ports are open, do the following:

1. Identify the services by using the **netstat** command as follows:

   ```
   # netstat -af inet
   ```

   The following is an example of this command output. The last column of the **netstat** command output indicates the state of each service. Services that are waiting for incoming connections are in the LISTEN state.

Active Internet connection (including servers)

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	(state)
tcp4	0	0	*.echo	*.*	LISTEN
tcp4	0	0	*.discard	*.*	LISTEN
tcp4	0	0	*.daytime	*.*	LISTEN
tcp	0	0	*.chargen	*.*	LISTEN
tcp	0	0	*.ftp	*.*	LISTEN
tcp4	0	0	*.telnet	*.*	LISTEN
tcp4	0	0	*.smtp	*.*	LISTEN
tcp4	0	0	*.time	*.*	LISTEN
tcp4	0	0	*.www	*.*	LISTEN
tcp4	0	0	*.sunrpc	*.*	LISTEN
tcp	0	0	*.smux	*.*	LISTEN
tcp	0	0	*.exec	*.*	LISTEN
tcp	0	0	*.login	*.*	LISTEN
tcp4	0	0	*.shell	*.*	LISTEN
tcp4	0	0	*.klogin	*.*	LISTEN
udp4	0	0	*.kshell	*.*	LISTEN
udp4	0	0	*.echo	*.*	
udp4	0	0	*.discard	*.*	
udp4	0	0	*.daytime	*.*	
udp4	0	0	*.chargen	*.*	

Active Internet connection (including servers)

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	(state)
udp4	0	0	*.time	*.*	
udp4	0	0	*.bootpc	*.*	
udp4	0	0	*.sunrpc	*.*	
udp4	0	0	255.255.255.255.ntp	*.*	
udp4	0	0	1.23.123.234.ntp	*.*	
udp4	0	0	localhost.domain.ntp	*.*	
udp4	0	0	name.domain..ntp	*.*	

.....

2. Open the **/etc/services** file and check the Internet Assigned Numbers Authority (IANA) services to map the service to port numbers within the operating system.

   The following is a sample fragment of the **/etc/services** file:

```
tcpmux                          1/tcp                # TCP Port Service Multiplexer
tcpmux                          1/tcp                # TCP Port Service Multiplexer
Compressnet                     2/tcp                # Management Utility
Compressnet                     2/udp                # Management Utility
Compressnet                     3/tcp                # Compression Process
Compressnet                     3/udp                Compression Process
Echo                            7/tcp
Echo                            7/udp
discard                         9/tcp                sink null
discard                         9/udp                sink null
..............
rfe                             5002/tcp             # Radio Free Ethernet
rfe                             5002/udp             # Radio Free Ethernet
rmonitor_secure                 5145/tcp
rmonitor_secure                 5145/udp
pad12sim                        5236/tcp
pad12sim                        5236/udp
sub-process                     6111/tcp             # HP SoftBench Sub-Process Cntl.
sub-process                     6111/udp             # HP SoftBench Sub-Process Cntl.
xdsxdm                          6558/ucp
xdsxdm                          6558/tcp
afs3-fileserver                 7000/tcp             # File Server Itself
```

```
afs3-fileserver                 7000/udp            # File Server Itself
af3-callback                    7001/tcp            # Callbacks to Cache Managers
af3-callback                    7001/udp            # Callbacks to Cache Managers
```

3. Close the unnecessary ports by removing the running services.

Identifying TCP and UDP Sockets

Identify TCP sockets that are in the LISTEN state and idle UDP sockets that are waiting for data to arrive. Use the **lsof** command, a variant of the **netstat -af** command. Beginning with AIX 5.1, the **lsof** command is included on the *AIX Toolbox for Linux Applications* CD.

For example, to display the TCP sockets in the LISTEN state and the UDP sockets in the IDLE state, run the **lsof** command as follows:

```
# lsof -i | egrep "COMMAND|LISTEN|UDP"
```

The output produced is similar to the following:

Command	PID	USER	FD	TYPE	DEVICE	SIZE/OFF	NODE	NAME
dtlogin	2122	root	5u	IPv4	0x70053c00	0t0	UDP	*:xdmcp
dtlogin	2122	root	6u	IPv4	0x70054adc	0t0	TCP	*:32768(LISTEN)
syslogd	2730	root	4u	IPv4	0x70053600	0t0	UDP	*:syslog
X	2880	root	6u	IPv4	0x70054adc	0t0	TCP	*:32768(LISTEN)
X	2880	root	8u	IPv4	0x700546dc	0t0	TCP	*:6000(LISTEN)
dtlogin	3882	root	6u	IPv4	0x70054adc	0t0	TCP	*:32768(LISTEN)
glbd	4154	root	4u	IPv4	0x7003f300	0t0	UDP	*:32803
glbd	4154	root	9u	IPv4	0x7003f700	0t0	UDP	*:32805
dtgreet	4656	root	6u	IPv4	0x70054adc	0t0	TCP	*:32768(LISTEN)

.....

After identifying the process ID, you can obtain more information about the program by running the following command:

```
" # ps -fp PID#"
```

The output contains the path to the command name, which you can use to access the program's man page.

Chapter 11. Internet Protocol (IP) Security

IP Security enables secure communications over the Internet and within company networks by securing data traffic at the IP layer. This allows individual users or organizations to secure traffic for all applications, without having to make any modifications to the applications. Therefore, the transmission of any data, such as e-mail or application-specific company data, can be made secure.

This chapter discusses the following topics:

IP Security Overview

This section discusses the following topics:

- IP Security and the Operating System
- IP Security Features
- Security Associations
- Tunnels and Key Management
- Native Filtering Capability
- Digital Certificate Support
- Benefits of a Virtual Private Network

IP Security and the Operating System

The operating system uses IP Security (IPsec), which is an open, standard security technology developed by the Internet Engineering Task Force (IETF). IPsec provides cryptography-based protection of all data at the IP layer of the communications stack. No changes are needed for existing applications. IPsec is the industry-standard network-security framework chosen by the IETF for both the IP Version 4 and 6 environments.

IPsec protects your data traffic using the following cryptographic techniques:

Authentication
: Process by which the identity of a host or end point is verified

Integrity Checking
: Process of ensuring that no modifications were made to the data while in transit across the network

Encryption
: Process of ensuring privacy by ″hiding″ data and private IP addresses while in transit across the network

Authentication algorithms prove the identity of the sender and data integrity by using a cryptographic hash function to process a packet of data (with the fixed IP header fields included) using a secret key to produce a unique digest. On the receiver side, the data is processed using the same function and key. If either the data has been altered or the sender key is not valid, the datagram is discarded.

Encryption uses a cryptographic algorithm to modify and randomize the data using a certain algorithm and key to produce encrypted data known as *cyphertext*. Encryption makes the data unreadable while in transit. After it is received, the data is recovered using the same algorithm and key (with symmetric encryption algorithms). Encryption must occur with authentication to verify the data integrity of the encrypted data.

These basic services are implemented in IPsec by the use of the Encapsulating Security Payload (ESP) and the Authentication Header (AH). ESP provides confidentiality by encrypting the original IP packet, building an ESP header, and putting the cyphertext in the ESP payload.

The AH can be used alone for authentication and integrity-checking if confidentiality is not an issue. With AH, the static fields of the IP header and the data have a hash algorithm applied to compute a keyed digest. The receiver uses its key to compute and compare the digest to make sure the packet is unaltered and the sender's identity is authenticated.

IP Security Features

The IP Security feature of this operating system provides the following functions:

- Hardware acceleration with the 10/100 Mbps Ethernet PCI Adapter II.
- AH support using RFC 2402, and ESP support using RFC 2406.
- Certificate Revocation List support with retrieval using HTTP or LDAP servers.
- Automatic key refreshment with tunnels using IETF Internet Key Exchange (IKE) protocol.
- X.509 Digital Certificate and preshared key support in IKE protocol during key negotiation.
- Manual tunnels can be configured to provide interoperability with other systems that do not support the automatic IKE key refreshment method, and for use of IP Version 6 tunnels.
- Tunnel mode and transport mode of encapsulation for host or gateway tunnels.
- Authentication algorithms of HMAC (Hashed Message Authentication Code) MD5 (Message Digest 5) and HMAC SHA (Secure Hash Algorithm).
- Encryption algorithms include 56 bit Data Encryption Standard (DES) Cipher Block Chaining (CBC) with 64 bit initial vector (IV), Triple DES, DES CBC 4 (32 bit IV).
- Dual IP Stack Support (IP version 4 and IP version 6).
- Both IP Version 4 and IP Version 6 traffic can be encapsulated and filtered. Because the IP stacks are separate, the IP Security function for each stack can be configured independently.
- IKE tunnels can be created using Linux configuration files (AIX 5.1 and later).
- Filtering of secure and nonsecure traffic by a variety of IP characteristics such as source and destination IP addresses, interface, protocol, port numbers, and more.
- Automatic filter-rule creation and deletion with most tunnel types.
- Use of host names for the destination address when defining tunnels and filter rules. The host names are converted to IP addresses automatically (as long as DNS is available).
- Logging of IP Security events to **syslog**.
- Use of system traces and statistics for problem determination.
- User-defined default action allows the user to specify whether traffic that does not match defined tunnels should be allowed.

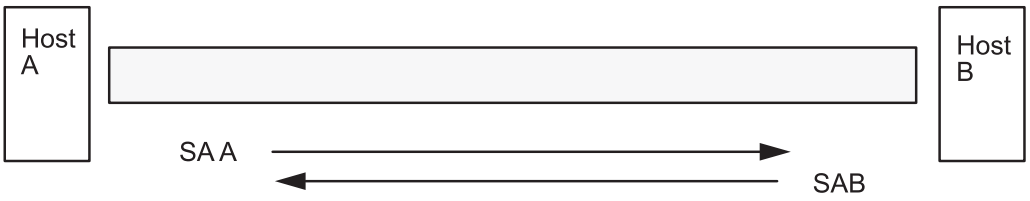
Internet Key Exchange (IKE) Features

The following features are available with Internet Key Exchange (beginning with AIX 4.3.2):

- Authentication with preshared keys and X.509 digital signatures.
- Use of main mode (identity protect mode) and aggressive mode.
- Support for Diffie Hellman groups 1, 2, and 5.
- ESP encryption support for Data Encryption Standard (DES), Triple DES, Null encryption; ESP authentication support with HMAC MD5 and HMAC SHA1.
- AH support for HMAC MD5 and HMAC SHA1.
- IP Version 4 and Version 6 support.

Security Associations

The building block on which secure communications is built is a concept known as a *security association*. Security associations relate a specific set of security parameters to a type of traffic. With data protected by IP Security, a separate security association exists for each direction and for each header type, AH or ESP. The information contained in the security association includes the IP addresses of the communicating parties, a unique identifier known as the Security Parameters Index (SPI), the algorithms selected for authentication or encryption, the authentication and encryption keys, and the key lifetimes. The following figure shows the security associations between Host A and Host B.



SA = Security Association, consisting of:

Destination address
SPI
Key
Crypto Algorithm and Format
Authentication Algorithm
Key Lifetime

Figure 6. Establishment of a Secure Tunnel Between Hosts A and B. This illustration shows a virtual tunnel running between Host A and Host B. Security association A is an arrow directed from Host A to Host B. Security association B is an arrow directed from Host B to Host A. A Security association consists of the Destination Address, SPI, Key, Crypto Algorithm and Format, Authentication Algorithm, and Key Lifetime.

The goal of key management is to negotiate and compute the security associations that protect IP traffic.

Tunnels and Key Management

To set up a secure communication between two hosts, security associations must be negotiated and managed during the use of the tunnel. The following types of tunnels are supported, each using a different key management technique:

- IKE tunnels (dynamically changing keys, IETF standard)
- Manual tunnels (static, persistent keys, IETF standard)

IKE Tunnel Support

IKE Tunnels are based on the ISAKMP/Oakley (Internet Security Association and Key Management Protocol) standards developed by the IETF. With this protocol, security parameters are negotiated and refreshed, and keys are exchanged securely. The following types of authentication are supported: preshared key and X.509v3 digital certificate signatures.

The negotiation uses a two-phase approach. The first phase authenticates the communicating parties, and specifies the algorithms to be used for securely communicating in phase 2. During phase 2, IP Security parameters to be used during data transfer are negotiated, security associations and keys are created and exchanged.

The following table shows the authentication algorithms that can be used with the AH and ESP security protocols for IKE tunnel support.

Algorithm	AH IP Version 4 & 6	ESP IP Version 4 & 6
HMAC MD5	X	X
HMAC SHA1	X	X
DES CBC 8		X
Triple DES CBC		X
ESP Null		X

Manual Tunnel Support

Manual tunnels provide backward compatibility, and they interoperate with machines that do not support IKE key management protocols. The disadvantage of manual tunnels is that the key values are static. The encryption and authentication keys are the same for the life of the tunnel and must be manually updated.

The following table shows the authentication algorithms that can be used with the AH and ESP security protocols for manual tunnel support.

Algorithm	AH IP Version 4	AH IP Version 6	ESP IP Version 4	ESP IP Version 6
HMAC MD5	X	X	X	X
HMAC SHA1	X	X	X	X
Triple DES CBC			X	X
DES CBC 8			X	X
DES CBC 4			X	X

Because IKE tunnels offer more effective security, IKE is the preferred key management method.

Native Filtering Capability

Filtering is a basic function in which incoming and outgoing packets can be accepted or denied based on a variety of characteristics. This allows a user or system administrator to configure the host to control the traffic between this host and other hosts. Filtering is done on a variety of packet properties, such as source and destination addresses, IP version (4 or 6), subnet masks, protocol, port, routing characteristics, fragmentation, interface, and tunnel definition.

Rules, known as *filter rules*, are used to associate certain kinds of traffic with a particular tunnel. In a basic configuration for manual tunnels, when a user defines a host-to-host tunnel, filter rules are autogenerated to direct all traffic from that host through the secure tunnel. If more specific types of traffic are desired (for instance, subnet to subnet), the filter rules can be edited or replaced to allow precise control of the traffic using a particular tunnel.

For IKE tunnels, the filter rules are also automatically generated and inserted in the filter table once the tunnel is activated.

Similarly, when the tunnel is modified or deleted, the filter rules for that tunnel are automatically deleted, which simplifies IP Security configuration and helps reduce human error. Tunnel definitions can be propagated and shared among machines and firewalls using import and export utilities, which is helpful in the administration of a large number of machines.

Filter rules associate particular types of traffic with a tunnel, but data being filtered does not necessarily need to travel in a tunnel. This aspect of filter rules lets the operating system provide basic firewall functionality to those who want to restrict traffic to or from their machine in an intranet or in a network that does not have the protection of a true firewall. In this scenario, filter rules provide a second barrier of protection around a group of machines.

After the filter rules are generated, they are stored in a table and loaded into the kernel. When packets are ready to be sent or received from the network, the filter rules are checked in the list from top to bottom to determine whether the packet should be permitted, denied, or sent through a tunnel. The criteria of the rule is compared to the packet characteristics until a match is found or the default rule is reached.

The IP Security function also implements filtering of non-secure packets based on very granular, user-defined criteria, which allows the control of IP traffic between networks and machines that do not require the authentication or encryption properties of IP Security.

Digital Certificate Support

IP Security supports the use of X.509 Version 3 digital certificates. The Key Manager tool manages certificate requests, maintains the key database, and performs other administrative functions.

Digital certificates are described in Digital Certificate Configuration. The Key Manager and its functions are described in Using the IBM Key Manager Tool

Virtual Private Networks and IP Security

A virtual private network (VPN) securely extends a private intranet across a public network such as the Internet. VPNs convey information across what is essentially a private tunnel through the Internet to and from remote users, branch offices, and business partners/suppliers. Companies can opt for Internet access through Internet service providers (ISPs) using direct lines or local telephone numbers and eliminate more expensive leased lines, long-distance calls, and toll-free telephone numbers. A VPN solution can use the IPsec security standard because IPsec is the IETF-chosen industry standard network security framework for both the IP Version 4 and 6 environments, and no changes are needed for existing applications.

A recommended resource for planning the implementation of a VPN in AIX is Chapter 9 of *A Comprehensive Guide to Virtual Private Networks, Volume III: Cross-Platform Key and Policy Management*, ISBN SG24-5309-00. This guide is also available on the Internet World Wide Web at http://www.redbooks.ibm.com/redbooks/SG245309.html.

Installing the IP Security Feature

The IP Security feature in AIX is separately installable and loadable. The file sets that need to be installed are as follows:

- **bos.net.ipsec.rte** (The run-time environment for the kernel IP Security environment and commands)
- **bos.msg.***LANG***.net.ipsec** (where *LANG* is the desired language, such as **en_US**)
- **bos.net.ipsec.keymgt**
- **bos.net.ipsec.websm**
- **bos.crypto-priv** (The file set for DES and triple DES encryption)

The **bos.crypto-priv** file set is located on the Expansion Pack. For IKE digital signature support, you must also install the **gskit.rte** fileset (AIX Version 4) or **gskkm.rte** (AIX 5.1) from the Expansion Pack.

After it is installed, IP Security can be separately loaded for IP Version 4 and IP Version 6, either by using the recommended procedure provided in “Loading IP Security” or by using the **mkdev** command.

Loading IP Security

> **Attention:** Loading IP Security enables the filtering function. Before loading, it is important to ensure the correct filter rules are created. Otherwise, all outside communication might be blocked.

Use SMIT or Web-based System Manager to automatically load the IP security modules when IP Security is started. Also, SMIT and Web-based System Manager ensure that the kernel extensions and IKE daemons are loaded in the correct order.

If the loading completes successfully, the **lsdev** command shows the IP Security devices as `Available`.

```
lsdev -C -c ipsec

   ipsec_v4 Available IP Version 4 Security Extension
   ipsec_v6 Available IP Version 6 Security Extension
```

After the IP Security kernel extension has been loaded, tunnels and filters are ready to be configured.

Planning IP Security Configuration

To configure IP Security, tunnels and filters must be configured. When a simple tunnel is defined for all traffic to use, the filter rules can be automatically generated. If more complex filtering is desired, filter rules can be configured separately.

You can configure IP Security using the Web-based System Manager Network plug-in, Virtual Private Network plug-in, or the System Management Interface Tool (SMIT). If using SMIT, the following fast paths are available:

smit ips4_basic
Basic configuration for IP version 4

smit ips6_basic
Basic configuration for IP version 6

Before configuring IP Security for your site, you must decide what method you intend to use; for example, whether you prefer to use tunnels or filters (or both), which type of tunnel best suits your needs, and so on. The following sections provide information you must understand before making these decisions:

- Hardware Acceleration
- Tunnels versus Filters
- Tunnels and Security Associations
- Choosing a Tunnel Type
- Using IKE with DHCP or Dynamically Assigned Addresses

Hardware Acceleration

The 10/100 Mbps Ethernet PCI Adapter II (Feature code 4962) offers standards-based IP Security and is designed to offload IP Security functions from the AIX operating system. When the 10/100 Mbps Ethernet PCI Adapter II is present in the AIX system, the IP Security stack uses the following capabilities of the adapter:

- Encryption and decryption using DES or Triple DES algorithms
- Authentication using the MD5 or SHA-1 algorithms
- Storage of the security-association information.

The functions on the adapter are used instead of the software algorithms. The 10/100 Mbps Ethernet PCI Adapter II is available for manual and IKE tunnels.

The IP Security hardware acceleration feature is available in the **5.1.0.25** or later level of the **bos.net.ipsec.rte** and **devices.pci.1410ff01.rte** filesets.

There is a limit on the number of security associations that can be offloaded to the network adapter on the receive side (inbound traffic). On the transmit side (outbound traffic), all packets that use a supported configuration are offloaded to the adapter. Some tunnel configurations can not be offloaded to the adapter.

The 10/100 Mbps Ethernet PCI Adapter II supports the following:

- DES, 3DES, or NULL encryption through ESP
- HMAC-MD5 or HMAC-SHA-1 authentication through ESP or AH, but not both. (If ESP and AH both used, ESP must be performed first. This is always true for IKE tunnels, but the user can select the order for manual tunnels.)
- Transport and Tunnel mode
- Offload of IPV4 packets

Note: The 10/100 Mbps Ethernet PCI Adapter II cannot handle packets with IP options.

To enable the 10/100 Mbps Ethernet PCI Adapter II for IP Security, you may have to detach the network interface and then enable the IPsec Offload feature.

To detach the network interface, do the following using the SMIT interface:

1. Login as the **root** user.
2. Type `smitty inet` at the command line and press Enter.
3. Select the **Remove a Network Interface** option and press Enter.
4. Select the network interface that corresponds to the 10/100 Mbps Ethernet PCI Adapter II and press Enter.

To enable the IPsec Offload feature, do the following using the SMIT interface:

1. Login as the **root** user.
2. Type `smitty eadap` at the command line and press Enter.
3. Select the **Change / Show Characteristics of an Ethernet Adapter** option and press Enter.
4. Select the 10/100 Mbps Ethernet PCI Adapter II and press Enter.
5. Change the **IPsec Offload** field to **yes** and press Enter.

To detach the network interface, from the command line, type the following:

```
# ifconfig enX detach
```

To enable the IPsec offload attribute, from the command line, type the following:

```
# chdev -l entX -a ipsec_offload=yes
```

To verify that the IPsec offload attribute has been enabled, from the command line, type the following:

```
# lsattr -El entX detach
```

To disable the IPsec offload attribute, from the command line, type the following:

```
# chdev -l entX -a ipsec_offload=no
```

Use the **enstat** command to ensure that your tunnel configuration is taking advantage of the IPsec offload attribute. The **enstat** command shows all the statistics of transmit and receive IPsec packets when the IPsec offload attribute is enabled. For example, if the Ethernet interface is *ent1*, type the following:

```
# entstat -d ent1
```

The output will be similar to the following:

```
.
.
.
10/100 Mbps Ethernet PCI Adapter II (1410ff01) Specific Statistics:
--------------------------------------------
.
.
.
Transmit IPsec packets: 3
Transmit IPsec packets dropped: 0
Receive IPsec packets: 2
Receive IPsec packets dropped: 0
```

Tunnels Versus Filters

Two distinct parts of IP Security are *tunnels* and *filters*. Tunnels require filters, but filters do not require tunnels.

- *Filtering* is a function in which incoming and outgoing packets can be accepted or denied based on a variety of characteristics called *rules*. This function allows a system administrator to configure the host

to control the traffic between this host and other hosts. Filtering is done on a variety of packet properties, such as source and destination addresses, IP Version (4 or 6), subnet masks, protocol, port, routing characteristics, fragmentation, interface, and tunnel definition. This filtering is done at the IP layer, so no changes are required to the applications.

- *Tunnels* define a security association between two hosts. These security associations involve specific security parameters that are shared between end points of the tunnel.

The following illustration indicates how a packet comes in from the network adapter to the IP stack. From there, the filter module is called to determine if the packet is permitted or denied. If a tunnel ID is specified, the packet is checked against the existing tunnel definitions. If the decapsulation from the tunnel is successful, the packet is passed to the upper-layer protocol. This function occurs in reverse order for outgoing packets. The tunnel relies on a filter rule to associate the packet with a particular tunnel, but the filtering function can occur without passing the packet to the tunnel.

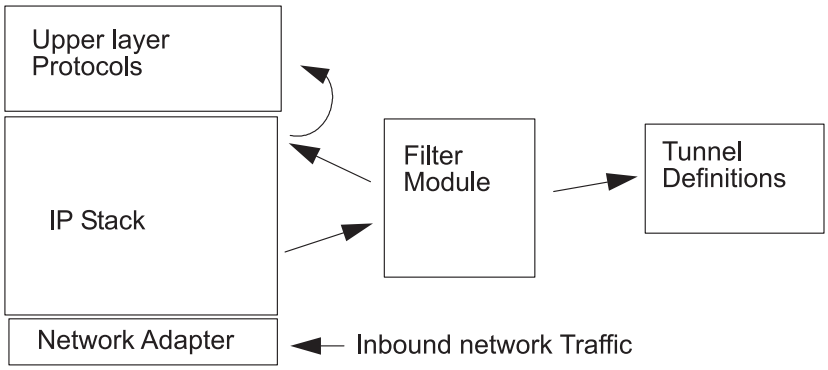
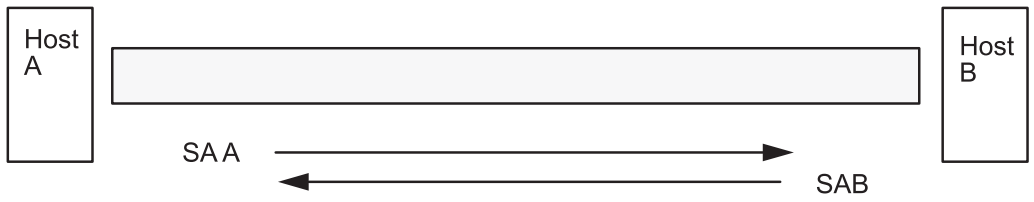


Figure 7. Network Packet Routing. This illustration shows the route a network packet takes. Inbound fron the network, the packet enters the network adapter. from there it goes to the IP stack where it is sent to the filter module. From the filter module, the packet is either sent to tunnel definitions or it is returned to the IP stack where it is forwarded to the upper-layer protocols.

Tunnels and Security Associations

Tunnels are used whenever you need to have data authenticated, or authenticated and encrypted. Tunnels are defined by specifying a security association between two hosts. The security association defines the parameters for the encryption and authentication algorithms and characteristics of the tunnel. The following illustration shows a virtual tunnel between Host A and Host B.



SA = Security Association, consisting of:

Destination address
SPI
Key
Crypto Algorithm and Format
Authentication Algorithm
Key Lifetime

Figure 8. Establishment of a Secure Tunnel Between Hosts A and B. This illustration shows a virtual tunnel running between Host A and Host B. Security association A is an arrow directed from Host A to Host B. Security association B is an arrow directed from Host B to Host A. A security association consists of the Destination Address, SPI, Key, Crypto Algorithm and Format, Authentication Algorithm, and Key Lifetime.

The Security Parameter Index (SPI) and the destination address identify a unique security association. These parameters are required for uniquely specifying a tunnel. Other parameters such as cryptographic algorithm, authentication algorithm, keys, and lifetime can be specified or defaults can be used.

Tunnel Considerations

IKE tunnels differ from manual tunnels because the configuration of security policies is a separate process from defining tunnel endpoints. In IKE, there is a two-step negotiation process. Each step of the negotiation process is called a *phase*, and each phase can have separate security policies.

When the Internet Key negotiation starts, it must set up a secure channel for the negotiations. This is known as the *key management* phase or *phase 1*. During this phase, each party uses preshared keys or digital certificates to authenticate the other and pass ID information. This phase sets up a security association during which the two parties determine how they plan to communicate securely and then which protections are to be used to communicate during the second phase. The result of this phase is an *IKE* or *phase 1* tunnel.

The second phase is known as the *data management* phase or *phase 2* and uses the IKE tunnel to create the security associations for AH and ESP that actually protect traffic. The second phase also determines the data that will be using the IP Security tunnel. For example, it can specify the following:

- A subnet mask
- An address range
- A protocol and port number combination

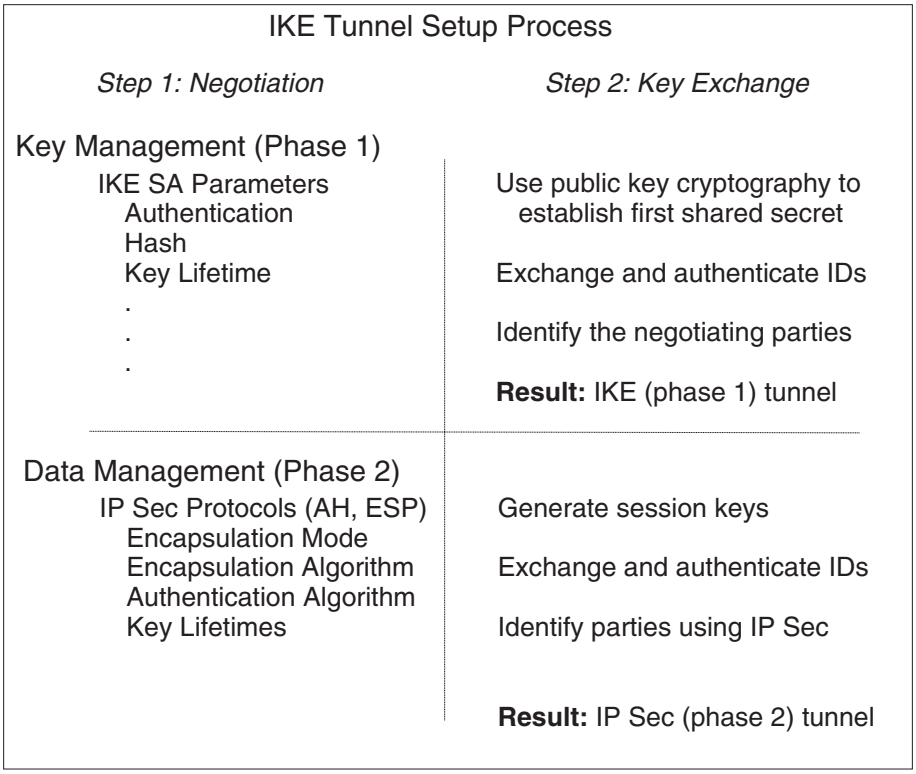


Figure 9. IKE Tunnel Setup Process. This illustration shows the two-step, two-phase process for setting up an IKE tunnel.

In many cases, the endpoints of the key management (IKE) tunnel will be the same as the endpoints of the data management (IP Security) tunnel. The IKE tunnel endpoints are the IDs of the machines carrying out the negotiation. The IP Security tunnel endpoints describe the type of traffic that will use the IP Security tunnel. For simple host-to-host tunnels, in which all traffic between two tunnels is protected with the same tunnel, the phase 1 and phase 2 tunnel endpoints are the same. When negotiating parties are two gateways, the IKE tunnel endpoints are the two gateways, and the IP Security tunnel endpoints are the machines or subnets (behind the gateways) or the range of addresses (behind the gateways) of the tunnel users.

Key Management Parameters and Policy

Phase 1 (the key management phase) sets the following parameters of an IKE tunnel configuration:

Key Management (Phase 1) Tunnel	Name of this IKE tunnel. For each tunnel, the endpoints of the negotiation must be specified. These are the two machines that plan to send and validate IKE messages. The name of the tunnel may describe the tunnel endpoints such as `VPN Boston` or `VPN Acme`.
Host Identity Type	ID type that will be used in the IKE exchange. The ID type and value must match the value for the preshared key to ensure that proper key lookup is performed. If a separate ID is used to search a preshared key value, the *host ID* is the key's ID and its *type* is KEY_ID. The KEY_ID type is useful if a single host has more than one preshared key value.
Host Identity	Value of the host ID represented as an IP address, a fully qualified domain name (FQDN), or a user at the fully qualified domain name (*user@FQDN*). For example, jdoe@studentmail.ut.edu.
IP Address	IP address of the remote host. This value is required when the host ID type is KEY_ID or whenever the host ID type cannot be resolved to an IP address. For example, if the user name cannot be resolved with a local nameserver, the IP address for the remote side must be entered.

You can customize key-management policy by specifying the parameters to be used during IKE negotiation. For example, there are key-management policies for preshared key or signature mode authentication. For Phase 1, the user must determine certain key-management security properties with which to carry out the exchange.

Data Management Parameters and Policy

The data management proposal parameters are set during phase 2 of an IKE tunnel configuration. They are the same IP Security parameters used in manual tunnels and describe the type of protection to be used for protecting data traffic in the tunnel. You can start more than one phase 2 tunnel under the same phase 1 tunnel.

The following endpoint ID types describe the type of data that uses the IP Security Data tunnel:

Host, **Subnet**, or **Range**	Describes whether the data traffic traveling in the tunnel will be for a particular host, subnet, or address range.
Host/Subnet ID	Contains the host or subnet identity of the local and remote systems passing traffic over this tunnel. Determines the IDs sent in the phase 2 negotiation and the filter rules that will be built if the negotiation is successful.
Subnet mask	Describes all IP addresses within the subnet (for example, host 9.53.250.96 and mask 255.255.255.0)
Starting IP Address Range	Provides the starting IP address for the range of addresses that will be using the tunnel (for example, 9.53.250.96 of 9.53.250.96 to 9.53.250.93)
Ending IP Address Range	Provides the ending IP address for the range of addresses that will be using the tunnel (for example, 9.53.250.93 of 9.53.250.96 to 9.53.250.93)
Port	Describes data using a specific port number (for example, 21 or 23)
Protocol	Describes data being transported with a specific protocol (for example, TCP or UDP). Determines the protocol sent in the phase 2 negotiation and the filter rules that will be built if the negotiation is successful. The protocol for the local endpoint must match the protocol for the remote end point.

Choosing a Tunnel Type

The decision to use manual tunnels or IKE tunnels depends on the tunnel support of the remote end and the type of key management desired. IKE tunnels are recommended (when available) because they offer industry-standard secure key negotiation and key refreshment. They also take advantage of the IETF ESP and AH header types and support anti-replay protection. You can optionally configure signature mode to allow digital certificates.

If the remote end uses one of the algorithms requiring manual tunnels, manual tunnels should be used. Manual tunnels ensure interoperability with a large number of hosts. Because the keys are static and difficult to change and might be cumbersome to update, they are not as secure. Manual tunnels can be used between a host running this operating system and any other machine running IP Security and having a common set of cryptographic and authentication algorithms. Most vendors offer Keyed MD5 with DES, or HMAC MD5 with DES. This subset works with almost all implementations of IP Security.

The procedure used in setting up manual tunnels, depends on whether you are setting up the first host of the tunnel or setting up the second host, which must have parameters matching the first host setup. When setting up the first host, the keys can be autogenerated, and the algorithms can be defaulted. When setting up the second host, import the tunnel information from the remote end, if possible.

Another important consideration is determining whether the remote system is behind a firewall. If it is, the setup must include information about the intervening firewall.

Using IKE with DHCP or Dynamically Assigned Addresses

One common scenario for using IP Security with an operating system is when remote systems are initiating IKE sessions with a server and their identity cannot be tied to a particular IP address. This case can occur in a Local Area Network (LAN) environment, such as using IP Security to connect to a server on a LAN and wanting to encrypt the data. Other common uses involve remote clients dialing into a server, and using either a fully qualified domain name (FQDN) or e-mail address (*user@FQDN*) to identify the remote ID.

In order to make a policy decision based on explicit information about the remote identity, aggressive mode must be used. In this case, the identity is sent in the first message of the negotiation and can be used to do a policy lookup on the security policy database. This will ensure that only specifically named remote identities will be able to negotiate using the IKE protocol.

For the Data Management phase (Phase 2), when the IP Security associations are being created to encrypt TCP or UDP traffic, a generic data manager tunnel can be configured. Therefore, any request that was authenticated during phase 1, will use the generic tunnel for defined Data Management phase if the IP address is not explicitly configured in the database. This allows any address to match the generic tunnel and can be used as long as the rigorous public key-based security validation was successful in phase 1.

Using XML to Define a Generic Data Management Tunnel

You can define a generic Data Management tunnel using the XML format understood by **ikedb**. Generic Data Management tunnels are used with DHCP. The XML format uses the tag name IPSecTunnel for what Web-based System Manager calls a Data Management tunnel. This is also referred to as a *phase 2 tunnel* in other contexts. A *generic Data Management tunnel* is not a true tunnel, but an **IPSecProtection** that is used if an incoming Data Management message (under a specific Key Management tunnel) does not match any Data Management tunnel defined for that Key Management tunnel. It is only used in the case where the AIX system is the responder. Specifying a generic Data Management tunnel **IPSecProtection** is optional.

The generic Data Management tunnel is defined in the **IKEProtection** element. There are two XML attributes, called **IKE_IPSecDefaultProtectionRef** and **IKE_IPSecDefaultAllowedTypes**, that are used for this.

First, you need to define an **IPSecProtection** that you would like to use as the default if no **IPSecTunnels** (Data Management tunnels) match. An **IPSecProtection** that is to be used as a default must have an **IPSec_ProtectionName** that begins with `_defIPSprot_`.

Now go to the **IKEProtection** that you would like to use this default **IPSecProtection**. Specify an **IKE_IPSecDefaultProtectionRef** attribute that contains the name of the default **IPSec_Protection**.

You must also specify a value for the **IKE_IPSecDefaultAllowedTypes** attribute in this **IKEProtection**. It can have one or more of the following values (if multiple values, they should be space-separated):

```
Local_IPV4_Address
Local_IPV6_Address
Local_IPV4_Subnet
Local_IPV6_Subnet
Local_IPV4_Address_Range
Local_IPV6_Address_Range
Remote_IPV4_Address
Remote_IPV6_Address
Remote_IPV4_Subnet
Remote_IPV6_Subnet
Remote_IPV4_Address_Range
Remote_IPV6_Address_Range
```

These values correspond to the ID types specified by the initiator. In the IKE negotiation, the actual IDs are ignored. The specified **IPSecProtection** is used if the **IKE_IPSecDefaultAllowedTypes** attribute contains a string beginning with `Local_` that corresponds to the initiator's local ID type, and contains a string beginning with `Remote_` that corresponds to the initiator's remote ID type. In other words, you must have at least one **Local**_ value and at least one **Remote**_ value in any **IKE_IPSecDefaultAllowedTypes** attribute in order for the corresponding **IPSec_Protection** to be used.

Example: An initiator sends the following to the AIX system in a phase 2 (Data Management) message:

```
local ID type:    IPV4_Address
local ID:         192.168.100.104

remote ID type:   IPV4_Subnet
remote ID:        10.10.10.2
remote netmask:   255.255.255.192
```

The AIX system does not have a Data Management tunnel matching these IDs. But it does have an **IPSecProtection** with the following attributes defined:

```
IKE_IPSecDefaultProtectionRef="_defIPSprot_protection4"
IKE_IPSecDefaultAllowedTypes="Local_IPV4_Address
                              Remote_IPV4_Address
                              Remote_IPV4_Subnet
                              Remote_IPV4_Address_Range"
```

The local ID type of the incoming message, **IPV4_Address**, matches one of the **Local**_ values of the allowed types, **Local_IPV4_Address**. Also, the remote ID of the message, **IPV4_Subnet**, matches the value **Remote_IPV4_Subnet**. Therefore the Data Management tunnel negotiation will proceed with `_defIPSprot_protection4` as the **IPSecProtection**.

The **/usr/samples/ipsec/default_p2_policy.xml** file is a full XML file defining a generic **IPSecProtection** that can be used as an example.

Configuring Internet Key Exchange Tunnels

This section provides information on how to configure Internet Key Exchange (IKE) tunnels using the Web-based System Manager interface, the System Management Interface Tool (SMIT), or the command line.

Using Web-based System Manager to Configure IKE Tunnels

The "Using the Basic Configuration Wizard" provides an easy way to define an IKE tunnel with preshared keys. For more advanced options, see "Advanced IKE Tunnel Configuration".

Using the Basic Configuration Wizard

You can define an IKE tunnel through Web-based System Manager using preshared keys or certificates as the authentication method. The Web-based System Manager adds new key management and data management IKE tunnels to the IP Security subsystem, allows you to input minimal data and choose some options, and makes use of common default values for such parameters as tunnel lifetime.

When using the basic configuration wizard, keep the following in mind:

- The wizard can be used only for initial tunnel configuration. To modify, delete, or activate a tunnel, use the **IKE Tunnel** plug-in or task bar.
- The tunnel name must be unique on your system, but you can use the same name on the remote system. For example, on the local and remote systems, the tunnel name can be *hostA_to_hostB*, but the Local IP Address and the Remote IP Address fields (endpoints) are switched.
- Phase 1 and phase 2 tunnels are defined with the same encryption and authentication algorithms.
- The preshared key must be entered in hexadecimal form (without a leading `0x`) or ASCII text.
- If digital certificates are chosen as the authentication method, you must use the Key Manager tool to create a digital certificate.
- The host ID type can only be `IP Address`.
- The transforms and proposals you create are assigned names that end with the user-defined tunnel name. You can view the transforms and proposals in Web-based System Manager through **VPN** and the **IKE Tunnel** plug-in.

Use the following procedure to configure a new tunnel using the wizard:

1. Open Web-based System Manager using the **wsm** command from the command line.
2. Select the Network plug-in.
3. Select **Virtual Private Networks (IP Security)**.
4. From the Console area, select the **Overview and Tasks** folder.
5. Select **Configure a Basic Tunnel Configuration wizard**.
6. Click on **Next** on the Step 1 Introduction panel, then follow the steps to configure an IKE tunnel.

   Online help is available if you need it.

   After a tunnel is defined using the wizard, the tunnel definition displays in the Web-based System Manager IKE tunnels list and can be activated or modified.

Advanced IKE Tunnel Configuration

You can configure key management and data management tunnels separately, using the following procedures.

Configuring Key Management Tunnels: IKE tunnels are configured using Web-based System Manager. Use the following procedure to add a key management tunnel:

1. Open Web-based System Manager using the **wsm** command.
2. Select the Network plug-in.
3. Select **Virtual Private Networks (IP Security)**.

4. From the Console area, select **Overview and Tasks**.
5. Select **Start IP Security**. This action loads the IP Security kernel extensions and starts the **isakmpd**, **tmd**, and **cpsd** daemons.

   A tunnel is created by defining the key management and data management endpoints and their associated security transforms and proposals.

   - Key management is the authentication phase. It sets up a secure channel between the negotiating parties needed before the final IP Security parameters and keys are computed.
   - Data management describes the type of traffic that will be using a particular tunnel. It can be configured for a single host or group of hosts (with the use of subnets or IP ranges) along with specified protocol and port numbers.

   The same key management tunnel can be used to protect multiple data management negotiations and key refreshes, as long as they take place between the same two endpoints; for example, between two gateways.
6. To define the key management tunnel endpoints, click **Internet Key Exchange (IKE) Tunnels** on the Identification tab.
7. Enter information to describe the identities of the systems taking part in the negotiations. In most cases, IP addresses are used, and a policy compatible with the remote side must be created.

   On the Transforms tab, use matching transforms on both sides, or contact the administrator on the remote end to define a matching transform. A transform containing several choices can be created to allow flexibility when proposing or matching on a transform.
8. If using preshared keys for authentication, enter the preshared key under the **key** tab. This value must match on both the remote and local machines.
9. Create a transform to be associated with this tunnel by using the **Add** button on the Transforms tab.

   To enable digital certificates and signature mode support, choose an authentication method of **RSA Signature** or **RSA Signature with CRL Checking**.

   For more information about digital certificates, see “Working with Digital Certificates and the Key Manager” on page 153.

Configuring Data Management Tunnels: To set up data management tunnel endpoints and proposals and to complete IKE tunnel setup, open Web-based System Manager, as described in “Configuring Key Management Tunnels” on page 147. A data management tunnel is created by doing the following:

1. Select a key management tunnel and define any unique options. Most data management options can remain as defined by the default.
2. Specify endpoint types (such as IP address, subnet, or IP address range) under the Endpoints tab. You can select a port number and protocol or accept the default.
3. On the Proposals panel, you can create a new proposal by clicking the **Add** button or clicking **OK** to create a proposal. If there are multiple proposals, you can use the Move Up or Move Down buttons to change the search order.

Group Support: Beginning with AIX 5.1, IP security supports grouping IKE IDs in a tunnel definition to associate multiple IDs with a single security policy without having to create separate tunnel definitions. Grouping is especially useful when setting up connections to several remote hosts, because you can avoid setting up or managing multiple tunnel definitions. Also, if changes must be made to a security policy, you do not need to change multiple tunnel definitions.

A group must be defined before using that group name in tunnel definition. The group’s size is limited to 1 Kbyte. A group name can be used in both key management tunnel and data management tunnel definitions, but it can be used only as a remote ID.

A group is composed of a group name and a list of IKE IDs and ID types. The IDs can all be the same type or a mix of the following:

- IPv4 addresses
- IPv6 addresses
- FQDN
- user@FQDN
- X500 DN types.

During a Security Association negotiation, the IDs in a group are searched linearly for the first match.

Web-based System Manager can be used to define a group that is to be used for the remote endpoint of a Key Management tunnel. Refer to the “Command Line Interface for IKE Tunnel Configuration” section for information on defining groups from the command line. To define a group using Web-based System Manager, use the following procedure:

1. Selecting a Key Management tunnel in the **IKE Tunnel** container.
2. Open the **properties** dialog.
3. Select the **Identification** tab.
4. Select **group ID definition** for the remote host identity type.
5. Select the **Configure Group Definition** button and enter the group members in the window.

Using the SMIT Interface for IKE Tunnel Configuration

You can use the SMIT interface to configure IKE tunnels and perform basic IKE database functions. SMIT uses underlying XML command functions to perform additions, deletions, and modifications to the IKE tunnel definitions. IKE SMIT is used in configuring IKE tunnels quickly and provides examples of the XML syntax used to create IKE tunnel definitions. The IKE SMIT menus also allow you to back up, restore, and initialize the IKE database.

To configure an IPv4 IKE tunnel, use the **smitty ike4** fast path. To configure an IPv6 IKE tunnel, use the **smitty ike6** fast path. The IKE database functions can be found in the Advanced IP Security Configuration menu.

All IKE database entries added through SMIT can be viewed or modified through the Web-based System Manager tool.

Command Line Interface for IKE Tunnel Configuration

The **ikedb** command, available in AIX 5.1 and later, allows a user to retrieve, update, delete, import, and export information in the IKE database. using an XML interface. The **ikedb** command allows the user to write to (put) or read from (get) the IKE database. The input and output format is an Extensible Markup Language (XML) file. The format of an XML file is specified by its Document Type Definition (DTD). The **ikedb** command allows the user to see the DTD that is used to validate the XML file when doing a put. While entity declarations can be added to the DTD using the **-e** flag, this is the only modification to the DTD that can be made. Any external DOCTYPE declaration in the input XML file will be ignored and any internal DOCTYPE declaration might result in an error. The rules followed to parse the XML file using the DTD are specified in the XML standard. The **/usr/samples/ipsec** file has a sample of a typical XML file that defines common tunnel scenarios. See the **ikedb** command description in the *AIX 5L Version 5.2 Commands Reference* for syntax details.

You can use the **ike** command to start, stop, and monitor IKE tunnels. The **ike** command can also be used to activate, remove, or list IKE and IP Security tunnels. See the **ike** command description in the *AIX 5L Version 5.2 Commands Reference* for syntax details.

The following examples show how to use **ike**, **ikedb**, and several other commands to configure and check the status of your IKE tunnel:

1. To start a tunnel negotiation (*activate* a tunnel) or to allow the incoming system to act as a responder (depending on the role that is specified), use the **ike** command with a tunnel number, as follows:

```
# ike cmd=activate numlist=1
```

You can also use remote id or IP addresses, as shown in the following examples:

```
# ike cmd=activate remid=9.3.97.256
# ike cmd=activate ipaddr=9.3.97.100, 9.3.97.256
```

Since it may take several seconds for the commands to complete, the command returns after the negotiation is started.

2. To display the tunnel status, use the **ike** command, as follows:

```
# ike cmd=list
```

The output looks similar to the following:

```
Phase 1 Tunnel ID       [1]
Phase 2 Tunnel ID       [1]
```

The output shows phase 1 and phase 2 tunnels that are currently active.

3. To get a verbose listing of the tunnel, use the **ike** command, as follows:

```
# ike cmd=list verbose
```

The output looks similar to the following:

```
Phase 1 Tunnel ID       1
Local ID Type:          Fully_Qualified_Domain_Name
Local ID:               bee.austin.ibm.com
Remote ID Type:         Fully_Qualified_Domain_Name
Remote ID:              ipsec.austin.ibm.com
Mode:                   Aggressive
Security Policy:        BOTH_AGGR_3DES_MD5
Role:                   Initiator
Encryption Alg:         3DES-CBC
Auth Alg:               Preshared Key
Hash Alg:               MD5
Key Lifetime:           28800 Seconds
Key Lifesize:           0 Kbytes
Key Rem Lifetime:       28737 Seconds
Key Rem Lifesize:       0 Kbytes
Key Refresh Overlap:    5%
Tunnel Lifetime:        2592000 Seconds
Tunnel Lifesize:        0 Kbytes
Tun Rem Lifetime:       2591937 Seconds
Status:                 Active

Phase 2 Tunnel ID       1
Local ID Type:          IPv4_Address
Local ID:               10.10.10.1
Local Subnet Mask:      N/A
Local Port:             any
Local Protocol:         all
Remote ID Type:         IPv4_Address
Remote ID:              10.10.10.4
Remote Subnet Mask:     N/A
Remote Port:            any
Remote Portocol:        all
Mode:                   Oakley_quick
Security Policy:        ESP_3DES_MD5_SHA_TUNNEL_NO_PFS
Role:                   Initiator
Encryption Alg:         ESP_3DES
AH Transform:           N/A
Auth Alg:               HMAC-MD5
PFS:                    No
SA Lifetime:            600 Seconds
SA Lifesize:            0 Kbytes
SA Rem Lifetime:        562 Seconds
```

```
SA Rem Lifesize:        0 Kbytes
Key Refresh Overlap:    15%
Tunnel Lifetime:        2592000 Seconds
Tunnel Lifesize:        0 Kbytes
Tun Rem Lifetime:       2591962 Seconds
Assoc P1 Tunnel:        0
Encap Mode:             ESP_tunnel
Status:                 Active
```

4. To display the filter rules in the dynamic filter table for the newly activated IKE tunnel, use the **lsfilt** command, as follows:

   ```
   # lsfilt -d
   ```

   The output looks similiar to the following:

   ```
   1 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 no udp eq 4001 eq 4001 both both no all
     packets 0 all
   2 *** Dynamic filter placement rule *** no
   0 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 yes all any 0 any 0 both both no all
     packets 0 all

   *** Dynamic table ***

   0 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 no udp eq 500 eq 500 local both no all
     packets 0
   0 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 no ah any 0 any 0 both inbound no all
     packets 0
   0 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 no esp any 0 any 0 both inbound no all
     packets 0
   1 permit 10.10.10.1 255.255.255.255 10.10.10.4 255.255.255.255 no all any 0 any
     0 both outbound yes all packets 1
   1 permit 10.10.10.4 255.255.255.255 10.10.10.1 255.255.255.255 no all any 0 any
     0 both inbound yes all packets 1
   ```

   This example shows a machine that has one IKE tunnel and no other tunnels. The dynamic filter placement rule (rule #2 in this example output of the static table) can be moved by the user to control placement relative to all other user-defined rules. The rules in the dynamic table are constructed automatically as tunnels are negotiated and corresponding rules are inserted into the filter table. These rules can be displayed, but not edited.

5. To turn on logging of the dynamic filter rules, set the logging option for rule #2 to yes, use the **chfilt** command, as shown in the following example:

   ```
   # chfilt -v 4 -n 2 -l y
   ```

   For more details on logging of IKE traffic, see “Logging Facilities” on page 172.

6. To deactivate the tunnel, use the **ike** command, as follows:

   ```
   # ike cmd=remove numlist=1
   ```

7. To view tunnel definitions, use the **ikedb** command, as follows:

   ```
   # ikedb -g
   ```

8. To put definitions to the IKE database from an XML file that has been generated on a peer machine and overwrite any existing objects in the database with the same name, use the **ikedb** command, as follows:

   ```
   # ikedb -pFs peer_tunnel_conf.xml
   ```

   The **peer_tunnel_conf.xml** is the XML file generated on a peer machine.

9. To get the definition of the phase 1 tunnel named *tunnel_sys1_and_sys2* and all dependent phase 2 tunnels with respective proposals and protections, use the **ikedb** command, as follows:

   ```
   # ikedb -gr -t IKETunnel -n tunnel_sys1_and_sys2
   ```

10. To delete all preshared keys from the database, use the **ikedb** command, as follows:

    ```
    # ikedb -d -t IKEPresharedKey
    ```

For general information on IKE tunnel group support, see the “Group Support” on page 148 section. You can use the **ikedb** command to define groups from the command line.

AIX IKE and Linux Affinity

To configure an AIX IKE tunnel using Linux configuration files (AIX 5.1 and later), use the **ikedb** command with the **-c** flag (conversion option), which lets you use the **/etc/ipsec.conf** and **/etc/ipsec.secrets** Linux configuration files as IKE tunnel definitions. The **ikedb** command parses the Linux configuration files, creates an XML file, and optionally adds the XML tunnel definitions to the IKE database. You can then view the tunnel definitions by using either the **ikedb -g** command or the Web-based System Manager.

IKE Tunnel Configuration Scenarios

The following scenarios describe the type of situations most customers encounter when trying to set up tunnels. These scenarios can be described as the branch office, business partner, and remote access cases.

- In the branch office case, the customer has two trusted networks that they want to connect together—the engineering group of one location to the engineering group of another. In this example, there are gateways that connect to each other and all the traffic passing between the gateways use the same tunnel. The traffic at either end of the tunnel is decapsulated and passes in the clear within the company intranet.

  In the first phase of the IKE negotiation, the IKE security association is created between the two gateways. The traffic that passes in the IP Security tunnel is the traffic between the two subnets, and the subnet IDs are used in the phase 2 negotiation. After the security policy and tunnel parameters are entered for the tunnel, a tunnel number is created. Use the **ike** command to start the tunnel.
- In the business partner scenario, the networks are not trusted, and the network administrator may want to restrict access to a smaller number of hosts behind the security gateway. In this case, the tunnel between the hosts carries traffic protected by IP Security for use between two particular hosts. The protocol of the phase 2 tunnel is AH or ESP. This host-to-host tunnel is secured within a gateway-to-gateway tunnel.
- In the remote access case, the tunnels are set up on demand and a high level of security is applied. The IP addresses may not be meaningful, therefore, fully qualified domain names or *user@* fully qualified domain names are preferred. Optionally, you can use KEYID to relate a key to a host ID.

Working with Digital Certificates and the Key Manager

Digital certificates bind an identity to a public key, through which you can verify the sender or the recipient of an encrypted transfer. Beginning with AIX 4.3.3, IP Security uses digital certificates to enable *public-key cryptography*, also known as *asymmetric cryptography*, which encrypts data using a private key known only to the user and decrypts it using an associated public (shared) key from a given public-private key pair. *Key pairs* are long strings of data that act as keys to a user's encryption scheme.

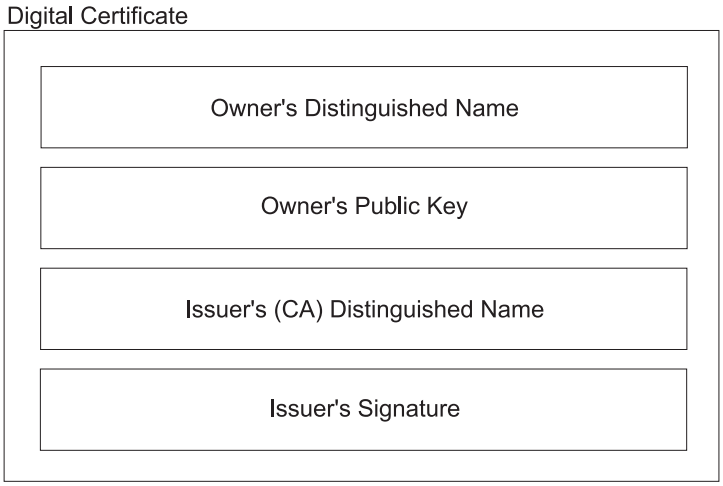
In public-key cryptography, the public key is given to anyone with whom the user wants to communicate. The sender digitally signs all secure communications with the corresponding private key for their assigned key pair. The recipient uses the public key to verify the sender's signature. If the message is successfully decrypted with the public key, the receiver can verify that the sender was authenticated.

Public-key cryptography relies on trusted, third parties, known as a *certification authorities* (*CAs*), to issue reliable digital certificates. The recipient specifies which issuing organizations or authorities are deemed trusted. A certificate is issued for a specific amount of time; when its expiration date has passed, it must be replaced.

AIX 4.3.3 and later versions provide the Key Manager tool, which manages digital certificates. The following sections provide conceptual information about the certificates themselves. Management tasks for these certificates are described in "Working with Digital Certificates and the Key Manager".

Format of Digital Certificates

The digital certificate contains specific pieces of information about the identity of the certificate owner and about the certification authority. See the following figure for an illustration of a digital certificate.



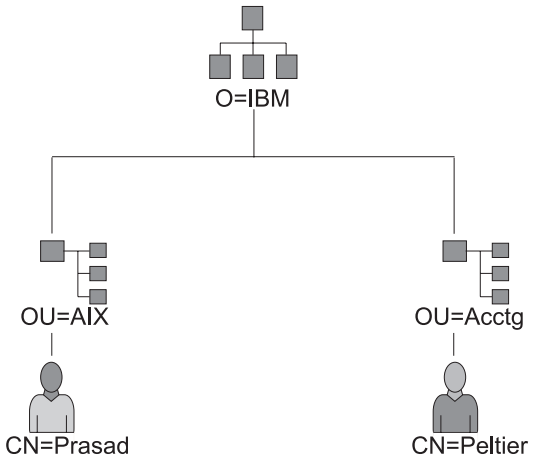
Contents of a Digital Certificate

Figure 10. Contents of a Digital Certificate. This illustration shows the four entities of a digital certificate. From the top they are, Owner's Distinguished Name, Owners Public Key, Issuer's (CA) Distinguished Name, and Issuer's Signature.

The following list further describes the contents of the digital certificate:

Owner's Distinguished Name
: Combination of the owner's common name and context (position) in the directory tree. In the following figure of a simple directory tree, for example, Prasad is the owner's common name and the context is country=US, organization=ABC, lower organization=SERV; therefore, the distinguished name is:

```
/C=US/O=ABC/OU=SERV/CN=prasad.austin.ibm.com
```



Example of Deriving Distinguished Name from Directory Tree

Figure 11. Example of Deriving Distinguished Name from Directory Tree. This illustration is a directory tree with O=ABC at the top level and branching to two entities on the second level. Level two contains OU=AIX and OU=Acctg on separate branches; each has a branch leading to a single entity on the last level.. The last level contains CN=Prasad and CN=Peltier respectively.

Owner's Public Key
Used by the recipients to decrypt data.

Subject Alternate Name
Can be an identifier such as an IP address, e-mail address, fully qualified domain name, and so on.

Issue Date
Date the digital certificate was issued.

Expiration Date
Date the digital certificate expires.

Issuer's Distinguished Name
Distinguished name of the Certification Authority.

Issuer's Digital Signature
Digital signature used to validate a certificate.

Security Considerations for Digital Certificates

A digital certificate alone cannot prove identity. The digital certificate only allows you to verify the identity of the digital certificate owner by providing the public key that is needed to check the owner's digital signature. You can safely send your public key to another because your data cannot be decrypted without the other part of the key pair, your private key. Therefore, the owner must protect the private key that belongs to the public key in the digital certificate. All communications of the owner of a digital certificate can be deciphered, if the private key is known. Without the private key, a digital certificate cannot be misused.

Certification Authorities and Trust Hierarchies

A digital certificate is only as trustworthy as the certification authority (CA) that issued it. As part of this trust, the policies under which certificates are issued should be understood. Each organization or user must determine which certification authorities can be accepted as trustworthy.

The Key Manager tool also allows organizations to create self-signed certificates, which can be useful for testing or in environments with a small number of users or machines.

As a user of a security service, you need to know its public key to obtain and validate any digital certificates. Also, simply receiving a digital certificate does not assure its authenticity. To verify its authenticity, you need the public key of the certification authority that issued that digital certificate. If you do not already hold an assured copy of the CA's public key, then you might need an additional digital certificate to obtain the CA's public key.

Certificate Revocation Lists (CRLs)

A digital certificate is expected to be used for its entire validity period. If needed, however, a certificate can be invalidated before its actual date of expiration. Invalidating the certificate may be necessary, for example, if an employee leaves the company or if the certificate's private key has been compromised. To invalidate a certificate, you must notify the appropriate Certificate Authority (CA) of the circumstances. When a CA revokes a certificate, it adds the invalid certificate serial number to a Certificate Revocation List (CRL).

CRLs are signed data structures that are issued periodically and made available in a public repository. CRLs can be retrieved from HTTP or LDAP servers. Each CRL contains a current time stamp and a **nextUpdate** time stamp. Each revoked certificate in the list is identified by its certificate serial number.

When configuring an IKE tunnel and using digital certificates as your authentication method, you can confirm the certificate has not been revoked by selecting RSA Signature with CRL Checking. If CRL Checking is enabled, the list is located and checked during the negotiation process to establish the key management tunnel.

> **Note:** To use this feature of IP Security, your system must be configured to use a SOCKS server (version 4 for HTTP servers), an LDAP server, or both. If you know which SOCKS or LDAP server you are using to obtain CRLs, you can make the necessary configuration selections by using Web-based System Manager. Select **CRL Configuration** from the Digital Certificates menu.

Uses for Digital Certificates in Internet Applications

Internet applications that use public-key cryptography systems must use digital certificates to obtain the public keys. There are many applications that use public-key cryptography, including the following ones:

Virtual Private Networks (VPN)

Virtual Private Networks, also called *secure tunnels*, can be set up between systems such as firewalls to enable protected connections between secure networks over unsecured communication links. All traffic destined to these networks is encrypted between the participating systems.

The protocols used in tunneling follow the IP Security and IKE standards, which allow for a secure, encrypted connection between a remote client (for example, an employee working from home) and a secure host or network.

Secure Sockets Layer (SSL)

SSL is a protocol that provides privacy and integrity for communications. It is used by Web servers for secure connections between Web servers and Web browsers, by the Lightweight Directory Access Protocol (LDAP) for secure connections between LDAP clients and LDAP servers, and by Host-on-Demand V.2 for connections between the client and the host system. SSL uses digital certificates for key exchange, server authentication, and, optionally, client authentication.

Secure Electronic Mail

Many electronic mail systems, using standards such as PEM or S/MIME for secure electronic mail, use digital certificates for digital signatures and for the exchange of keys to encrypt and decrypt mail messages.

Digital Certificates and Certificate Requests

A signed digital certificate contains fields for the owner's distinguished name, the owner's public key, the CA's distinguished name and the CA's signature. A self-signed digital certificate contains its owner's distinguished name, public key, and signature.

A *certificate request* must be created and sent to a CA to request a digital certificate. The certificate request contains fields for the requestor's distinguished name, public key, and signature. The CA verifies the requestor's signature with the public key in the digital certificate to ensure that:

- The certificate request was not modified in transit between the requestor and the CA.
- The requestor possesses the corresponding private key for the public key that is in the certificate request.

The CA is also responsible for verifying to some level the identity of the requestor. Requirements for this verification can range from very little proof to absolute assurance of the owner's identity.

The Key Manager Tool

The Key Manager tool manages digital certificates, and is located in the **gskkm.rte** file set on the Expansion Pack.

This section describes how to use Key Manager to do the following tasks:

1. Creating a Key Database
2. Adding a CA Root Digital Certificate
3. Establishing Trust Settings
4. Deleting a CA Root Digital Certificate
5. Requesting a Digital Certificate
6. Adding (Receiving) a New Digital Certificate
7. Deleting a Digital Certificate
8. Changing a Database Password
9. Creating IKE Tunnels using Digital Certificates

To set up digital certificates and signature support, at minimum you must do tasks 1, 2, 3, 4, 6, and 7. Then, use Web-based System Manager to create an IKE tunnel and associate a policy with the tunnel that uses RSA Signature as the authentication method.

You can create and configure a key database from the Web-based System Manager VPN Overview window by selecting the **Managing Digital Certificates** option, or by using the **certmgr** command to open the Key Manager tool from the command line.

Creating a Key Database

A key database enables VPN endpoints to connect using valid digital certificates. The key database (*.kdb) format is used with IP Security VPNs.

The following types of CA digital certificates are provided with Key Manager:

- RSA Secure Server Certification Authority
- Thawte Personal Premium Certification Authority
- Thawte Personal Freemail Certification Authority
- Thawte Personal Basic Certification Authority
- Thawte Personal Server Certification Authority
- Thawte Server Certification Authority
- Verisign Class 1 Public Primary Certification Authority

- Verisign Class 2 Public Primary Certification Authority
- Verisign Class 3 Public Primary Certification Authority
- Verisign Class 4 Public Primary Certification Authority

These signature digital certificates enable clients to attach to servers that have valid digital certificates from these signers. After you create a key database, you can use it as created to attach to a server that has a valid digital certificate from one of the signers.

To use a signature digital certificate that is not on this list, you must request it from the CA and add it to your key database. See “Adding a CA Root Digital Certificate”.

To create a key database using the **certmgr** command, use the following procedure:

1. Start the Key Manager tool by typing:
   ```
   # certmgr
   ```
2. Select **New** from the Key Database File pull down menu.
3. Accept the default value, **CMS key database file**, for the **Key database type** field.
4. Enter the following file name in the **File Name** field:
   ```
   ikekey.kdb
   ```
5. Enter the following location of the database in the **Location** field:
   ```
   /etc/security
   ```

   Note: The key database must be named **ikekey.kbd** and it must be placed in the **/etc/security** directory. Otherwise, IP Security cannot function correctly.
6. Click **OK**. The **Password Prompt** screen is displayed.
7. Enter a password in the **Password** field, and enter it again in the **Confirm Password** field.
8. If you want to change the number of days until the password expires, enter the desired number of days in the **Set expiration time?** field. The default value for this field is 60 days. If you do not want the password to expire, clear the **Set expiration time?** field.
9. To save an encrypted version of the password in a stash file, select the **Stash the password to a file?** field and enter **yes**.

   Note: You must stash the password to enable the use of digital certificates with IP Security.
10. Click **OK**. A confirmation screen displays, verifying that you have created a key database.
11. Click **OK** again and you return to the IBM Key Management screen. You can either perform other tasks or exit the tool.

Adding a CA Root Digital Certificate

After you have requested and received a root digital certificate from a CA, you can add it to your database. Most root digital certificates are of the form *.arm, such as the following:

```
cert.arm
```

To add a CA root digital certificate to a database, use the following procedure:

1. Unless you are already using Key Manager, start the tool by typing:
   ```
   # certmgr
   ```
2. From the main screen, select **Open** from the Key Database File pull down menu.
3. Highlight the key database file to which you want to add a CA root digital certificate and click **Open**.
4. Enter the password and click **OK**. When your password is accepted, you are returned to the IBM Key Management screen. The title bar now shows the name of the key database file you selected, indicating that the file is now open and ready to be worked with.
5. Select **Signer Certificates** from the Personal/Signer Certificates pull down menu.

6. Click **Add**.
7. Select a data type from the Data type pull down menu, such as:

   `Base64-encoded ASCII data`
8. Enter a certificate file name and location for the CA root digital certificate, or click **Browse** to select the name and location.
9. Click **OK**.
10. Enter a label for the CA root digital certificate, such as `Test CA Root Certificate`, and click **OK**. You are returned to the `Key Management` screen. The **Signer Certificates** field now shows the label of the CA root digital certificate you just added. You can either perform more tasks or exit the tool.

Establishing Trust Settings

Installed CA certificates are set to **trusted** by default. To change the trust setting, do the following:

1. Unless you are already using Key Manager, start the tool by typing:

   `# certmgr`
2. From the main screen, select **Open** from the Key Database File pull down menu.
3. Highlight the key database file in which you want to change the default digital certificate and click **Open**.
4. Enter the password and click **OK**. After your password is accepted, you are returned to the IBM Key Management screen. The title bar shows the name of the key database file you selected, indicating that the file is now open.
5. Select **Signer Certificates** from the Personal/Signer Certificates pull down menu.
6. Highlight the certificate you want to change and click **View/Edit**, or double-click on the entry. The Key Information screen is displayed for the certificate entry.
7. To make this certificate a trusted root certificate, select the box next to **Set the certificate as a trusted root** and click **OK**. If the certificate is not trusted, clear the check box instead and click **OK**.
8. Click **OK** from the Signer Certificates screen. You are returned to the IBM Key Management screen. You can either perform other tasks or exit the tool.

Deleting a CA Root Digital Certificate

If you no longer want to support one of the CAs in your signature digital certificate list, you must delete the CA root digital certificate.

> **Note:** Before deleting a CA root digital certificate, create a backup copy in case you later want to re-create the CA root.

To delete a CA root digital certificate from a database, use the following procedure:

1. Unless you are already using Key Manager, start the tool by typing:

   `# certmgr`
2. From the main screen, select **Open** from the `Key Database File` pull down menu.
3. Highlight the key database file from which you want to delete a CA root digital certificate and click **Open**.
4. Enter the password and click **OK**. After your password is accepted, you are returned to the **Key Management** screen. The title bar shows the name of the key database file you selected, indicating that the file is now open and ready to be edited.
5. Select **Signer Certificates** from the `Personal/Signer Certificates` pull down menu.
6. Highlight the certificate you want to delete and click **Delete**. The `Confirm` screen is displayed.
7. Click **Yes**. You are returned to the IBM Key Management screen. The label of the CA root digital certificate no longer appears in the **Signer Certificates** field. You can either perform other tasks or exit the tool.

Requesting a Digital Certificate

To acquire a digital certificate, generate a request using Key Manager and submit the request to a CA. The request file you generate is in the PKCS#10 format. The CA then verifies your identity and sends you a digital certificate.

To request a digital certificate, use the following procedure:

1. Unless you are already using Key Manager, start the tool by typing:
   ```
   # certmgr
   ```
2. From the main screen, select **Open** from the Key Database File pull down menu.
3. Highlight the **/etc/security/ikekey.kdb** key database file from which you want to generate the request and click **Open**.
4. Enter the password and click **OK**. After your password is accepted, you are returned to the IBM Key Management screen. The title bar shows the name of the key database file you selected, indicating that the file is now open and ready to be edited.
5. Select **Personal Certificate Requests** from the Personal/Signer Certificates pull down menu (in AIX Version 4) or select **Create —> New Certificate Request** (beginning in AIX 5.1).
6. Click **New**.
7. From the following screen, enter a **Key Label** for the self-signed digital certificate, such as:
   ```
   keytest
   ```
8. Enter a **Common Name** (the default is the host name) and **Organization**, and then select a **Country**. For the remaining fields, either accept the default values, or choose new values.
9. Define the **Subject Alternate** name. The optional fields associated with **Subject Alternate** are email address, IP address, and DNS name. For a tunnel type of IP address, type the same IP address that is configured in the IKE tunnel into the IP address field. For a tunnel ID type of *user@FQDN*, complete the email address field. For a tunnel ID type of FQDN, type a fully qualified domain name (for example, *hostname.companyname*.com) in the DNS name field.
10. At the bottom of the screen, enter a name for the file, such as:
    ```
    certreq.arm
    ```
11. Click **OK**. A confirmation screen is displayed, verifying that you have created a request for a new digital certificate.
12. Click **OK**. You are returned to the IBM Key Management screen. The **Personal Certificate Requests** field now shows the key label of the new digital certificate request (PKCS#10) created.
13. Send the file to a CA to request a new digital certificate. You can either perform other tasks or exit the tool.

Adding (Receiving) a New Digital Certificate

After you receive a new digital certificate from a CA, you must add it to the key database from which you generated the request.

To add (receive) a new digital certificate, use the following procedure:

1. Unless you are already using Key Manager, start the tool by typing:
   ```
   # certmgr
   ```
2. From the main screen, select **Open** from the Key Database File pull down menu.
3. Highlight the key database file from which you generated the certificate request and click **Open**.
4. Enter the password and click **OK**. After your password is accepted, you are returned to the IBM Key Management screen. The title bar shows the name of the key database file you selected, indicating that the file is now open and ready to be edited.
5. Select **Personal Certificate Requests** from the Personal/Signer Certificates pull down menu.
6. Click **Receive** (to add the newly received digital certificate to your database).

7. Select the data type of the new digital certificate from the **Data type** pull down menu. The default is **Base64-encoded ASCII data**.
8. Enter the certificate file name and location for the new digital certificate, or click **Browse** to select the name and location.
9. Click **OK**.
10. Enter a descriptive label for the new digital certificate, such as:

    ```
    VPN Branch Certificate
    ```

11. Click **OK**. You are returned to the IBM Key Management screen. The **Personal Certificates** field now shows the label of the new digital certificate you just added. You can either perform other tasks or exit the tool.

    If there is an error loading the certificate, check that the certificate file begins with the text —-BEGIN CERTIFICATE—- and ends with the text —-END CERTIFICATE—-.

    For example:

    ```
    -----BEGIN CERTIFICATE-----
    ajdkfjaldfwwwwwwwwwwadafdw
    kajf;kdsajkflasasfkjafdaff
    akdjf;ldasjkf;safdfdasfdas
    kaj;fdljk98dafdas43adfadfa
    -----END CERTIFICATE-----
    ```

    If the text does not match, edit the certificate file so that it starts and ends appropriately.

Deleting a Digital Certificate

Note: Before deleting a digital certificate, create a backup copy in case you later want to re-create it.

To delete a digital certificate from your database, use the following procedure:

1. Unless you are already using Key Manager, start the tool by typing:

   ```
   # certmgr
   ```

2. From the main screen, select **Open** from the Key Database File pull down menu.
3. Highlight the key database file from which you want to delete the digital certificate and click **Open**.
4. Enter the password and click **OK**. After your password is accepted, you are returned to the IBM Key Management screen. The title bar shows the name of the key database file you selected, indicating that the file is now open and ready to be edited.
5. Select **Personal Certificate Requests** from the Personal/Signer Certificates pull down menu.
6. Highlight the digital certificate you want to delete and click **Delete**. The Confirm screen is displayed.
7. Click **Yes**. You are returned to the IBM Key Management screen. The label of the digital certificate you just deleted no longer appears in the **Personal Certificates** field. You can either perform other tasks or exit the tool.

Changing a Database Password

To change the key database, use the following procedure:

1. Unless you are already using Key Manager, start the tool by typing:

   ```
   # certmgr
   ```

2. From the main screen, select **Change Password** from the Key Database File pull down menu.
3. Enter a new password in the **Password** field, and enter it again in the **Confirm Password** field.
4. If you want to change the number of days until the password expires, enter the desired number of days in the **Set expiration time?** field. The default value for this field is 60 days. If you do not want the password to expire, clear the **Set expiration time?** field.
5. To save an encrypted version of the password in a stash file, select the **Stash the password to a file?** field and enter **yes**.

Note: You must stash the password to enable the use of digital certificates with IP Security.

6. Click **OK**. A message in the status bar indicates that the request completed successfully.
7. Click **OK** again and you return to the IBM Key Management screen. You can either perform other tasks or exit the tool.

Creating IKE Tunnels Using Digital Certificates

To create IKE tunnels that use digital certificates, you must use Web-based System Manager and the Key Manager tool.

To enable the use of digital certificates when defining the key management IKE tunnel policies, you must configure a transform that uses signature mode. *Signature mode* uses an RSA signature algorithm for authentication. IP Security provides the Web-based System Manager dialog ″Add/Change a Transform″ to allow you to select an authentication method of RSA Signature or RSA Signature with CRL Checking.

At least one endpoint of the tunnel must have a policy defined that uses a signature mode transform. You can also define other transforms using signature mode through Web-based System Manager.

The IKE key management tunnel types (the **Host Identity Type** field on the Identification tab) supported by IP Security are as follows:

- IP address
- Fully Qualified Domain Name (FQDN)
- *user@FQDN*
- X.500 Distinguished Name
- Key identifier

Use **Web-based System Manager** to select host-identity types on the Key Management Tunnel Properties - Identification tab. If you select **IP Address, FQDN**, or **user@FQDN**, you must enter values in Web-based System Manager and then provide these values to the CA. This information is used as the Subject Alternate Name in the personal digital certificate.

For example, if you choose a host identity type of **X.500 Distinguished Name** from the Web-based System Manager pull-down list on the **Identification** tab, and you enter the **Host identity** as **/C=US/O=ABC/OU=SERV/CN=*name*.austin.ibm.com**, the following are the exact values that you must enter in Key Manager when creating a digital certificate request:

- Common name: ***name*.austin.ibm.com**
- Organization: **ABC**
- Organizational unit: **SERV**
- Country : **US**

The **X.500 Distinguished Name** entered is the name set up by your system or LDAP administrator. Entering an organizational unit value is optional. The CA then uses this information when creating the digital certificate.

For another example, if you choose a host identity type of **IP Address** from the pull-down list, and you enter the host identity as **10.10.10.1**, the following are the exact values you must enter in the digital certificate request:

- Common name: ***name*.austin.ibm.com**
- Organization: **ABC**
- Organizational unit: **SERV**
- Country : **US**
- Subject alternate IP address field: **10.10.10.1**

After you create the digital certificate request with this information, the CA uses this information to create the personal digital certificate.

When requesting a personal digital certificate, the CA needs the following information:

- You are requesting a X.509 certificate.
- The signature format is MD5 with RSA encryption.
- Whether you are specifying Subject Alternate Name. Alternate name types are:
  - IP address
  - Fully qualified domain name (FQDN)
  - *user@FQDN*

  The following subject alternate-name information is included in the certificate request file.
- Your planned key use (the digital signature bit must be selected).
- The Key Manager digital certificate request file (in PKCS#10 format).

For specific steps using the Key Manager tool to create a certificate request, see "Requesting a Digital Certificate" on page 159.

Before activating the IKE tunnel, you must add the personal digital certificate you received from the CA into the Key Manager database, **ikekey.kdb**. For more information, see "Adding (Receiving) a New Digital Certificate" on page 159.

IP Security supports the following types of personal digital certificates:

Subject DN

The Subject Distinguished Name must be in the following format and order:

`/C=US/O=ABC/OU=SERV/CN=`*name*`.austin.ibm.com`

The Key Manager tool allows only one **OU** value.

Subject DN and Subject Alternate Name as an IP address

The Subject Distinguished Name and Subject Alternate Name can be designated as an IP address, as shown in the following:

`/C=US/O=ABC/OU=SERV/CN=`*name*`.austin.ibm.com` and `10.10.10.1`

Subject DN and Subject Alternate Name as FQDN

The Subject Distinguished Name and Subject Alternate Name can be designated as a fully qualified domain name, as shown in the following:

`/C=US/O=ABC/OU=SERV/CN=`*name*`.austin.ibm.com` and `bell.austin.ibm.com`.

Subject DN and Subject Alternate Name as *user@FQDN*

The Subject Distinguished Name and Subject Alternate Name can be designated as a user address (*user_ID@fully_qualified_domain_name*), as shown in the following:

`/C=US/O=ABC/OU=SERV/CN=`*name*`.austin.ibm.com` and *name*`@austin.ibm.com`.

Subject DN and multiple Subject Alternate Names

The Subject Distinguished Name can be associated with multiple Subject Alternate Names, as shown in the following:

`/C=US/O=ABC/OU=SERV/CN=`*name*`.austin.ibm.com` and `bell.austin.ibm.com`, `10.10.10.1`, and *user*`@`*name*`.austin.ibm.com`.

Configuring Manual Tunnels

The following procedures configure IP Security to use manual tunnels.

Setting Up Tunnels and Filters

To set up a manual tunnel, it is not necessary to separately configure the filter rules. As long as all traffic between two hosts goes through the tunnel, the necessary filter rules are automatically generated. The process of setting up a tunnel is to define the tunnel on one end, import the definition on the other end, and activate the tunnel and filter rules on both ends. The tunnel is then ready to use.

Information about the tunnel must be made to match on both sides if it is not explicitly supplied. For instance, the encryption and authentication algorithms specified for the source will be used for the destination if the destination values are not specified.

Creating a Manual Tunnel on the First Host

You can configure a tunnel using the Web-based System Manager Network application, the SMIT**ips4_basic** fast path (for IP Version 4) or the SMIT **ips6_basic** fast path (for IP version 6). You can also create the tunnel manually use the following procedure.

The following is a sample of the **gentun** command used to create a manual tunnel:

```
gentun -v 4 -t manual -s 5.5.5.19 -d 5.5.5.8 \
   -a HMAC_MD5 -e DES_CBC_8 -N 23567
```

You can use the **lstun -v 4** command to list the characteristics of the manual tunnel created by the previous example. The output looks similar to the following:

```
Tunnel ID            : 1
IP Version           : IP Version 4
Source               : 5.5.5.19
Destination          : 5.5.5.8
Policy               : auth/encr
Tunnel Mode          : Tunnel
Send AH Algo         : HMAC_MD5
Send ESP Algo        : DES_CBC_8
Receive AH Algo      : HMAC_MD5
Receive ESP Algo     : DES_CBC_8
Source AH SPI        : 300
Source ESP SPI       : 300
Dest AH SPI          : 23576
Dest ESP SPI         : 23576
Tunnel Life Time     : 480
Status               : Inactive
Target               : -
Target Mask          : -
Replay               : No
New Header           : Yes
Snd ENC-MAC Algo     : -
Rcv ENC-MAC Algo     : -
```

To activate the tunnel, type the following:

```
mktun -v 4 -t1
```

The filter rules associated with the tunnel are automatically generated.

To view the filter rules, use the **lsfilt -v 4** command. The output looks similar to the following:

```
Rule 4:
Rule action           : permit
Source Address        : 5.5.5.19
Source Mask           : 255.255.255.255
Destination Address   : 5.5.5.8
```

```
Destination Mask      : 255.255.255.255
Source Routing        : yes
Protocol              : all
Source Port           : any 0
Destination Port      : any 0
Scope                 : both
Direction             : outbound
Logging control       : no
Fragment control      : all packets
Tunnel ID number      : 1
Interface             : all
Auto-Generated        : yes

Rule 5:
Rule action           : permit
Source Address        : 5.5.5.8
Source Mask           : 255.255.255.255
Destination Address   : 5.5.5.19
Destination Mask      : 255.255.255.255
Source Routing        : yes
Protocol              : all
Source Port           : any 0
Destination Port      : any 0
Scope                 : both
Direction             : inbound
Logging control       : no
Fragment control      : all packets
Tunnel ID number      : 1
Interface             : all
Auto-Generated        : yes
```

To activate the filter rules, including the default filter rules, use the **mktun -v 4 -t 1** command.

To set up the other side (when it is another machine using this operating system), the tunnel definition can be exported on host A and then imported to host B.

The following command exports the tunnel definition into a file named **ipsec_tun_manu.exp** and any associated filter rules to the file **ipsec_fltr_rule.exp** in the directory indicated by the **-f** flag:

```
exptun -v 4 -t 1 -f /tmp
```

Creating a Manual Tunnel on the Second Host

To create the matching end of the tunnel, the export files are copied and imported into the remote machine by using the following command:

```
imptun -v 4 -t 1 -f /tmp
```

where

1 Is the tunnel to be imported

/tmp Is the directory where the import files reside

The tunnel number is generated by the system. You can obtain it from the output of the **gentun** command or by using the **lstun** command to list the tunnels and determine the correct tunnel number to import. If there is only one tunnel in the import file, or if all the tunnels are to be imported, the **-t** option is not needed.

If the remote machine is not running this operating system, the export file can be used as a reference for setting up the algorithm, keys, and security parameters index (SPI) values for the other end of the tunnel.

Export files from a firewall product can be imported to create tunnels. To do this, use the **-n** option when importing the file, as follows:

```
imptun -v 4 -f /tmp -n
```

Setting Up Filters

Filtering can be set up to be simple, using mostly autogenerated filter rules, or can be customized by defining very specific filter functions based on the properties of the IP packets. Matches on incoming packets are done by comparing the source address and SPI value to those listed in the filter table. Therefore, this pair must be unique.

Each line in a filter table is known as a *rule*. A collection of rules determine what packets are accepted in and out of the machine and how they are directed. Filter rules can be control many aspects of communications, including source and destination addresses and masks, protocol, port number, direction, fragment control, source routing, tunnel, and interface type.

The types of filter rules are as follows:

- "Static Filter Rules" are created in the filter table to be used for the general filtering of traffic or for associating with manual tunnels. They can be added, deleted, modified, and moved. An optional description text field can be added to identify a specific rule.
- "Autogenerated Filter Rules and User Specified Filter Rules" on page 169 (also called *autogenerated* filter rules) are a specific set of rules created for use of IKE tunnels. Both static and dynamic filter rules are created based on data management tunnel information and on data management tunnel negotiation.
- "Predefined Filter Rules" on page 170 are generic filter rules that cannot be modified, moved, or deleted, such as the `all traffic` rule, the `ah` rule, and the `esp` rule. They pertain to all traffic.

Associated with these filter rules are *Subnet masks*, which group IDs that are associated with a filter rule, and the host-firewall-host configuration option. The following sections describe the different types of filter rules and their associated features.

Static Filter Rules

Each static filter rule contains several space-separated fields. The following list provides the name of each field (an example of each field from rule 1 is shown in parentheses):

- Rule_number (`1`)
- Action (`permit`)
- Source_addr (`0.0.0.0`)
- Source_mask (`0.0.0.0`)
- Dest_addr (`0.0.0.0`)
- Dest_mask (`0.0.0.0`)
- Source_routing (`no`)
- Protocol (`udp`)
- Src_prt_operator (`eq`)
- Src_prt_value (`4001`)
- Dst_prt_operator (`eq`)
- Dst_prt_value (`4001`)
- Scope (`both`)
- Direction (`both`)
- Logging (`no`)
- Fragment (`all packets`)
- Tunnel (`0`)
- Interface (`all`).

Further explanation of static filter rules follows this example:

```
1 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 no udp eq 4001 eq 4001 both both no all
   packets 0 all

2 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 no ah any 0 any 0 both both no all packets
   0 all

3 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 no esp any 0 any 0 both both no all packets
   0 all

4 permit 10.0.0.1 255.255.255.255 10.0.0.2 255.255.255.255 no all any 0 any 0 both
   outbound no all packets 1 all outbound traffic

5 permit 10.0.0.2 255.255.255.255 10.0.0.1 255.255.255.255 no all any 0 any 0 both
   inbound no all packets 1 all


6 permit 10.0.0.1 255.255.255.255 10.0.0.3 255.255.255.255 no tcp lt 1024 eq 514 local
   outbound yes all packets 2 all


7 permit 10.0.0.3 255.255.255.255 10.0.0.1 255.255.255.255 no tcp/ack eq 514 lt 1024
   local inbound yes all packets 2 all


8 permit 10.0.0.1 255.255.255.255 10.0.0.3 255.255.255.255 no tcp/ack lt 1024 lt 1024
   local outbound yes all packets 2 all


9 permit 10.0.0.3 255.255.255.255 10.0.0.1 255.255.255.255 no tcp lt 1024 lt 1024 local
   inbound yes all packets 2 all


10 permit 10.0.0.1 255.255.255.255 10.0.0.4 255.255.255.255 no icmp any 0 any 0 local
   outbound yes all packets 3 all


11 permit 10.0.0.4 255.255.255.255 10.0.0.1 255.255.255.255 no icmp any 0 any 0 local
   inbound yes all packets 3 all


12 permit 10.0.0.1 255.255.255.255 10.0.0.5 255.255.255.255 no tcp gt 1023 eq 21 local
   outbound yes all packets 4 all


13 permit 10.0.0.5 255.255.255.255 10.0.0.1 255.255.255.255 no tcp/ack eq 21 gt 1023 local
   inbound yes all packets 4 all


14 permit 10.0.0.5 255.255.255.255 10.0.0.1 255.255.255.255 no tcp eq 20 gt 1023 local
   inbound yes all packets 4 all


15 permit 10.0.0.1 255.255.255.255 10.0.0.5 255.255.255.255 no tcp/ack gt 1023 eq 20 local
   outbound yes all packets 4 all


16 permit 10.0.0.1 255.255.255.255 10.0.0.5 255.255.255.255 no tcp gt 1023 gt 1023 local
   outbound yes all packets 4 all


17 permit 10.0.0.5 255.255.255.255 10.0.0.1 255.255.255.255 no tcp/ack gt 1023 gt 1023 local
   inbound yes all packets 4 all


18 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 no all any 0 any 0 both both yes all
   packets
```

Each rule in the previous example is described as follows:

Rule 1

For the Session Key daemon. This rule only appears in IP Version 4 filter tables. It uses port number 4001 to control packets for refreshing the session key. Rule 1 an example of how the port number can be used for a specific purpose.

Note: Do not modify this filter rule, except for logging purposes.

Rules 2 and 3

Allow processing of Authentication Headers (AH) and Encapsulating Security Payload (ESP) headers.

Note: Do not modify Rules 2 and 3, except for logging purposes.

Rules 4 and 5

Set of autogenerated rules that filter traffic between addresses 10.0.0.1 and 10.0.0.2 through tunnel 1. Rule 4 is for outbound traffic, and rule 5 is for inbound traffic.

Note: Rule 4 has a user-defined description of *outbound traffic*.

Rules 6 through 9

Set of user-defined rules that filter outbound **rsh**, **rcp**, **rdump**, **rrestore**, and **rdist** services between addresses 10.0.0.1 and 10.0.0.3 through tunnel 2. In this example, logging is set to `yes`, so that the administrator can monitor this type of traffic.

Rules 10 and 11

Set of user-defined rules that filter both inbound and outbound **icmp** services of any type between addresses 10.0.0.1 and 10.0.0.4 through tunnel 3.

Rules 12 through 17

User-defined filter rules that filter outbound file transfer protocol (FTP) service from 10.0.0.1 and 10.0.0.5 through tunnel 4.

Rule 18

Autogenerated rule always placed at the end of the table. In this example, it permits all packets that do not match the other filter rules. It can be set to deny all traffic not matching the other filter rules.

Each rule can be viewed separately (using **lsfilt**) to list each field with its value. For example:

```
Rule 1:
Rule action          : permit
Source Address       : 0.0.0.0
Source Mask          : 0.0.0.0
Destination Address  : 0.0.0.0
Destination Mask     : 0.0.0.0
Source Routing       : yes
Protocol             : udp
Source Port          : eq  4001
Destination Port     : eq  4001
Scope                : both
Direction            : both
Logging control      : no
Fragment control     : all packets
Tunnel ID number     : 0
Interface            : all
Auto-Generated       : yes
```

The following list contains all the parameters that can be specified in a filter rule:

-v IP version: 4 or 6.

-a	Action: **d** Deny **p** Permit
-s	Source address. Can be an IP address or hostname.
-m	Source subnet mask.
-d	Destination address. Can be an IP address or hostname.
-M	Destination subnet mask.
-g	Source routing control: `y` or `n`.
-c	Protocol. Values can be `udp`, `icmp`, `tcp`, `tcp/ack`, `ospf`, `pip`, `esp`, `ah` and `all`.
-o	Source port or ICMP type operation.
-p	Source port or ICMP type value.
-O	Destination port or ICMP code operation.
-P	Destination port or ICMP code value.
-r	Routing: **r** Forwarded packets **l** Local destined/originated packets **b** Both
-l	Log control. **y** Include in log **n** Do not include in log.
-f	Fragmentation. **y** Applies to fragments headers, fragments, and non-fragments **o** Applies only to fragments and fragment headers **n** Applies only to non-fragments **h** Applies only to non-fragments and fragment headers
-t	Tunnel ID.
-i	Interface, such as `tr0` or `en0`.

For more information, see the **genfilt** and **chfilt** command descriptions.

Autogenerated Filter Rules and User Specified Filter Rules

Certain rules are autogenerated for the use of the IP Security filter and tunnel code. Autogenerated rules include:

- Rules for the session key daemon that refresh the IP version 4 keys in IKE (AIX 4.3.2 and later)
- Rules for the processing of AH and ESP packets.

Filter rules are also autogenerated when defining tunnels. For manual tunnels, autogenerated rules specify the source and destination addresses and the mask values, as well as the tunnel ID. All traffic between those addresses will flow through the tunnel.

For IKE tunnels, autogenerated filter rules determine protocol and port numbers during IKE negotiation. The IKE filter rules are kept in a separate table, which is searched after the static filter rules and before the autogenerated rules. IKE filter rules are inserted in a default position within the static filter table, but they can be moved by the user.

Autogenerated rules permit all traffic over the tunnel. User-defined rules can place restrictions on certain types of traffic. Place these user-defined rules before the autogenerated rules, because IP Security uses the first rule it finds that applies to the packet. The following is an example of user-defined filter rules that filter traffic based on ICMP operation.

```
1 permit 10.0.0.1 255.255.255.255 10.0.0.4 255.255.255.255 no icmp any 8 any 0
   local outbound no all packets 3 all
2 permit 10.0.0.4 255.255.255.255 10.0.0.1 255.255.255.255 no icmp any 0 any 0 local
   inbound no all packets 3 all
3 permit 10.0.0.4 255.255.255.255 10.0.0.1 255.255.255.255 no icmp any 8 any 0 local
   inbound no all packets 3 all
4 permit 10.0.0.1 255.255.255.255 10.0.0.4 255.255.255.255 no icmp any 0 any 0 local
   outbound no all packets 3 all
```

To simplify the configuration of a single tunnel, filter rules are autogenerated when tunnels are defined. This function can be suppressed by specifying the **-g** flag in the **gentun**. You can find a sample filter file with **genfilt** commands to generate filter rules for different TCP/IP services in **/usr/samples/ipsec/filter.sample**.

Predefined Filter Rules

Several predefined filter rules are autogenerated with certain events. When the `ipsec_v4` or `ipsec_v6` device is loaded, a predefined rule is inserted into the filter table and then activated. By default, this predefined rule is to permit all packets, but it is user-configurable and you can set it to deny all packets.

> **Note:** When configuring remotely, ensure that the deny rule is not enabled before the configuration is complete, to prevent your session from getting locked out of the machine. The situation can be avoided either by setting the default action to permit or by configuring a tunnel to the remote machine before activating IP Security.

Both IPv4 and IPv6 filter tables have a predefined rule. Either may be independently changed to deny all. This will keep traffic from passing unless that traffic is specifically defined by additional filter rules. The only other option to change on the predefined rules is **chfilt** with the **-l** option, which allows packets matching that rule to be logged.

To support IKE tunnels, a dynamic filter rule is placed in the IPv4 filter table. This is the position at which dynamic filter rules are inserted into the filter table. This position can be controlled by the user by moving its position up and down the filter table. After the tunnel manager daemon and **isakmpd** daemon are initialized to allow IKE tunnels to be negotiated, rules are automatically created in the dynamic filter table to handle IKE messages as well as AH and ESP packets.

Subnet Masks

Subnet masks are used to group a set of IDs that are associated with a filter rule. The mask value is ANDed with the ID in the filter rules and compared to the ID specified in the packet. For example, a filter rule with a source IP address of `10.10.10.4` and a subnet mask of `255.255.255.255` specified that an exact match must occur of the decimal IP address, as shown in the following:

	Binary	**Decimal**
Source IP address	`1010.1010.1010.0100`	`10.10.10.4`
Subnet mask	`1111.1111.1111.1111`	`255.255.255.255`

A `10.10.10.x` subnet is specified as `1111.1111.1111.0` or `255.255.255.0`. An incoming address would have the subnet mask appended to it, then the combination would be compared to the ID in the filter rule. For example, an address of `10.10.10.100` becomes `10.10.10.0` after the subnet mask is applied, which matches the filter rule.

A subnet mask of `255.255.255.240` allows any value for the last four bits in the address.

Host-Firewall-Host Configuration

The host-firewall-host configuration option for tunnels allows you to create a tunnel between your host and a firewall, then automatically generate the necessary filter rules for correct communication between your host and a host behind the firewall. The autogenerated filter rules permit all rules between the two non-firewall hosts over the tunnel specified. The default rules—for user datagram protocol (UDP), Authentication Headers (AH), and Encapsulating Security Payload (ESP) headers—should already handle the host to firewall communication. The firewall will have to be configured appropriately to complete the setup. You should use the export file from the tunnel you created to enter the SPI values and keys that the firewall needs.

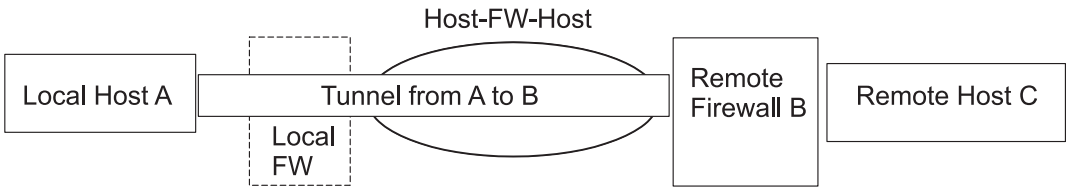


Figure 12. Host-Firewall-Host. This illustration shows a Host-Firewall-Host configuration. Host A has a tunnel running through a local firewall and out to the internet. Then it goes to Remote Firewall B, and then on to Remote Host C.

Logging Facilities

This section describes the configuration and format of system logs relating to IP Security. As hosts communicate with each other, the transferred packets may be logged to the system log daemon, **syslogd**. Other important messages about IP Security also display. An administrator may choose to monitor this logging information for traffic analysis and debugging assistance. The following are the steps for setting up the logging facilities.

1. Edit the **/etc/syslog.conf** file to add the following entry:

   ```
   local4.debug var/adm/ipsec.log
   ```

   Use the `local4` facility to record traffic and IP Security events. Standard operating system priority levels apply. You should set the priority level of `debug` until traffic through IP Security tunnels and filters show stability and proper movement.

   > **Note:** The logging of filter events can create significant activity at the IP Security host and can consume large amounts of storage.

2. Save **/etc/syslog.conf**.
3. Go to the directory you specified for the log file and create an empty file with the same name. In the case above, you would change to **/var/adm** directory and issue the command:

   ```
   touch ipsec.log
   ```

4. Issue a **refresh** command to the **syslogd** subsystem:

   ```
   refresh -s syslogd
   ```

5. If using IKE tunnels, ensure the **/etc/isakmpd.conf** file specifies the desired **isakmpd** logging level. (See “IP Security Problem Determination” on page 176 for more information on IKE logging.)
6. While creating filter rules for your host, if you would like packets matching a specific rule to be logged, set the **-l** parameter for the rule to **Y** (yes) using the **genfilt** or the **chfilt** commands.
7. Turn on packet logging and start the **ipsec_logd** daemon using the following command:

   ```
   mkfilt -g start
   ```

   You can stop packet logging by issuing the following command:

   ```
   mkfilt -g stop
   ```

The following sample log file contains traffic entries and other IP Security log entries:

```
1. Aug 27 08:08:40 host1 : Filter logging daemon ipsec_logd (level 2.20)
   initialized at 08:08:40 on 08/27/97A
2. Aug 27 08:08:46 host1 : mkfilt: Status of packet logging set to Start
   at 08:08:46 on 08/27/97
3. Aug 27 08:08:47 host1 : mktun: Manual tunnel 2 for IPv4, 9.3.97.244, 9.3.97.130
   activated.
4. Aug 27 08:08:47 host1 : mkfilt: #:1 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0
   udp eq  4001 eq  4001  both both l=n f=y t=0 e= a=
5. Aug 27 08:08:47 host1 : mkfilt: #:2 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0
   ah any 0 any 0  both both l=n f=y t=0 e= a=
6. Aug 27 08:08:47 host1 : mkfilt: #:3 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0
   esp any 0 any 0  both both l=n f=y t=0 e= a=
7. Aug 27 08:08:47 host1 : mkfilt: #:4 permit 10.0.0.1 255.255.255.255 10.0.0.2
   255.255.255.255 icmp any 0 any 0  local outbound l=y f=y t=1 e= a=
8. Aug 27 08:08:47 host1 : mkfilt: #:4 permit 10.0.0.2 255.255.255.255 10.0.0.1
   255.255.255.255 icmp any 0 any 0  local inbound l=y f=y t=1 e= a=
9. Aug 27 08:08:47 host1 : mkfilt: #:6 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0
    all any 0 any 0  both both l=y f=y t=0 e= a=
10. Aug 27 08:08:47 host1 : mkfilt: Filter support (level 1.00) initialized at
    08:08:47 on 08/27/97
11. Aug 27 08:08:48 host1 : #:6 R:p  o:10.0.0.1 s:10.0.0.1 d:10.0.0.20 p:udp
    sp:3327 dp:53 r:l a:n f:n T:0 e:n l:67
12. Aug 27 08:08:48 host1 : #:6 R:p  i:10.0.0.1 s:10.0.0.20 d:10.0.0.1 p:udp
```

```
    sp:53 dp:3327 r:l a:n f:n T:0 e:n l:133
13. Aug 27 08:08:48 host1 : #:6 R:p  i:10.0.0.1 s:10.0.0.15 d:10.0.0.1 p:tcp
    sp:4649 dp:23 r:l a:n f:n T:0 e:n l:43
14. Aug 27 08:08:48 host1 : #:6 R:p  o:10.0.0.1 s:10.0.0.1 d:10.0.0.15 p:tcp
    sp:23 dp:4649 r:l a:n f:n T:0 e:n l:41
15. Aug 27 08:08:48 host1 : #:6 R:p  i:10.0.0.1 s:10.0.0.15 d:10.0.0.1 p:tcp
    sp:4649 dp:23 r:l a:n f:n T:0 e:n l:40
16. Aug 27 08:08:51 host1 : #:4 R:p  o:10.0.0.1 s:10.0.0.1 d:10.0.0.2 p:icmp
    t:8 c:0 r:l a:n f:n T:1 e:n l:84
17. Aug 27 08:08:51 host1 : #:5 R:p  i:10.0.0.1 s:10.0.0.2 d:10.0.0.1 p:icmp
    t:0 c:0 r:l a:n f:n T:1 e:n l:84
18. Aug 27 08:08:52 host1 : #:4 R:p  o:10.0.0.1 s:10.0.0.1 d:10.0.0.2 p:icmp
    t:8 c:0 r:l a:n f:n T:1 e:n l:84
19. Aug 27 08:08:52 host1 : #:5 R:p  i:10.0.0.1 s:10.0.0.2 d:10.0.0.1 p:icmp
    t:0 c:0 r:l a:n f:n T:1 e:n l:84
20. Aug 27 08:32:27 host1 : Filter logging daemon terminating at 08:32:27 on
    08/27/97l
```

The following paragraphs explain the log entries.

1 Filter logging daemon activated.

2 Filter packet logging set to on with the **mkfilt -g start** command.

3 Tunnel activation, showing tunnel ID, source address, destination address, and time stamp.

4-9 Filters have been activated. Logging shows all loaded filter rules.

10 Message showing activation of filters.

11-12 These entries show a DNS lookup for a host.

13-15 These entries show a partial Telnet connection (the other entries have been removed from this example for space reasons).

16-19 These entries show two pings.

20 Filter logging daemon shutting down.

The following example shows two hosts negotiating a phase 1 and a phase 2 tunnel from the initiating host's point of view. (The **isakmpd** logging level has been specified as **isakmp_events**.)

```
1. Dec  6 14:34:42 host1 Tunnel Manager: 0: TM is processing a
    Connection_request_msg
 2. Dec  6 14:34:42 host1 Tunnel Manager: 1: Creating new P1 tunnel object (tid)
 3. Dec  6 14:34:42 host1 isakmpd: 192.168.100.103 >>> 192.168.100.104 ( SA PROPOSAL
    TRANSFORM  )
 4. Dec  6 14:34:42 host1 isakmpd: ::ffff:192.168.100.103 <<< 192.168.100.104 ( SA
    PROPOSAL TRANSFORM  )
 5. Dec  6 14:34:42 host1 isakmpd: Phase I SA Negotiated
 6. Dec  6 14:34:42 host1 isakmpd: 192.168.100.103 >>> 192.168.100.104 ( KE NONCE  )
 7. Dec  6 14:34:42 host1 isakmpd: ::ffff:192.168.100.103 <<< 192.168.100.104 ( KE
    NONCE  )
 8. Dec  6 14:34:42 host1 isakmpd: Encrypting the following msg to send: ( ID HASH
     )
 9. Dec  6 14:34:42 host1 isakmpd: 192.168.100.103 >>> 192.168.100.104 ( Encrypted
    Payloads )
10. Dec  6 14:34:42 host1 isakmpd: ::ffff:192.168.100.103 <<< 192.168.100.104 (
    Encrypted Payloads )
11. Dec  6 14:34:42 host1 Tunnel Manager: 1: TM is processing a P1_sa_created_msg
    (tid)
12. Dec  6 14:34:42 host1 Tunnel Manager: 1:   Received good P1 SA, updating P1
    tunnel (tid)
13. Dec  6 14:34:42 host1 Tunnel Manager: 0: Checking to see if any P2 tunnels need
    to start
14. Dec  6 14:34:42 host1 isakmpd: Decrypted the following received msg: ( ID HASH
     )
15. Dec  6 14:34:42 host1 isakmpd:  Phase I Done !!!
```

```
16. Dec  6 14:34:42 host1 isakmpd: Phase I negotiation authenticated
17. Dec  6 14:34:44 host1 Tunnel Manager: 0: TM is processing a
    Connection_request_msg
18. Dec  6 14:34:44 host1 Tunnel Manager: 0: Received a connection object for an
    active P1 tunnel
19. Dec  6 14:34:44 host1 Tunnel Manager: 1: Created blank P2 tunnel (tid)
20. Dec  6 14:34:44 host1 Tunnel Manager: 0: Checking to see if any P2 tunnels need
    to start
21. Dec  6 14:34:44 host1 Tunnel Manager: 1: Starting negotiations for P2 (P2 tid)
22. Dec  6 14:34:45 host1 isakmpd: Encrypting the following msg to send: ( HASH SA
    PROPOSAL TRANSFORM NONCE ID ID  )
23. Dec  6 14:34:45 host1 isakmpd: 192.168.100.103 >>> 192.168.100.104 ( Encrypted
    Payloads )
24. Dec  6 14:34:45 host1 isakmpd: ::ffff:192.168.100.103 <<< 192.168.100.104 (
    Encrypted Payloads )
25. Dec  6 14:34:45 host1 isakmpd: Decrypted the following received msg: ( HASH SA
    PROPOSAL TRANSFORM NONCE ID ID  )
26. Dec  6 14:34:45 host1 isakmpd: Encrypting the following msg to send: ( HASH  )
27. Dec  6 14:34:45 host1 isakmpd: 192.168.100.103 >>> 192.168.100.104 ( Encrypted
    Payloads )
28. Dec  6 14:34:45 host1 isakmpd: Phase II SA Negotiated
29. Dec  6 14:34:45 host1 isakmpd: PhaseII negotiation complete.
30. Dec  6 14:34:45 host1 Tunnel Manager: 0: TM is processing a P2_sa_created_msg
31. Dec  6 14:34:45 host1 Tunnel Manager: 1: received p2_sa_created for an existing
    tunnel as initiator (tid)
32. Dec  6 14:34:45 host1 Tunnel Manager: 1: Filter::AddFilterRules: Created filter
    rules for tunnel
33. Dec  6 14:34:45 host1 Tunnel Manager: 0: TM is processing a List_tunnels_msg
```

The following paragraphs explain the log entries.

1-2 The **ike cmd=activate phase=1** command initiates a connection.

3-10 The **isakmpd** daemon negotiates a phase 1 tunnel.

11-12 The Tunnel Manager receives a valid phase 1 security association from the responder.

13 The Tunnel Manager checks whether **ike cmd=activate** has a phase 2 value for more work. It does not.

14-16 The **isakmpd** daemon finishes the phase 1 negotiation.

17-21 The **ike cmd=activate phase=2** command initiates a phase 2 tunnel.

22-29 The **isakmpd** daemon negotiates a phase 2 tunnel.

30-31 The Tunnel Manager receives a valid phase 2 security association from responder.

32 The Tunnel Manager writes the dynamic filter rules.

33 The **ike cmd=list** command views the IKE tunnels.

Labels in Field Entries

The fields in the log entries are abbreviated to reduce DASD space requirements:

The rule number that caused this packet to be logged.

R Rule Type

- **p** Permit
- **d** Deny

i/o Direction the packet was traveling when it was intercepted by the filter support code. Identifies IP address of the adapter associated with the packet:

- For inbound (i) packets, this is the adapter that the packet arrived on.
- For outbound (o) packets, this is the adapter that the IP layer has determined should handle the transmission of the packet.

s Specifies the IP address of the sender of the packet (extracted from the IP header).

d Specifies the IP address of the intended recipient of the packet (extracted from the IP header).

p Specifies the high-level protocol that was used to create the message in the data portion of the packet. May be a number or name, for example: `udp`, `icmp`, `tcp`, `tcp/ack`, `ospf`, `pip`, `esp`, `ah`, or `all`.

sp/t Specifies the protocol port number associated with the sender of the packet (extracted from the TCP/UDP header). When the protocol is ICMP or OSPF, this field is replaced with **t**, which specifies the IP type.

dp/c Specifies the protocol port number associated with the intended recipient of the packet (extracted from the TCP/UDP header). When the protocol is ICMP, this field is replaced with **c**, which specifies the IP code.

- Specifies that no information is available

r Indicates whether the packet had any local affiliation.

- **f** Forwarded packets
- **l** Local packets
- **o** Outgoing
- **b** Both

l Specifies the length of a particular packet in bytes.

f Identifies if the packet is a fragment.

T Indicates the tunnel ID.

i Specifies what interface the packet came in on.

IP Security Problem Determination

This section includes some hints and tips that may assist you when you encounter a problem. It is recommended that logging be set up when IPSec is first configured. Logs are very useful in determining what occurs with the filters and tunnels. (For detailed log information, see “Logging Facilities” on page 172.)

Troubleshooting Manual Tunnel Errors

Error: Issuing **mktun** command results in the following error:

```
insert_tun_man4(): write failed : The requested resource is busy.
```

Problem: The tunnel you requested to activate is already active or you have colliding SPI values.

To fix: Issue the **rmtun** command to deactivate, then issue the mktun command to activate. Check to see if the SPI values for the failing tunnel match any other active tunnel. Each tunnel should have its own unique SPI values.

Error: Issuing **mktun** command results in the following error:

```
Device ipsec_v4 is in Defined status.
Tunnel activation for IP Version 4 not performed.
```

Problem: You have not made the IP Security device available.

To fix: Issue the following command:

```
mkdev -l ipsec -t 4
```

You may have to change **-t** option to 6 if you are getting the same error for IP Version 6 tunnel activation. The devices must be in available state. To check the IP Security device state, issue the following command:

```
lsdev -Cc ipsec
```

Error: Issuing a **gentun** command results in the following error:

```
Invalid Source IP address
```

Problem: You have not entered a valid IP address for the source address.

To fix: For IP Version 4 tunnels, check to see that you have entered an available IP Version 4 address for the local machine. You cannot use host names for the source when generating tunnels, you may only use host names for the destination.

For IP Version 6 tunnels, check to see that you entered an available IP Version 6 address. If you type `netstat -in` and no IP Version 6 addresses exist, run **/usr/sbin/autoconf6** (interface) for a link local autogenerated address (using MAC address) or use the **ifconfig** command to manually assign an address.

Error: Issuing a **gentun** command results in the following error:

```
Invalid Source IP address
```

Problem: You have not entered a valid IP address for the source address.

To fix: For IP Version 4 tunnels, check to see that you have entered an available IP Version 4 address for the local machine. You cannot use host names for the source when generating tunnels, you may only use host names for the destination.

For IP Version 6 tunnels, check to see that you entered an available IP Version 6 address. If you type `netstat -in` and no IP Version 6 addresses exist, run **/usr/sbin/autoconf6** (interface) for a link local auto-generated address (using MAC address) or use **ifconfig** to manually assign an address.

Error: Issuing **mktun** command results in the following error:

```
insert_tun_man4(): write failed : A system call received a parameter that is not valid.
```

Problem: Tunnel generation occurred with invalid ESP and AH combination or without the use of the new header format when necessary.

To fix: Check to see what authentication algorithms are in use by the particular tunnel in question. Remember that the HMAC_MD5 and HMAC_SHA algorithms require the new header format. The new header format can be changed using the SMIT fast path **ips4_basic** or the **-z** parameter with the **chtun** command. Also remember that DES_CBC_4 cannot be used with the new header format.

Error: Starting IP Security from Web-based System Manager results in a `Failure` message.

Problem: The IP Security daemons are not running.

To fix: View which daemons are running by entering the `ps -ef` command. The following daemons are associated with IP Security:

- **tmd**
- **isakmpd**
- **cpsd**

The **cpsd** daemon is active only if the digital certificate code is installed (the fileset named **gskit.rte** or **gskkm.rte**) and you have configured the Key Manager tool to contain digital certificates.

If the daemons are not active, stop IP Security using Web-based System Manager and then restart it, which automatically starts the appropriate daemons.

Error: Trying to use IP Security results in the following error:

```
The installed bos.crypto is back level and must be updated.
```

Problem: The **bos.net.ipsec.*** files have been updated to a newer version, but the corresponding **bos.crypto.*** files have not.

To fix: Update the **bos.crypto.*** files to the version that corresponds with the updated **bos.net.ipsec.*** files.

Troubleshooting IKE Tunnel Errors

The following sections describe errors that can occur when using IKE tunnels.

IKE Tunnel Process Flow

The IKE tunnels are set up by the communication of the **ike** command or the Web-based System Manager VPN panels with the following daemons:

Table 10. Daemons used by IKE tunnels.

tmd	Tunnel Manager daemon
isakmpd	IKE daemon
cpsd	Certificate proxy daemon

For IKE tunnels to be correctly set up, the **tmd** and **isakmpd** daemons must be running. If IP Security is set to start at reboot, these daemons start automatically. Otherwise, they must be started using Web-based System Manager.

The Tunnel Manager gives requests to the **isakmpd** command to start a tunnel. If the tunnel already exists or is not valid (for instance, has an invalid remote address), it reports an error. If negotiation has started, it may take some time, depending on network latency, for the negotiation to complete. The **ike cmd=list** command can list the state of the tunnel to determine if the negotiation was successful. Also, the Tunnel Manager logs events to **syslog** to the levels of **debug**, **event**, and **information**, which can be used to monitor the progress of the negotiation.

The sequence is as follows:

1. Use Web-based System Manager or the **ike** command to initiate a tunnel.
2. The **tmd** daemon gives the **isakmpd** daemon a connection request for key management (phase 1).
3. The **isakmpd** daemon responds with `SA created` or an error.
4. The **tmd** daemon gives the **isakmpd** daemon a connection request for a data management tunnel (phase 2).
5. The **isakmpd** daemon responds with `SA created` or an error.
6. Tunnel parameters are inserted into the kernel tunnel cache.
7. Filter rules are added to the kernel dynamic filter table.

When the machine is acting as a responder, the **isakmpd** daemon notifies the Tunnel Manager **tmd** daemon that a tunnel has been negotiated successfully and a new tunnel is inserted into the kernel. In such cases, the process starts with step 3 and continues until step 7, without the **tmd** daemon issuing connection requests.

IKE Logging

The **isakmpd**, **tmd** and **cpsd** daemons log events to **syslog**. For the **isakmpd** daemon, logging is enabled using the **ike cmd=log** command. The **/etc/isakmpd.conf** configuration file can be set up to specify the logging level. The level can be set to **none**, **errors**, **isakmp_events**, or **information**.

> **Note:** In versions earlier than AIX 5.1, the **isakmpd** daemon logged to a separate file, which was also specified in **/etc/isakmpd.conf** file.

The configuration file parameter that can be set for logging is log_level. The IKE daemons use the following levels of logging:

none No logging (the default)

error Only logging protocol and API errors

isakmp_events
Only logging IKE protocol events and errors

Information
Logging protocol and implementation information (recommended for debugging)

The syntax for this option is simply as follows:

```
log_level
```

The **isakmpd** daemon code either initiates by sending a proposal or responds by evaluating a proposal. If the proposal is accepted, a security association is created and the tunnel is set up. If the proposal is not accepted or the connection times out before the negotiation completes, the **isakmpd** daemon indicates an error. The entries in **syslog** from **tmd** indicate whether the negotiation succeeded. A failure caused by an invalid certificate logs to **syslog**. To determine the exact cause of a failed negotiation, check the logging file specified in **/etc/syslog.conf**.

The syslog facility adds a prefix to each log line, noting the date and time, the machine, and the program. The following example uses `googly` as the machine name and `isakmpd` as the program name:

```
Nov 20 09:53:50 googly isakmpd: ISAKMP_MSG_HEADER
Nov 20 09:53:50 googly isakmpd:  Icookie : 0xef06a77488f25315, Rcookie :0x0000000000000000
Nov 20 09:53:51 googly isakmpd:  Next Payload : 1(SA), Maj Ver : 1, Min Ver : 0
Nov 20 09:53:51 googly isakmpd:  Xchg Type : 2 (ID protected), Flag= 0, Encr : No,COMMIT : No
Nov 20 09:53:51 googly isakmpd:  Msg ID  : 0x00000000
```

To improve clarity, the **grep** command can be used to extract log lines of interest (such as all **isakmpd** logging) and the **cut** command can be used to remove the prefix from each line. The **isakmpd** log examples in the rest of this section have been tailored in a similar way.

Parse Payload Logging Function

The security association (SA) between two end points is established by exchanging IKE messages. The Parse Payload function parses the messages in a human-readable format. The logging can be enabled by editing the **/etc/isakmpd.conf** file. The logging entry in the **/etc/isakmpd.conf** file looks similar to the following:

```
information
```

The type of IKE payloads that Parse Payload logs depends on the content of the IKE message. Examples include SA Payload, Key Exchange Payload, Certificate Request Payload, Certificate Payload, and Signature Payload. The following is an example of a Parse Payload log in which an ISAKMP_MSG_HEADER is followed by five payloads:

```
ISAKMP_MSG_HEADER
        Icookie : 0x9e539a6fd4540990, Rcookie : 0x0000000000000000
        Next Payload : 1(SA), Maj Ver : 1, Min Ver : 0
        Xchg Type : 4 (Aggressive), Flag= 0, Encr : No,COMMIT : No
        Msg ID  : 0x00000000
        len     : 0x10e(270)
SA Payload:
        Next Payload : 4(Key Exchange), Payload len : 0x34(52)
        DOI          : 0x1(INTERNET)
        bitmask      : 1(SIT_IDENTITY_ONLY
   Proposal Payload:
        Next Payload : 0(NONE), Payload len : 0x28(40)
        Proposal # : 0x1(1), Protocol-ID : 1(ISAKMP)
        SPI size : 0x0(0), # of Trans : 0x1(1)
   Transform Payload:
        Next Payload : 0(NONE), Payload len : 0x20(32)
        Trans # : 0x1(1), Trans.ID : 1(KEY_IKE)
        Attr : 1(Encr.Alg     ), len=0x2(2)
        Value=0x1(1),(DES-cbc)
        Attr : 2(Hash Alg     ), len=0x2(2)
        Value=0x1(1),(MD5)
        Attr : 3(Auth Method  ), len=0x2(2)
        Value=0x3(3),(RSA Signature)
        Attr : 4(Group Desc   ), len=0x2(2)
        Value=0x1(1),(default 768-bit MODP group)
        Attr : 11(Life Type    ), len=0x2(2)
        Value=0x1(1),(seconds)
        Attr : 12(Life Duration), len=0x2(2)
        Value=0x7080(28800)
Key Payload:
        Next Payload : 10(Nonce), Payload len : 0x64(100)

        Key Data :
        33 17 68 10 91 1f ea da 38 a0 22 2d 84 a3 5d 5d
        a0 e1 1f 42 c2 10 aa 8d 9d 14 0f 58 3e c4 ec a3
        9f 13 62 aa 27 d8 e5 52 8d 5c c3 cf d5 45 1a 79
        8a 59 97 1f 3b 1c 08 3e 2a 55 9b 3c 50 cc 82 2c
        d9 8b 39 d1 cb 39 c2 a4 05 8d 2d a1 98 74 7d 95
        ab d3 5a 39 7d 67 5b a6 2e 37 d3 07 e6 98 1a 6b

Nonce Payload:
        Next Payload : 5(ID), Payload len : 0xc(12)

        Nonce Data:
        6d 21 73 1d dc 60 49 93
ID Payload:
        Next Payload : 7(Cert.Req), Payload len : 0x49(73)
        ID type      : 9(DER_DN), Protocol : 0, Port = 0x0(0)
Certificate Request Payload:
        Next Payload : 0(NONE), Payload len : 0x5(5)
        Certificate Encoding Type: 4(X.509 Certificate - Signature)
```

Within each payload, a Next Payload field points to the payload following the current payload. If the current payload is the last one in the IKE message, the `Next Payload` field has the value of zero (None).

Each Payload in the example has information pertaining to the negotiations that are going on. For example, the SA payload has the Proposal and Transform Payloads, which in turn show the encryption algorithm, authentication mode, hash algorithm, SA life type, and SA duration that the initiator is proposing to the responder.

Also, the SA Payload consists of one or more Proposal Payloads and one or more Transform Payloads. The `Next Payload` field for Proposal Payload has a value of either 0 if it is the only Proposal Payload or a value of 2 if it is followed by one more Proposal Payloads. Similarly, the `Next Payload` field for a Transform Payload has a value of 0 if it is the only Transform Payload, or a value of 3 if it is followed by one more Transform Payloads, as shown in the following example:

```
ISAKMP_MSG_HEADER
        Icookie : 0xa764fab442b463c6, Rcookie : 0x0000000000000000
        Next Payload : 1(SA), Maj Ver : 1, Min Ver : 0
        Xchg Type : 2 (ID protected), Flag= 0, Encr : No,COMMIT : No
        Msg ID  : 0x00000000
        len     : 0x70(112)
SA Payload:
        Next Payload : 0(NONE), Payload len : 0x54(84)
        DOI          : 0x1(INTERNET)
        bitmask      : 1(SIT_IDENTITY_ONLY
   Proposal Payload:
        Next Payload : 0(NONE), Payload len : 0x48(72)
        Proposal # : 0x1(1), Protocol-ID : 1(ISAKMP)
        SPI size : 0x0(0), # of Trans : 0x2(2)
   Transform Payload:
        Next Payload : 3(Transform), Payload len : 0x20(32)
        Trans # : 0x1(1), Trans.ID : 1(KEY_IKE)
        Attr : 1(Encr.Alg     ), len=0x2(2)
        Value=0x5(5),(3DES-cbc)
        Attr : 2(Hash Alg     ), len=0x2(2)
        Value=0x1(1),(MD5)
        Attr : 3(Auth Method  ), len=0x2(2)
        Value=0x1(1),(Pre-shared Key)
        Attr : 4(Group Desc   ), len=0x2(2)
        Value=0x1(1),(default 768-bit MODP group)
        Attr : 11(Life Type    ), len=0x2(2)
        Value=0x1(1),(seconds)
        Attr : 12(Life Duration), len=0x2(2)
        Value=0x7080(28800)
   Transform Payload:
        Next Payload : 0(NONE), Payload len : 0x20(32)
        Trans # : 0x2(2), Trans.ID : 1(KEY_IKE)
        Attr : 1(Encr.Alg     ), len=0x2(2)
        Value=0x1(1),(DES-cbc)
        Attr : 2(Hash Alg     ), len=0x2(2)
        Value=0x1(1),(MD5)
        Attr : 3(Auth Method  ), len=0x2(2)
        Value=0x1(1),(Pre-shared Key)
        Attr : 4(Group Desc   ), len=0x2(2)
        Value=0x1(1),(default 768-bit MODP group)
        Attr : 11(Life Type    ), len=0x2(2)
        Value=0x1(1),(seconds)
        Attr : 12(Life Duration), len=0x2(2)
        Value=0x7080(28800)
```

The IKE Message Header of a Parse Payload log shows the exchange type (Main Mode or Aggressive Mode), the length of the entire message, the message identifier, and so on.

The Certificate Request Payload requests a certificate from the responder. The responder sends the certificate in a separate message. The following example shows the Certificate Payload and Signature Payload that are sent to a peer as a part of an SA negotiation. The certificate data and the signature data are printed in hex format.

```
ISAKMP_MSG_HEADER
        Icookie : 0x9e539a6fd4540990, Rcookie : 0xc7e0a8d937a8f13e
        Next Payload : 6(Certificate), Maj Ver : 1, Min Ver : 0
        Xchg Type : 4 (Aggressive), Flag= 0, Encr : No,COMMIT : No
        Msg ID  : 0x00000000
        len     : 0x2cd(717)
Certificate Payload:

        Next Payload : 9(Signature), Payload len : 0x22d(557)
        Certificate Encoding Type: 4(X.509 Certificate - Signature)
        Certificate: (len 0x227(551) in bytes
        82 02 24 30 82 01 8d a0 03 02 01 02 02 05 05 8e
        fb 3e ce 30 0d 06 09 2a 86 48 86 f7 0d 01 01 04
        05 00 30 5c 31 0b 30 09 06 03 55 04 06 13 02 46
        49 31 24 30 22 06 03 55 04 0a 13 1b 53 53 48 20
        43 6f 6d 6d 75 6e 69 63 61 74 69 6f 6e 73 20 53
        65 63 75 72 69 74 79 31 11 30 0f 06 03 55 04 0b
        13 08 57 65 62 20 74 65 73 74 31 14 30 12 06 03
        55 04 03 13 0b 54 65 73 74 20 52 53 41 20 43 41
        30 1e 17 0d 39 39 30 39 32 31 30 30 30 30 30 30
        5a 17 0d 39 39 31 30 32 31 32 33 35 39 35 39 5a
        30 3f 31 0b 30 09 06 03 55 04 06 13 02 55 53 31
        10 30 0e 06 03 55 04 0a 13 07 49 42 4d 2f 41 49
        58 31 1e 30 1c 06 03 55 04 03 13 15 62 61 72 6e
        65 79 2e 61 75 73 74 69 6e 2e 69 62 6d 2e 63 6f
        6d 30 81 9f 30 0d 06 09 2a 86 48 86 f7 0d 01 01
        01 05 00 03 81 8d 00 30 81 89 02 81 81 00 b2 ef
        48 16 86 04 7e ed ba 4c 14 d7 83 cb 18 40 0a 3f
        55 e9 ad 8f 0f be c5 b6 6d 19 ec de 9b f5 01 a6
        b9 dd 64 52 34 ad 3d cd 0d 8e 82 6a 85 a3 a8 1c
        37 e4 00 59 ce aa 62 24 b5 a2 ea 8d 82 a3 0c 6f
        b4 07 ad 8a 02 3b 19 92 51 88 fb 2c 44 29 da 72
        41 ef 35 72 79 d3 e9 67 02 b2 71 fa 1b 78 13 be
        f3 05 6d 10 4a c7 d5 fc fe f4 c0 b8 b8 fb 23 70
        a6 4e 16 5f d4 b1 9e 21 18 82 64 6d 17 3b 02 03
        01 00 01 a3 0f 30 0d 30 0b 06 03 55 1d 0f 04 04
        03 02 07 80 30 0d 06 09 2a 86 48 86 f7 0d 01 01
        04 05 00 03 81 81 00 75 a4 ee 9c 3a 18 f2 de 5d
        67 d4 1c e4 04 b4 e5 b8 5e 9f 56 e4 ea f0 76 4a
        d0 e4 ee 20 42 3f 20 19 d4 25 57 25 70 0a ea 41
        81 3b 0b 50 79 b5 fd 1e b6 0f bc 2f 3f 73 7d dd
        90 d4 08 17 85 d6 da e7 c5 a4 d6 9a 2e 8a e8 51
        7e 59 68 21 55 4c 96 4d 5a 70 7a 50 c1 68 b0 cf
        5f 1f 85 d0 12 a4 c2 d3 97 bf a5 42 59 37 be fe
        9e 75 23 84 19 14 28 ae c4 c0 63 22 89 47 b1 b6
        f4 c7 5d 79 9d ca d0
Signature Payload:
        Next Payload : 0(NONE), Payload len : 0x84(132)

        Signature: len 0x80(128) in bytes
        9d 1b 0d 90 be aa dc 43 95 ba 65 09 b9 00 6d 67
        b4 ca a2 85 0f 15 9e 3e 8d 5f e1 f0 43 98 69 d8
        5c b6 9c e2 a5 64 f4 ef 0b 31 c3 cb 48 7c d8 30
        e3 a2 87 f4 7c 9d 20 49 b2 39 00 fa 8e bf d9 b0
        7d b4 8c 4e 19 3a b8 70 90 88 2c cf 89 69 5d 07
        f0 5a 81 58 2e 15 40 37 b7 c8 d6 8c 5c e2 50 c3
        4d 19 7e e0 e7 c7 c2 93 42 89 46 6b 5f f8 8b 7d
        5b cb 07 ea 36 e5 82 9d 70 79 9a fe bd 6c 86 36
```

Digital Certificate and Signature Mode Problems

Error: The **cpsd** (Certificate Proxy Server daemon) does not start. An entry similar to the following appears in the log file:

```
Sep 21 16:02:00 ripple CPS[19950]: Init():LoadCaCerts() failed, rc=-12
```

Problem: The certificate database has not opened or has not been found.

To Fix: Ensure that the Key Manager certificate databases are present in **/etc/security**. The following files make up the database: **ikekey.crl**, **ikekey.kdb**, **ikekey.rdb**, **ikekey.sth**.

If only the **ikekey.sth** file is missing, the `stash password` option was not selected when the Key Manager database was created. The password must be stashed to enable using digital certificates with IP Security. (See Creating a Key Database for more information.)

Error: Key Manager gives the following error when receiving a certificate:

```
Invalid Base64-encoded data was found
```

Problem: Superfluous data has been found in the certificate file or else data was lost or corrupted.

To Fix: The 'DER' Encoded Certificate should be contained within the following strings (shown below). No other characters should precede or follow other than the BEGIN and END CERTIFICATE strings.

```
-----BEGIN CERTIFICATE-----
MIICMTCCAZqgAwIBAgIFFKZtANowDQYJKoZIhvcNAQEFBQAwXDELMAkGA1UEBhMC
RkkxJDAiBgNVBAoTG1NTSCBDb21tdW5pY2F0aW9ucyBTZWN1cml0eTERMA8GA1UE
CxMIV2ViIHRlc3QxFDASBgNVBAMTC1Rlc3QgUlNBIENBMB4XDTk5MDkyMTAwMDAw
MFoXDTk5MTAyMTIzNTk1OVowOzELMAkGA1UEBhMCVVMxDDAKBgNVBAoTA0lCTTEe
MBwGA1UEAxMVcmlwcGxlLmF1c3Rpbi5pYm0uY29tMIGfMA0GCSqGSIb3DQEBAQUA
A4GNADCBiQKBgQC5EZqo6n7tZrpAL6X4L7mf4yXQSm+m/NsJLhp6afbFpPvXgYWC
wq4pvOtvxgum+FHrE0gysNjbKkE4Y6ixC9PGGAKHnhM3vrmvFjnl1G6KtyEz58Lz
BWW39QS6NJ1LqqP1nT+y3+Xzvfv8Eonqzno8mglCWMX09SguLmWoU1PcZQIDAQAB
oyAwHjALBgNVHQ8EBAMCBaAwDwYDVR0RBAgwBocECQNhhzANBgkqhkiG9w0BAQUF
AOBgQA6bgp4Zay34/fyAlyCkNNAYJRrN3Vc4NHN7IGjUziN6jK5UyB5zL37FERW
hT9ArPLzK7yEZs+MDNvB0bosyGWEDYPZr7EZHhYcoBP4/cd0V5rBFmA8Y2gUthPi
Ioxpi4+KZGHYyLqTrm+8Is/DVJaQmCGRPynHK35xjT6WuQtiYg==
-----END CERTIFICATE-----
```

The following options can help you diagnose and solve this problem.

- If data was lost or corrupted, recreate the Certificate
- Use an ASN.1 parser (available on the Internet World Wide Web) to check whether the certificate is valid by parsing the certificate successfully.

Error: Key Manager gives the following error when receiving a personal certificate:

```
No request key was found for the certificate
```

Problem: A Personal Certificate Request does not exist for the personal certificate being received.

To Fix: Create the Personal Certificate Request again and request a new certificate.

Error: Web-based System Manager gives the following error when you configure an IKE tunnel:

```
Error 171 in the Key Management (Phase 1) Tunnel operation:
PUT_IRL_FAILED
```

Problem: One cause for this error is that the host identity type, which is configured on the IKE dialog (Identification tab), is invalid. This can happen when the host identity type selected from the pull-down list does not logically match the type entered in the `Host Identity` field. For example, if you select a host identity type of **X500 Distinguished Name**, you must to enter a properly formatted distinguished name in the `Host Identity` field.

To Fix: Ensure the distinguished name you enter is correct for the type selected in the host identity pull-down list.

Error: An IKE negotiation fails and an entry similar to the following appears in the log file:

```
inet_cert_service::channelOpen():clientInitIPC():error,rc =2
(No such file or directory)
```

Problem: The **cpsd** is not running or has died

To Fix: Start IP Security using Web-based System Manager. This action also starts the appropriate daemons.

Error: An IKE negotiation fails and an entry similar to the following appears in the log file:

```
CertRepo::GetCertObj:  DN Does Not Match:  ("/C=US/O=IBM/CN=ripple.austin.ibm.com")
```

Problem: The X.500 Distinguished Name (DN) entered while defining the IKE tunnel does not match the X.500 DN in the personal certificate.

To Fix: Change the IKE tunnel definition in Web-based System Manager to match the distinguished name in the certificate.

Error: While defining IKE tunnels in Web-based System Manager, the Digital certificate check box is disabled under the Authentication Method tab.

Problem: The policy associated with this tunnel does not use RSA signature mode authentication.

To Fix: Change the transform of the associated policy to use the RSA signature authentication method. For example, you can choose *IBM_low_CertSig* as a key management policy when defining a IKE tunnel.

Tracing Facilities

Tracing is a debug facility for tracing kernel events. Traces can be used to get more specific information about events or errors occurring in the kernel filter and tunnel code.

The SMIT IP Security trace facility is available through the Advanced IP Security Configuration menu. The information captured by this trace facility includes information on Error, Filter, Filter Information, Tunnel, Tunnel Information, Capsulation/Decapsulation, Capsulation Information, Crypto, and Crypto Information. By design, the error trace hook provides the most critical information. The info trace hook can generate critical information and may have an impact on system performance. This tracing will provide clues as to what a problem may be. Tracing information will also be required when speaking with a service technician. To access the tracing facility, use the SMIT fast path **smit ips4_tracing** (for IP Version 4) or **smit ips6_tracing** (for IP Version 6).

ipsecstat

You can issue the **ipsecstat** command to generate the following sample report. This sample report shows that the IP Security devices are in the available state, that there are three authentication algorithms installed, three encryption algorithms installed, and that there is a current report of packet activity. This information may be useful to you in determining where a problem exists if you are troubleshooting your IP Security traffic.

```
IP Security Devices:
ipsec_v4 Available
ipsec_v6 Available

Authentication Algorithm:
HMAC_MD5 -- Hashed MAC MD5 Authentication Module
HMAC_SHA -- Hashed MAC SHA Hash Authentication Module
KEYED_MD5 -- Keyed MD5 Hash Authentication Module

Encryption Algorithm:
CDMF -- CDMF Encryption Module
DES_CBC_4 -- DES CBC 4 Encryption Module
DES_CBC_8 -- DES CBC 8 Encryption Module
3DES_CBC -- Triple DES CBC Encryption Module
```

```
IP Security Statistics -
Total incoming packets:  1106
Incoming AH packets:326
Incoming ESP packets:  326
Srcrte packets allowed:  0
Total outgoing packets:844
Outgoing AH packets:527
Outgoing ESP packets:  527
Total incoming packets dropped:  12
  Filter denies on input:  12
  AH did not compute: 0
  ESP did not compute:0
  AH replay violation:0
  ESP replay violation:  0
Total outgoing packets dropped:0
  Filter denies on input:0
Tunnel cache entries added: 7
Tunnel cache entries expired:  0
Tunnel cache entries deleted:  6
```

Note: Beginning with AIX 4.3.3, CDMF support has been removed because DES is now available worldwide. Reconfigure any tunnels that use CDMF to use DES or Triple DES.

IP Security Reference

List of Commands

ike cmd=activate	Starts an Internet Key Exchange (IKE) negotiation (AIX 4.3.2 and later).
ike cmd=remove	Deactivates IKE tunnels (AIX 4.3.2 and later)
ike cmd=list	Lists IKE tunnels (AIX 4.3.2 and later)
ikedb	Provides the interface to the IKE tunnel database(AIX 5.1 and later)
gentun	Creates a tunnel definition
mktun	Activates tunnel definition(s)
chtun	Changes a tunnel definition
rmtun	Removes a tunnel definition
lstun	Lists tunnel definition(s)
exptun	Exports tunnel definition(s)
imptun	Imports tunnel definition(s)
genfilt	Creates a filter definition
mkfilt	Activates filter definition(s)
mvfilt	Moves a filter rule
chfilt	Changes a filter definition
rmfilt	Removes a filter definition
lsfilt	Lists filter definition(s)
expfilt	Exports filter definition(s)
impfilt	Imports filter definition(s)
ipsec_convert	Lists status of IP security
ipsecstat	Lists status of IP security
ipsectrcbuf	Lists the contents of IP security tracing buffer
unloadipsec	Unloads a crypto module

List of Methods

defipsec	Defines an instance of IP Security for IP Version 4 or IP Version 6
cfgipsec	Configures and loads **ipsec_v4** or **ipsec_v6**
ucfgipsec	Unconfigures **ipsec_v4** or **ipsec_v6**

Chapter 12. Network Information Services (NIS) and NIS+ Security

This chapter gives an overview of how NIS+ protects its namespace. The chapter includes the following sections:

Operating System Security Mechanisms

Operating system security is provided by gates that users must pass through before entering the operating system environment, and permission matrixes that determine what they are able to do once inside. In some contexts, *secure RPC* passwords have been referred to as *network passwords*.

The overall system is composed of four gates and two permission matrixes:

Dialup gate
: To access a given operating system environment from the outside through a modem and phone line, you must provide a valid login ID and dial-up password.

Login gate
: To enter a given operating system environment you must provide a valid login ID and user password.

Root gate
: To gain access to root privileges, you must provide a valid root user password.

Secure RPC gate
: In an NIS+ environment running at security level 2 (the default), when you try to use NIS+ services and gain access to NIS+ objects (servers, directories, tables, table entries, and so on) your identity is confirmed by NIS+, using the secure RPC process.

  Entering the secure RPC gate requires presentation of a secure RPC password. Your secure RPC password and your login password normally are identical. When that is the case, you are passed through the gate automatically without having to re-enter your password. (In some contexts, *secure RPC* passwords have been referred to as *network passwords*. See the Secure RPC Password versus Login Password section in the *AIX 5L Version 5.2 Network Information Services (NIS and NIS+) Guide* for information about handling two passwords that are not identical.)

  A set of *credentials* is used to automatically pass your requests through the secure RPC gate. The process of generating, presenting, and validating your credentials is called *authentication* because it confirms who you are and that you have a valid secure RPC password. This authentication process is automatically performed every time you request NIS+ service.

  In an NIS+ environment running in NIS-compatibility mode, the protection provided by the secure RPC gate is significantly weakened because everyone has read rights for all NIS+ objects and modify rights for those entries that apply to them regardless of whether or not they have a valid credential (that is, regardless of whether or not the authentication process has confirmed their identity and validated their secure RPC password). Because this situation allows *anyone* to have read rights for all NIS+ objects and modify rights for those entries that apply to them, an NIS+ network running in compatibility mode is less secure than one running in normal mode. (In secure

RPC terminology, any user without a valid credential is considered a member of the **nobody** class. See “Authorization Classes” on page 194 for a description of the four classes.)

For details on how to administer NIS+ authentication and credentials, see the Administering NIS+ Credentials section in the *AIX 5L Version 5.2 Network Information Services (NIS and NIS+) Guide*.

File and directory matrix

Once you have gained access to an operating system environment, your ability to read, execute, modify, create, and destroy files and directories is governed by the applicable permissions.

NIS+ objects matrix

Once you have been properly authenticated to NIS+, your ability to read, modify, create, and destroy NIS+ objects is governed by the applicable permissions. This process is called *NIS+ authorization*.

For details on NIS+ permissions and authorization, see the Administering NIS+ Access Rights section in the *AIX 5L Version 5.2 Network Information Services (NIS and NIS+) Guide*.

NIS+ Security Mechanisms

NIS+ security is an integral part of the NIS+ namespace. You cannot set up security independently from the namespace. For this reason, instructions for setting up security are woven through the steps used to set up the other components of the namespace. Once an NIS+ security environment has been set up, you can add and remove users, change permissions, reassign group members, and perform all other routine administrative tasks needed to manage an evolving network.

The security features of NIS+ protect the information in the namespace, as well as the structure of the namespace itself, from unauthorized access. Without these security features, any NIS+ client could obtain, change, or even damage information stored in the namespace.

NIS+ security serves two purposes:

Authentication
Authentication is used to identify NIS+ principals. Every time a principal (either user or machine) tries to access an NIS+ object, the user's identity and secure RPC password is confirmed and validated. (You should not have to enter a password as part of the authentication process. However, if for some reason your secure RPC password is different from your login password, you must perform a **keylogin** the first time you try accessing NIS+ objects or services. To perform a **keylogin**, you must provide a valid secure RPC password. See the Secure RPC Password versus Login Password section in the *AIX 5L Version 5.2 Network Information Services (NIS and NIS+) Guide*.)

Authorization
Authorization is used to specify access rights. Every time NIS+ principals try to access NIS+ objects, they are placed in one of four authorization classes (owner, group, world, nobody). The NIS+ security system allows NIS+ administrators to specify different read, modify, create, or destroy rights to NIS+ objects for each class. For example, a given class could be permitted to modify a particular column in the passwd table but not read that column, or a different class could be allowed to read some entries of a particular table but not others.

For example, a given NIS+ table may allow one class to both read and modify the information in the table, but a different class is only allowed to read the information, and a third class is not even allowed to do that. This is similar in concept to the operating system's file and directory permissions system. (See "Authorization Classes" on page 194 for more information on classes.)

Authentication and authorization prevents someone with root privileges on machine A from using the **su** command to assume the identity of a second user who is either not logged in at all or logged in on machine B, and then accessing NIS+ objects with the second user's NIS+ access privileges.

Note, however, that NIS+ cannot prevent someone who knows another user's login password from assuming that other user's identity and NIS+ access privileges. Nor can NIS+ prevent a user with root privileges from assuming the identity of another user who is logged in from the *same* machine.

The following figure details this process.

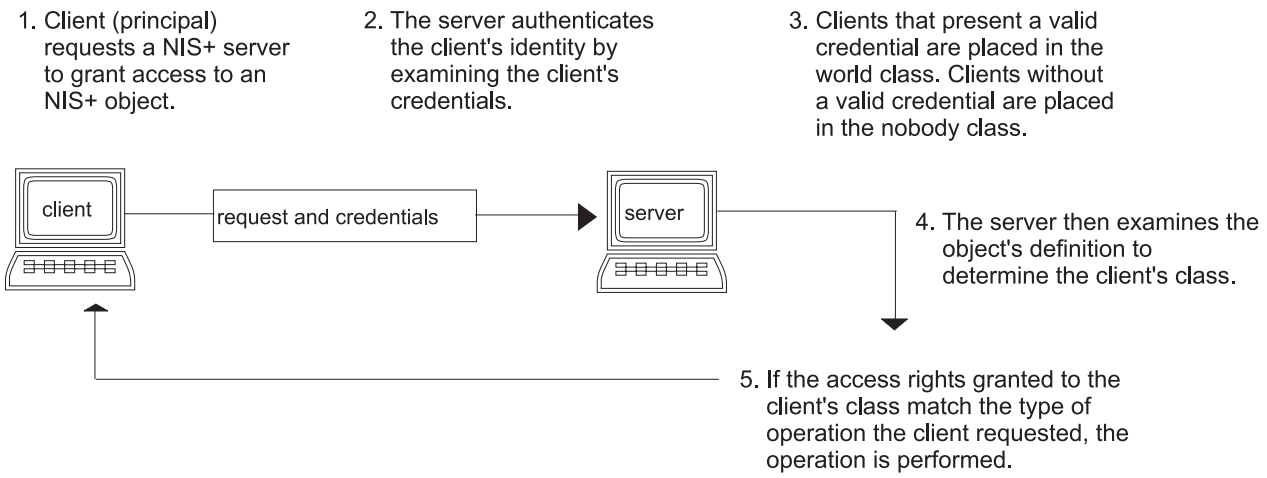


Figure 13. Summary of NIS+ Security Process. This illustration shows a representation of the NIS+ security process.

1. *The client/principal requests an NIS+ server to grant access to an NIS+ object.*
2. *The server authenticates the client's identity by examining the client's credentials.*
3. *The clients with valid credentials are placed in the world class.*
4. *The clients without valid credentials are placed in the nobody class.*
5. *The server examines the object's definition to determine the client's class.*
6. *If the access rights granted to the client's class match the type of operation requested, the operation is performed.*

NIS+ Principals

NIS+ principals are the entities (clients) that submit requests for NIS+ services. An NIS+ principal may be someone who is logged in to a client machine as a regular user, someone who is logged in as root user, or any process that runs with root user permission on an NIS+ client machine. Thus, an NIS+ principal can be a client user or a client workstation.

An NIS+ principal can also be the entity that supplies an NIS+ service from an NIS+ server. Because all NIS+ servers are also NIS+ clients, much of this discussion also applies to servers.

NIS+ Security Levels

NIS+ servers operate at one of two security levels. These levels determine the types of credential principals must submit for their requests to be authenticated. NIS+ is designed to run at the most secure level, which is security level 2. Level 0 is provided only for testing, setup, and debugging purposes. These security levels are summarized in the following table.

> **Note:** Use Web-based System Manager, SMIT, or the **passwd** command to change your own password regardless of security level or credential status.

NIS+ security levels

Severity level	Description
0	Security level 0 is designed for testing and setting up the initial NIS+ namespace. An NIS+ server running at security level 0 grants any NIS+ principal full access rights to all NIS+ objects in the domain. Level 0 is for setup purposes only and should only be used by administrators for that purpose. Level 0 should *not* be used on networks in normal operation by regular users.
1	Security level 1 uses AUTH_SYS security. This level is not supported by NIS+ and should *not* be used.

NIS+ security levels

Severity level	Description
2	Security level 2 is the default. The highest level of security currently provided by NIS+, it authenticates only requests that use data encryption standard (DES) credentials. Requests with no credentials are assigned to the nobody class and have whatever access rights have been granted to that class. Requests that use invalid DES credentials are retried. After repeated failure to obtain a valid DES credential, requests with invalid credentials fail with an authentication error. (A credential might not be valid for a variety of reasons, such as the principal making the request is not logged in through **keylogin** on that machine, the clocks are out of sync, there is a key mismatch, and so on.)

NIS+ Authentication and Credentials

NIS+ credentials authenticate the identity of each principal requesting an NIS+ service or access to an NIS+ object. The NIS+ credential/authorization process is an implementation of the Secure RPC system.

The credential/authentication system prevents someone from assuming another's identity. That is, it prevents someone with root privileges on one machine from using the **su** command to assume the identity of a second user who is either not logged in at all or logged in on another machine and then accessing NIS+ objects with the second user's NIS+ access privileges.

> **Note:** NIS+ cannot prevent someone who knows another user's login password from assuming that other user's identity and the other user's NIS+ access privileges. Nor can NIS+ prevent a user with root privileges from assuming the identity of another user who is currently logged in on the *same* machine.

Once a server authenticates a principal, it then checks the NIS+ object that the principal wants to access to verify what operations that principal is authorized to perform. (See "NIS+ Authorization and Access" on page 194 for further information on authorization.)

User and Machine Credentials

There are two basic types of principal, *users* and *machines,* and thus two different types of credentials:

User credentials
: When someone is logged in to an NIS+ client as a regular user, requests for NIS+ services include that person's user credentials.

Machine credentials
: When a user is logged in to an NIS+ client as root user, request for services use the client workstation's credentials.

DES versus Local Credentials

NIS+ principals can have two types of credential: DES and local.

DES Credentials

Data Encryption Standard (DES) credentials provide secure authentication. When this guide refers to NIS+ checking a credential to authenticate an NIS+ principal, it is the DES credential that NIS+ is validating. (Note that using DES credentials is only one method of achieving authentification. Do not equate DES credentials with NIS+ credentials.)

Each time a principal requests an NIS+ service or access to an NIS+ object, the software uses the credential information stored for that principal to generate a credential for that principal. DES credentials are generated from information created for each principal by an NIS+ administrator, as explained in the Administering NIS+ Credentials section in the *AIX 5L Version 5.2 Network Information Services (NIS and NIS+) Guide*.

- When the validity of a principal's DES credential is confirmed by NIS+, that principal is *authenticated*.
- A principal must be authenticated before being placed in the owner, group, or world authorization classes. In other words, you must have a valid DES credential in order to be placed in one of those classes. (Principals without a valid DES credential are automatically placed in the nobody class.)
- DES credential information is always stored in the cred table of the principal's home domain, regardless of whether that principal is a client user or a client workstation.

Local Credentials

Local credentials are a map between a user's user ID number and their NIS+ principal name which includes their home domain name. When users log in, the system looks up their local credential, which identifies their home domain where their DES credential is stored. The system uses that information to get the user's DES credential information.

When users log in to a remote domain, those requests use their local credential which points back to their home domain. NIS+ then queries the user's home domain for that user's DES credential information. This allows a user to be authenticated in a remote domain even though the user's DES credential information is not stored in that domain. The following figure illustrates this concept.

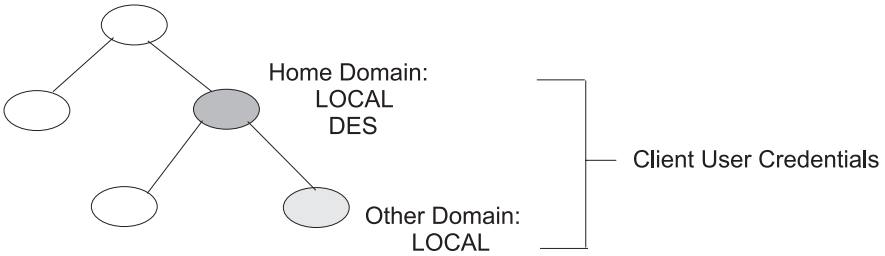


Figure 14. Credential and Domains. This illustration shows a domain hierarchy. The user's home domain has local and DES credentials. The subdomain only has local credentials. Home and the subdomain are labeled Client User Credentials.

Credentials and Domains: Local credential information can be stored in any domain. To log into a remote domain and be authenticated, a client user *must* have a local credential in the cred table of the remote domain. If a user does not have a local credential in a remote domain the user is trying to access, NIS+ cannot locate the user's home domain to obtain the user's DES credential. In such a case, the user would not be authenticated and would be placed in the nobody class.

User Types and Credential Types

A user can have both types of credential, but a machine can *only* have a DES credential.

Root cannot have NIS+ access, as root, to other machines because the root UID of every machine is always zero. If root (UID=0) of machine A tried to access machine B as root, that would conflict with machine B's already existing root (UID=0). Thus, a local credential is not appropriate for a client *workstation*; it is allowed only for a client *user*.

NIS+ Authorization and Access

The basic purpose of NIS+ authorization is to specify the access rights that each NIS+ principal has for each NIS+ object and service.

Once the principal making an NIS+ request is authenticated, NIS+ places that principal in an authorization class. The access rights (permissions) that specify which operations a principal may do with a given NIS+ object are assigned on a class basis. In other words, one authorization class may have certain access rights while a different class has different rights.

Authorization classes
There are four authorization classes: owner, group, world, and nobody. (See "Authorization Classes" for details.)

Access rights
There are four types of access rights (permissions): create, destroy, modify, and read. (See "NIS+ Access Rights" on page 196 for details.)

Authorization Classes

NIS+ objects do not grant access rights directly to NIS+ principals. Instead, they grant access rights to four *classes* of principal:

Owner
The principal who happens to be the object's owner gets the rights granted to the owner class.

Group Each NIS+ object has one group associated with it. The members of an object's group are specified by the NIS+ administrator. The principals who belong to the object's group class get the rights granted to the group class. (In this context, *groups* refers to NIS+ groups, and not operating system or net groups. See "Group Class" on page 195 for a description of NIS+ groups.)

World The world class encompasses all NIS+ principals that a server has been able to authenticate. (That is, everyone who has been authenticated but who is not in either the owner or group classes.)

Nobody
All principals belong to the nobody class, including those who are not authenticated.

See the following figure for an illustration of the classes.

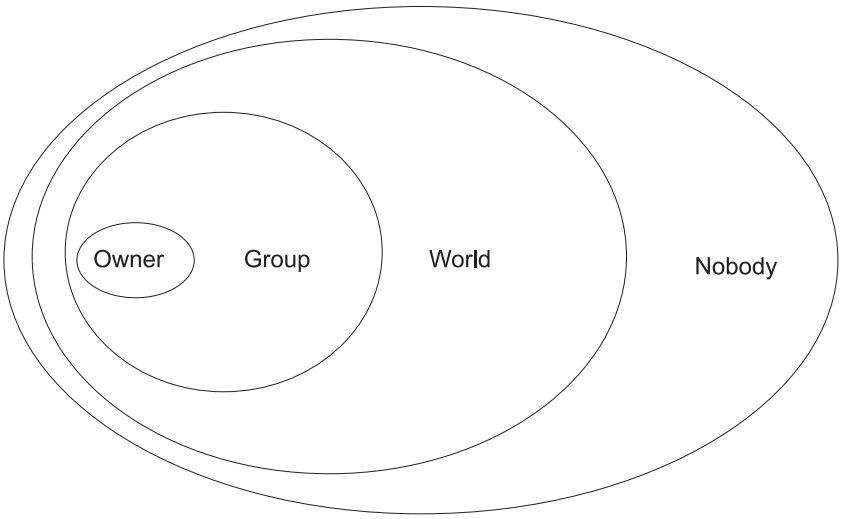


Figure 15. Authorization Classes. This illustration shows a series of ovals within ovals that represents the relationship between authorization classes. The smallest oval is Owner, encompassed by a larger oval labeled Group, encompassed by an oval labeled World, encompassed by an oval labeled Nobody.

For any NIS+ request, the system determines which class the requesting principal belongs to and the principal can then use whatever access rights belong to that class.

An object can grant any combination of access rights to each of these classes. Normally, however, a higher class is assigned the same rights as all the lower classes, as well as possible additional rights.

For instance, an object could grant read access to the nobody and world classes, both read and modify access to the group class, and read, modify, create, and destroy access to the owner class.

The four classes are described in detail, as follows.

Owner Class

The owner is a *single* NIS+ principal.

A principal making a request for access to an NIS+ object must be authenticated (present a valid DES credential) before being granted owner-access rights.

By default, an object's owner is the principal that created the object. However, an object's owner can cede ownership to another principal by two different methods:

- The principal specifies a different owner at the time the object is created (see the Specifying Access Rights in Commands section in the *AIX 5L Version 5.2 Network Information Services (NIS and NIS+) Guide*).
- The principal changes the ownership of the object after it is created (see the Changing Ownership of Objects and Entries section in the *AIX 5L Version 5.2 Network Information Services (NIS and NIS+) Guide*).

Once a principal gives up ownership, that principal gives up all owner's access rights to the object and keeps only the rights the object assigns to either the group, the world, or nobody.

Group Class

The object's group is a *single* NIS+ group. (In this context, *group* refers to NIS+ groups, and not operating system or net groups.)

A principal making a request for access to an NIS+ object must be authenticated (present a valid DES credential) and belong to the group before being granted group-access rights.

An NIS+ group is a collection of NIS+ principals, grouped together as a convenience for providing access to the namespace. The access rights granted to an NIS+ group apply to all the principals that are members of that group. (An object's owner, however, does not need to belong to the object's group.)

When an object is created, the creator can opt for a default group. A nondefault group can be specified either when the object is created or at any time later.

Information about NIS+ groups is stored in NIS+ group **objects**, under the **groups_dir** subdirectory of every NIS+ domain. (Note that information about NIS+ groups is not stored in the NIS+ group table. That table stores information about operating system groups.) Instructions for administering NIS+ groups are provided in the Administering NIS+ Groups section in the *AIX 5L Version 5.2 Network Information Services (NIS and NIS+) Guide*.

World Class

The world class contains all NIS+ principals that are authenticated by NIS+; that is, all members in the owner and group class, as well as all other principals who present a valid DES credential.

Access rights granted to the world class apply to all authenticated principals.

Nobody Class

The nobody class contains all principals, even those without a valid DES credential.

Authorization Classes and the NIS+ Object Hierarchy

NIS+ security applies authorization classes independently to a hierarchy of objects. Directory objects are the top level of the default hierarchy, then group or table objects, then columns, then entries. The following definitions provide more information about each level:

Directory level
: Each NIS+ domain contains two NIS+ directory objects: **groups_dir** and **org_dir**. Each **groups_dir** directory object contains various groups. Each **org_dir** directory object contains various tables.

Group or table level
: Groups contain individual entries and possibly other groups. Tables contain both columns and individual entries.

Column level
: Each table has one or more columns.

Entry (row) level
: Each group or table has one or more entries.

The four authorization classes apply at each level. Thus, a directory object has an owner and a group. Each table within a directory object has its own owner and group, which can be different from the owner and group of the directory object. Within a table, a column or an entry can have its own owner or group, which can be different from the owner and group of the table as a whole or of the directory object as a whole.

NIS+ Access Rights

NIS+ objects specify access rights for NIS+ principals in the same way that operating system files specify permissions for operating system users. Access rights specify the types of operations that NIS+ principals are allowed to perform on NIS+ objects. (You can examine these by using the **niscat -o** command.)

NIS+ operations vary among different types of objects, but all operations fall into one of the four access-rights categories: read, modify, create, and destroy.

Read A principal with read rights to an object can view the contents of that object.

Modify
A principal with modify rights to an object can change the contents of that object.

Destroy
A principal with destroy rights to an object can destroy or delete the object.

Create
A principal with create rights to a higher level object can create new objects within that level. If you have create rights to a NIS+ directory object, you can create new tables within that directory. If you have create rights to a NIS+ table, you can create new columns and entries within that table.

Every communication from an NIS+ client to an NIS+ server is a request to perform one of these operations on a specific NIS+ object. For instance, when an NIS+ principal requests the IP address of another workstation, it is effectively requesting read access to the **hosts** table object, which stores that type of information. When a principal asks the server to add a directory to the NIS+ namespace, it is actually requesting **modify** access to the directory's parent object.

These rights logically evolve down from directory to table to table column and entry levels. For example, to create a new table, you must have create rights for the NIS+ directory object where the table will be stored. When you create that table, you become its default owner. As owner, you can assign yourself create rights to the table which allows you to create new entries in the table. If you create new entries in a table, you become the default owner of those entries. As table owner, you can also grant table-level create rights to others. For example, you can give your table's group class table-level create rights. In that case, any member of the table's group can create new entries in the table. The individual member of the group who creates a new table entry becomes the default owner of that entry.

NIS+ Security and Administrative Rights

NIS+ does not enforce any requirement that there be a single NIS+ administrator. Whoever has administrative rights over an object—that is, the authority to create, destroy, and for some objects, modify rights—is considered to be an NIS+ administrator for that object.

Whoever creates an NIS+ object sets the initial access rights to that object. If the creator restricts administrative rights to the object's owner (initially the creator), then only the owner has administrative power over that object. On the other hand, if the creator grants administrative rights to the object's group, then everyone in that group has administrative power over that object.

Theoretically, you could grant administrative rights to the world class, or even the nobody class. The software allows you to do that. But granting administrative rights beyond the group class effectively nullifies NIS+ security. Thus, if you grant administrative rights to either the World or the nobody class you are, in effect, defeating the purpose of NIS+ security.

NIS+ Security Reference

Use the following commands to administer passwords, credentials, and keys (see the appropriate command descriptions for more information):

chkey Changes a principal's secure RPC key pair. Unless you want to re-encrypt your current private key with a new password, use the **passwd** command instead. The **chkey** command does not affect the principal's entry either in the passwd table or **/etc/passwd** file.

keylogin
Decrypts and stores a principal's secret key with the **keyserv**.

keylogout
Deletes stored secret key from **keyserv**.

keyserv
Enables the server for storing private encryption keys.

newkey
Creates a new key pair in public-key database.

nisaddcred
Creates credentials for NIS+ principals.

nisupdkeys
Updates public keys in directory objects.

passwd
Changes and administers principal's password.

Chapter 13. Network File System (NFS) Security

In addition to the standard UNIX authentication system, the Network File System (NFS) provides a means to authenticate users and machines in networks on a message-by-message basis. This additional authentication system uses Data Encryption Standard (DES) encryption and public key cryptography.

This chapter discusses the following topics:

Secrecy

Throughout history, various groups of people have sought to communicate in such a way that only the sender and receiver know the contents of a given message. To achieve this secrecy, senders and receivers use a *cipher*, a scheme for converting a *plaintext* message into *ciphertext* and back again. *Encryption* is the process of converting plain text into cipher text, and *decryption* is the process of converting cipher text into plain text.

One of the earliest ciphers, the *Caesar cipher*, is attributed to Julius Caesar. In this cipher, one letter is substituted for another. For example, 'A' becomes 'C', 'B' becomes 'D', ..., 'Y' becomes 'A', and 'Z' becomes 'B'. So, the Caesar cipher encrypts the phrase **ATTACK AT DAWN** as **CVVCEM CV FCYP**.

If the Carthaginians could *cryptanalyze* the Caesar cipher and break it, the Roman *cryptographers* would have to invent an entirely new cipher. Since cipher development is an expensive process, the Romans might use a cipher *key* in order to get more use out of their cipher. For example, instead of specifying a letter-for-letter substitute, the Romans might specify a key *K*, where K indicates the number of positions to shift a letter. That is, if K = 2, then 'A' becomes 'C'. If K = 4, then 'A' becomes 'E', and so on. With this scheme, if the Carthaginians break the cipher, all the Romans must do is change the key. Of course, the Carthaginians might realize what algorithm the Italians were using, and exhaustively try every value of K from 1 to 26. If the Carthaginians had a computer, their task would be a trivial programming exercise.

Data Encryption Standard

Modern ciphers are designed to address the fact that computers can be powerful tools for an intruder attempting to break a cipher. In 1977, the U.S. government adopted a cipher as its Data Encryption Standard. This cipher is widely used in industry. DES is a highly complex algorithm which converts 64-bit blocks of plain text into 64-bit blocks of cipher text using a 56-bit key. Because of the complexity of the algorithm and the size of the cipher key, DES is essentially unbreakable. For example, if an intruder had a computer which could compute the DES algorithm at a rate of one key per microsecond, the computer would need over two thousand years to try every possible key.

Public Key Cryptography

A significant weakness in any encryption algorithm is the key that is used. If the sender and receiver are to communicate securely using a cipher, both the sender and receiver must know the key. They must agree upon a key either using a separate communications link, which itself must be secure, or in person.

To address this problem, two researchers (Diffie and Hellman) developed a technique by which the sender and receiver can exchange keys publicly without compromising the security of their communication. Their technique has three requirements:

- Decipher(Encipher(plaintext, E), D) = plaintext

  where E is the encryption key (known to the public), and D is the decryption key (known only by the receiver).

  That is, Encipher and Decipher functions are inverses of each other. Therefore, if you take the encrypted text string returned by Encipher(plaintext, E), and use it along with the key D for the Decipher function, Decipher returns the original plain text.
- An intruder cannot deduce Decipher() from Encipher().
- Encipher() is unbreakable.

The following outline describes how a sender sends a secret message to a receiver.

1. The sender obtains the receiver public encryption key.
2. The sender converts the plain text message into cipher text by computing the result:

   ```
   ciphertext = Encipher( plaintext, E )
   ```
3. The sender sends the cipher text message to the receiver.
4. The receiver receives the cipher text message and converts it into plaintext by computing the result:

   ```
   plaintext = Decipher( ciphertext, D )
   ```

Even if an intruder intercepts the message, the intruder will not be able to decipher it because the intruder does not have the decipher key. (For that matter, the sender cannot decipher the cipher text message either.)

Authentication

A primary application of secrecy is *authentication*. One common method of authentication (which is the standard UNIX method of authentication) uses a password. When a user wants to log in, the operating system requires the user to provide a password known only by the operating system and the user. If the user provides the correct password, the operating system concludes that the user is who the user claims to be. Note that this method requires the operating system to store the user passwords in a file on the system, although in encrypted form. This means that two different entities know a single password.

Public key cryptography provides an alternative to password authentication. Suppose a sender wants to send a message, and the receiver wants to be certain that the message is from the sender and not from an intruder pretending to be the sender. The authentication process will occur in the following manner:

1. First, the sender enciphers a ″request to send″ message using the public key of the receiver, and then sends the request.
2. The receiver receives the ″request to send″ message and deciphers it using the receiver private key.
3. The receiver enciphers a ″token″ message using the public key of the sender, and then sends the token.
4. The sender receives the token and deciphers it with the sender private key. When the sender sends messages to the receiver, the sender will begin each message with the token, signifying that the sender is, in fact, the sender. If an intruder attempts to send messages in the name of the sender, the receiver will reject them because the intruder does not know what the token is.

Note that, unlike password authentication, the receiver is able to authenticate the sender without knowing the sender private key. For more information on authentication systems, see Understanding RPC Authentication in *AIX 5L Version 5.2 Communications Programming Concepts*.

NFS Authentication

NFS uses the DES algorithm for different purposes. NFS uses DES to encrypt a time stamp in the Remote Procedure Call (RPC) messages sent between NFS servers and clients. This encrypted time stamp authenticates machines just as the ″token″ authenticates the sender.

Because NFS can authenticate every single RPC message exchanged between NFS clients and servers, this provides an additional, optional, level of security for each file system. By default, file systems are exported with the standard UNIX authentication. To take advantage of this additional level of security, you can specify the **secure** option when you export a file system.

Public Key Cryptography for Secure NFS

Both the public key and the secret key of the user are are stored and indexed by net name in the **publickey.byname** map. The secret key is DES-encrypted with the user login password. The **keylogin** command uses the encrypted secret key, decrypts it with the login password, then gives it to a secure local key server to save for use in future RPC transactions. Users are not aware of their public and secret keys because the **yppasswd** command, in addition to changing the login password, generates the public and secret keys automatically.

The **keyserv** daemon is an RPC service that runs on each NIS and NIS+ machine. For information on how NIS+ uses **keyserv**, see *AIX 5L Version 5.2 Network Information Services (NIS and NIS+) Guide*. Within NIS, **keyserv** executes the following three public key subroutines:

- **key_setsecret** subroutine
- **key_encryptsession** subroutine
- **key_decryptsession** subroutine.

The **key_setsecret** subroutine tells the key server to store the secret key of the user (SK_A) for future use; it is normally called by the **keylogin** command. The client program calls the **key_encryptsession** subroutine to generate the encrypted conversation key, which is passed in the first RPC transaction to a server. The key server looks up the server public key and combines it with the secret key of the client (set up by a previous **key_setsecret** subroutine) to generate the common key. The server asks the key server to decrypt the conversation key by calling the **key_decryptsession** subroutine.

Implicit in these subroutine calls is the name of the caller, which must be authenticated in some manner. The key server cannot use DES authentication to do this, because it would create a deadlock. The key server solves this problem by storing the secret keys by the user ID (UID) and only granting requests to local root processes. The client process then executes a root user owned **setuid** subroutine which makes the request on the part of the client, telling the key server the real UID of the client.

Authentication Requirements

Secure NFS authentication is based on the ability of a sender to encrypt the current time, which the receiver can then decrypt and check against its own clock. This process has two requirements:

- The two agents must agree on the current time.
- The sender and receiver must be using the same DES encryption key.

Agreeing on the Current Time

If the network uses time synchronization, then the **timed** daemon keeps the client and server clocks synchronized. If not, the client computes the proper time stamps based on the server clock. To do this, the

client determines the server time before starting the RPC session, and then computes the time difference between its own clock and that of the server. The client then adjusts its time stamp accordingly. If, during the course of an RPC session, the client and server clocks become unsynchronized to the point where the server begins rejecting the client requests, the client will redetermine the server time.

Using the Same DES Key

The client and server compute the same DES encryption key by using public key cryptography. For any client A and server B, there is a key that only A and B can deduce. This key is called the *common key*. The client derives the common key by computing the following formula:

$$K_{AB} = PK_B^{SK}A$$

where *K* stands for the *common Key*, *PK* stands for *Public Key*, and *SK* stands for *Secret Key*, and each of these keys is a 128-bit number. The server derives the same common key by computing the following formula:

$$K_{AB} = PK_A^{SK}B$$

Only the server and client can calculate this common key since doing so requires knowing one secret key or the other. Because the common key has 128 bits, and DES uses a 56-bit key, the client and server extract 56 bits from the common key to form the DES key.

Authentication Process

When a client wants to talk to a server, it randomly generates a key used for encrypting the time stamps. This key is known as the *conversation key* (*CK*). The client encrypts the conversation key using the DES common key (described in Authentication Requirements) and sends it to the server in the first RPC transaction. This process is illustrated in the following figure.

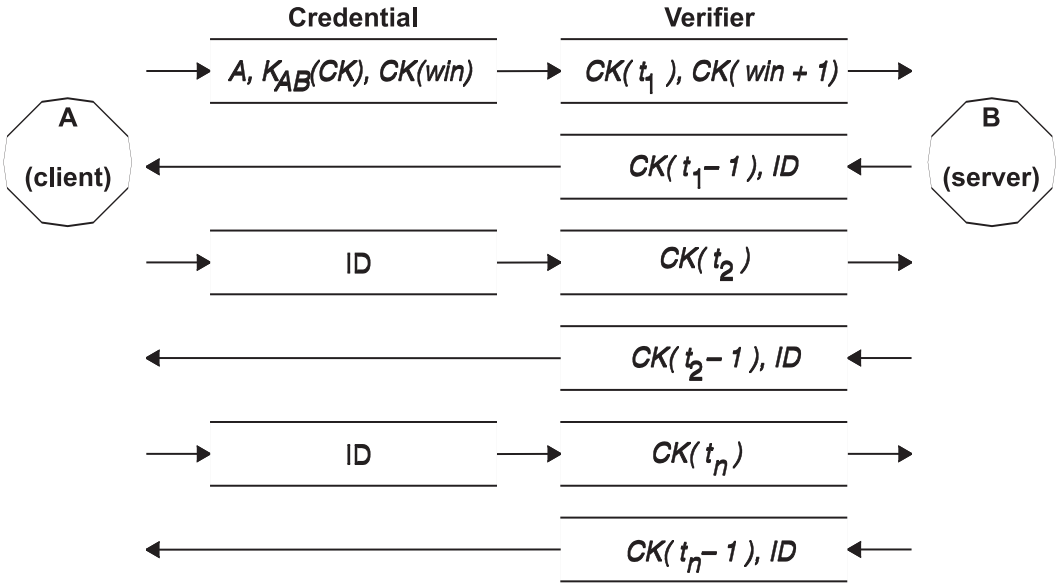


Figure 16. Authentication Process. This figure illustrates the authentication process which is described in the surrounding text.

This figure shows client A connecting to server B. The term *K*(*CK*) means *CK* is encrypted with the DES common key *K*. In its first request, the client RPC credential contains the client name (*A*), the conversation key (*CK*), and the variable called *win* (window) encrypted with *CK*. (The default window size is 30

minutes.) The client verifier in the first request contains the encrypted time stamp and an encrypted verifier of the specified window, *win* + 1. The window verifier makes guessing the right credential much more difficult, and increases security.

After authenticating the client, the server stores the following items in a credential table:

- Client name, *A*
- Conversation key, *CK*
- Window
- Time stamp.

The server only accepts time stamps that are chronologically greater than the last one seen, so any replayed transactions are guaranteed to be rejected. The server returns to the client in the verifier an index ID into the credential table, plus the client time stamp minus one, encrypted by *CK*. The client knows that only the server could have sent such a verifier, because only the server knows what time stamp the client sent. The reason for subtracting one from the time stamp is to ensure that it is not valid and cannot be reused as a client verifier. After the first RPC transaction, the client sends just its ID and an encrypted time stamp to the server, and the server sends back the client time stamp minus one, encrypted by *CK*.

Naming Network Entities for DES Authentication

DES authentication does its naming by using net names. The following paragraphs describe how NIS handles DES authentication. For information on how NIS+ handles DES authentication, see *AIX 5L Version 5.2 Network Information Services (NIS and NIS+) Guide*.

A *net name* is a string of printable characters to authenticate. The public and secret keys are stored on a per-net-name rather than a per-user-name basis. The **netid.byname** NIS map maps the net name into a local UID and group-access list.

User names are unique within each domain. Net names are assigned by concatenating the operating system and user ID with the NIS and Internet domain names. A good convention for naming domains is to append the Internet domain name (com, edu, gov, mil) to the local domain name.

Network names are assigned to machines as well as to users. A net name of a machine is formed much like that of a user. For example, a machine named `hal` in the `eng.ibm.com` domain has the net name `unix.hal@eng.ibm.com`. Correct authentication of machines is important for diskless machines that need full access to their home directories over the network.

To authenticate users from any remote domain, make entries for them in two NIS databases. One is an entry for their public and secret keys; the other is for their local UID and group-access list mapping. Users in the remote domain can then access all of the local network services, such as the NFS and remote logins.

The /etc/publickey File

The **/etc/publickey** file contains names and public keys, which NIS and NIS+ use to create the **publickey** map. The **publickey** map is used for secure networking. Each entry in the file consists of a network user name (which refers to either a user or a host name), followed by the user public key (in hexadecimal notation), a colon, and the user encrypted secret key (also in hexadecimal notation). By default, the only user in the **/etc/publickey** file is the user nobody.

Do not use a text editor to alter the **/etc/publickey** file because the file contains encryption keys. To alter the **/etc/publickey** file, use either the **chkey** or **newkey** commands.

Booting Considerations of Public Key Systems

When restarting a machine after a power failure, all of the stored secret keys are lost, and no process can access secure network services, such as mounting an NFS. Root processes could continue if there were someone to enter the password that decrypts the secret key of the root user . The solution is to store the root user decrypted secret key in a file that the key server can read.

Not all **setuid** subroutine calls operate correctly. For example, if a **setuid** subroutine is called by owner `A`, and owner `A` has not logged into the machine since it started, the subroutine cannot access any secure network services as `A`. However, most **setuid** subroutine calls are owned by the root user, and the root user secret key is always stored at startup time.

Performance Considerations of Secure NFS

Secure NFS affects system performance in the following ways:

- First, both the client and server must compute the common key. The time it takes to compute the common key is about one second. As a result, it takes about two seconds to establish the initial RPC connection, because both client and server have to perform this operation. After the initial RPC connection, the key server caches the results of previous computations, and so it does not have to recompute the common key every time.
- Each RPC transaction requires the folowing DES encryption operations:
  1. The client encrypts the request time stamp.
  2. The server decrypts it.
  3. The server encrypts the reply time stamp.
  4. The client decrypts it.

Because system performance is reduced by secure NFS, weigh the benefits of increased security against system performance requirements.

Checklist for Administering Secure NFS

Use the following checklist to help ensure that secure NFS operates properly:

- When mounting a file system with the **-secure** option on a client, the server name must match the server host name in the **/etc/hosts** file. If a name server is being used for host name resolution, make sure the host information returned by the name server matches the entry in the **/etc/hosts** file. Authentication errors result if these names do not match because the net names for machines are based on the primary entries in the **/etc/hosts** file and keys in the **publickey** map are accessed by net name.
- Do not mix secure and nonsecure exports and mounts. Otherwise, file access might be determined incorrectly. For example, if a client machine mounts a secure file system without the **secure** option or mounts an nonsecure system with the **secure** option, users have access as `nobody`, rather than as themselves. This condition also occurs if a user unknown to NIS or NIS+ attempts to create or modify files on a secure file system.
- Because NIS must propagate a new map after each use of the **chkey** and **newkey** commands, only use these commands when the network is lightly loaded.
- Do not delete the **/etc/keystore** file or the **/etc/.rootkey** file. If you reinstall, move, or upgrade a machine, save the **/etc/keystore** and **/etc/.rootkey** files.
- Instruct users to use the **yppasswd** command rather than the **passwd** command to change passwords. Doing so keeps passwords and private keys synchronized.

- Because the **login** command does not retrieve keys out of the **publickey** map for the **keyserv** daemon, the user must execute the **keylogin** command. You may want to place the **keylogin** command in each user **profile** file to execute the command automatically during login. Note that the **keylogin** command requires the user to enter their password again.
- When you generate keys for the root user at each host with either the **newkey -h** or **chkey** command, you must run the **keylogin** command to pass the new keys to the **keyserv** daemon. The keys are stored in the **/etc/.rootkey** file, which is read by the **keyserv** daemon each time the daemon is started.
- Periodically verify that the **yppasswdd** and **ypupdated** daemons are running on the NIS master server. These daemons are necessary for maintaining the **publickey** map.
- Periodically verify that that the **keyserv** daemon is running on all machines using secure NFS.

Configuring Secure NFS

To configure secure NFS on NIS master and slave servers, use the Web-based System Manager Network application or use the following procedure. For information on using NFS with NIS+ see *AIX 5L Version 5.2 Network Information Services (NIS and NIS+) Guide*.

1. On the NIS master server, create an entry for each user in the NIS **/etc/publickey** file using the **newkey** command. This command has the following options.
   - For a regular user, type:

     ```
     smit newkey
     ```

     or

     ```
     newkey -u username
     ```

     For a root user on a host machine, type:

     ```
     newkey -h hostname
     ```
   - Alternatively, users can establish their own public keys using the **chkey** or **newkey** commands.
2. Create the NIS **publickey** map by following the instructions in *AIX 5L Version 5.2 Network Information Services (NIS and NIS+) Guide*. The corresponding NIS **publickey.byname** map resides only on the NIS servers.
3. Uncomment the following stanzas in the **/etc/rc.nfs** file:

   ```
   #if [ -x /usr/sbin/keyserv ]; then
   #  startsrc -s keyserv
   #fi
   #if [ -x /usr/lib/netsvc/yp/rpc.ypupdated -a -d /etc/yp/`domainname` ]; then
   #  startsrc -s ypupdated
   #fi
   #DIR=/etc/passwd
   #if [ -x /usr/lib/netsvc/yp/rpc.yppasswdd -a -f $DIR/passwd ]; then
   #  startsrc -s yppasswdd
   #fi
   ```
4. Start the **keyserv**, **ypupdated**, and **yppasswdd** daemons by using the **startsrc** command.

To configure secure NFS on NIS clients, start the **keyserv** daemon by using the **startsrc** command.

Exporting a File System Using Secure NFS

You can export a secure NFS using the Web-based System Manager Network application or by using one of the following procedures.

- To export a secure NFS file system using SMIT, do the following:
  1. Verify that NFS is already be running by issuing the **lssrc -g nfs** command. The output should indicate that the **nfsd** and the **rpc.mountd** daemons are active.

2. Verify that the **publickey** map exists and that the **keyserv** daemon is running. See “Configuring Secure NFS” on page 207 for more information.
3. Run the **smit mknfsexp** fast path.
4. Specify the appropriate values for the PATHNAME of directory to export, MODE to export directory, and EXPORT directory now, system restart or both fields. Specify **yes** for the Use SECURE option field.
5. Specify any other optional characteristics, or accept the default values by leaving the remaining fields as they are.
6. Exit SMIT. If the **/etc/exports** file does not exist, then it will be created.
7. Repeat steps 3 through 6 for each directory you want to export.

- To export a secure NFS file system using a text editor, do the following:
  1. Open the **/etc/exports** file with your favorite text editor.
  2. Create an entry for each directory to be exported, using the full path name of the directory. List each directory to be exported starting in the left margin. No directory should include any other directory that is already exported. See the **/etc/exports** file documentation for a description of the full syntax for entries in the **/etc/exports** file, including how to specify the **secure** option.
  3. Save and close the **/etc/exports** file.
  4. If NFS is currently running, type:

     ```
     /usr/sbin/exportfs -a
     ```

     The **-a** option tells the **exportfs** command to send all information in the **/etc/exports** file to the kernel.
- To export an NFS file system temporarily (that is, without changing the **/etc/exports** file):

  Enter:

  ```
  exportfs -i -o secure /dirname
  ```

  where *dirname* is the name of the file system you want to export. The **exportfs -i** command specifies that the **/etc/exports** file is not to be checked for the specified directory, and all options are taken directly from the command line.

Mounting a File System Using Secure NFS

To mount a secure NFS directory explicitly, do the following:

1. Verify that the NFS server has exported the directory by issuing the command:

   ```
   showmount -e ServerName
   ```

   where *ServerName* is the name of the NFS server. This command displays the names of the directories currently exported from the NFS server. If the directory you want to mount is not listed, export the directory from the server.
2. Establish the local mount point using the **mkdir** command. For NFS to complete a mount successfully, a directory that acts as the mount point (or place holder) of an NFS mount must be present. This directory should be empty. This mount point can be created like any other directory, and no special attributes are needed.
3. Verify that the **publickey** map exists and that the **keyserv** daemon is running. For more information, see “Configuring Secure NFS” on page 207.
4. Enter:

   ```
   mount -o secure ServerName:/remote/directory /local/directory
   ```

   where *ServerName* is the name of the NFS server, */remote/directory* is the directory on the NFS server you want to mount, and */local/directory* is the mount point on the NFS client.

Note: Only the root user can mount a secure NFS.

Chapter 14. Enterprise Identity Mapping

Today's network environments are made up of a complex group of systems and applications, resulting in the need to manage multiple user registries. Dealing with multiple user registries quickly grows into a large administrative problem that affects users, administrators, and application developers. Enterprise Identity Mapping (EIM) allows administrators and application developers to easily address this problem.

This chapter describes the problems, outlines current industry approaches, and explains the EIM approach.

Managing Multiple User Registries

Many administrators manage networks that include different systems and servers, each with a unique way of managing users through various user registries. In these complex networks, administrators are responsible for managing each user's identities and passwords across multiple systems. Additionally, administrators often must synchronize these identities and passwords. Users are burdened with remembering multiple identities and passwords and with keeping them synchronized. Because user and administrator overhead in this environment is expensive, administrators often spend valuable time troubleshooting failed login attempts and resetting forgotten passwords instead of managing the enterprise.

The problem of managing multiple user registries also affects application developers who want to provide multiple-tier or heterogeneous applications. Customers have important business data spread across many different types of systems, with each system possessing its own user registries. Consequently, developers must create proprietary user registries and associated security semantics for their applications. Although this solves the problem for the application developer, it increases the overhead for users and administrators.

Current Approaches

Several current industry approaches for solving the problem of managing multiple user registries are available, but they all provide incomplete solutions. For example, Lightweight Directory Access Protocol (LDAP) provides a distributed user registry solution. However, to use solutions such as LDAP, administrators must manage yet another user registry and security semantics or replace existing applications that are built to use those registries.

Using this type of solution, administrators must manage multiple security mechanisms for individual resources, thereby increasing administrative overhead and potentially increasing the likelihood of security exposures. When multiple mechanisms support a single resource, the chances of changing the authority through one mechanism and forgetting to change the authority for one or more of the other mechanisms is much higher. For example, a security exposure can result when a user is appropriately denied access through one interface, but allowed access through one or more other interfaces.

After completing this work, administrators find that they have not completely solved the problem. Generally, enterprises have invested too much money in current user registries and in their associated security semantics to make using this type of solution practical. Creating another user registry and associated security semantics solves the problem for the application provider, but not the problems for users or administrators.

Another solution is to use a single sign-on approach. Several products are available that allow administrators to manage files that contain all of a user's identities and passwords. However, this approach has several weaknesses:

- It addresses only one of the problems that users face. Although it allows users to sign on to multiple systems by supplying one identity and password, the user is still required to have passwords on other systems, or the need to manage these passwords.

- It introduces a new problem by creating a security exposure because clear-text or decryptable passwords are stored in these files. Passwords should never be stored in clear-text files or be easily accessible by anyone, including administrators.
- It does not solve the problems of third-party application developers that provide heterogeneous, multiple-tier applications. They must still provide proprietary user registries for their applications.

Despite these weaknesses, some enterprises use these solutions because they provide some relief for the multiple user registry problems.

Using Enterprise Identity Mapping

The EIM architecture describes the relationships between individuals or entities (such as file servers and print servers) in the enterprise and the many identities that represent them within an enterprise. In addition, EIM provides a set of APIs that allow applications to ask questions about these relationships.

For example, given a person's user identity in one user registry, you can determine which identity in another user registry represents that same person. If the user has authenticated with one identity and you can map that identity to the appropriate identity in another user registry, the user does not need to provide credentials for authentication again. You need only know which identity represents that user in another user registry. Therefore, EIM provides a generalized identity-mapping function for the enterprise.

The ability to map between a user's identities in different registries provides many benefits. Primarily, applications can have the flexibility of using one registry for authentication while using an entirely different registry for authorization. For example, an administrator could map an SAP identity (or better yet, SAP could do the mapping itself) to access SAP resources.

Identity mapping requires that administrators do the following:

1. Create EIM identifiers that represent people or entities in their enterprise.
2. Create EIM registry definitions that describe the existing user registries in their enterprise.
3. Define the relationship between the user identities in those registries to the EIM identifiers that they created.

No code changes are required to existing registries. Mappings are not required for all identities in a user registry. EIM allows one-to-many mappings (in other words, a single user with more than one identity in a single user registry). EIM also allows many-to-one mappings (in other words, multiple users sharing a single identity in a single user registry, which although supported is not advised for security reasons). An administrator can represent any user registry of any type in EIM.

EIM does not require copying existing data to a new repository and trying to keep both copies synchronized. The only new data that EIM introduces is the relationship information. Administrators manage this data in an LDAP directory, which provides the flexibility of managing the data in one place and having replicas wherever the information is used.

For more information about Enterprise Identity Mapping, refer to the following Web site:

`http://publib.boulder.ibm.com/eserver/`

Part 3. Appendixes

Appendix A. Security Checklist

This appendix provides a checklist of security actions to perform on a newly installed or existing system. Although this list is not a complete security checklist, it can be used as a foundation to build a security checklist for your environment.

1. When installing a new system, install AIX from secure base media. Perform the following procedures at installation time:
   - Do not install desktop software, such as CDE, GNOME, or KDE, on servers.
   - Install required security fixes and any recommended maintenance level fixes. See the eServer pSeries Support Fixes website (http://techsupport.services.ibm.com/server/fixes?view=pSeries) for the newest service bulletins, security advisories, and fix information.
   - Back up the system after the initial installation and store the system backup in a secure location.
2. Establish access control lists for restricted files and directories.
3. Disable unneccessary user accounts and system accounts, such as daemon, bin, sys, adm, lp, uucp. Deleting accounts is not recommended because it deletes account information, such as user IDs and user names, which may still be associated with data on system backups. If a user is created with a previously deleted user ID and the system backup is restored on the system, the new user might have unexpected access to the restored system.
4. Review the **/etc/inetd.conf**, **/etc/inittab**, **/etc/rc.nfs**, and **/etc/rc.tcpip** files on a regular basis and remove all unnecessary daemons and services.
5. Verify that the permissions for the following files are set correctly:

```
-rw-rw-r-- root     system  /etc/filesystems
-rw-rw-r-- root     system  /etc/hosts
-rw------- root     system  /etc/inittab
-rw-r--r-- root     system  /etc/vfs
-rw-r--r-- root     system  /etc/security/failedlogin
-rw-rw---- root     audit   /etc/security/audit/hosts
```

6. Disable the root account from being able to remotely log in. The root account should be able to log in only from the system console.
7. Enable system auditing. For more information, see Chapter 3, “Auditing” on page 47.
8. Enable a login control policy. For more information, see “Login Control” on page 17.
9. Disable user permissions to run the **xhost** command. For more information, see “Managing X11 and CDE Concerns” on page 20.
10. Prevent unauthorized changes to the **PATH** environment variable. For more information, see “PATH Environment Variable” on page 28.
11. Disable telnet, rlogin, and rsh. For more information, see Chapter 9, “TCP/IP Security” on page 119.
12. Establish user account controls. For more information, see “User Account Control” on page 27.
13. Enforce a strict password policy. For more information, see “Passwords” on page 38.
14. Establish disk quotas for user accounts. For more information, see “Recovering from Over-Quota Conditions” on page 44.
15. Allow only administrative accounts to use the **su** command. Monitor the **su** command’s logs in the **/var/adm/sulog** file.
16. Enable screen locking when using X-Windows.
17. Restrict access to the **cron** and **at** commands to only the accounts that need access to them.
18. Alias the **ls** command to show hidden files and characters in a file name.
19. Alias the **rm** command to avoid accidentally deleting files from the system.
20. Disable unnecessary network services. For more information, see Chapter 10, “Network Services” on page 129.
21. Perform frequent system backups and verify the integrity of backups.

22. Subscribe to security-related e-mail distribution lists.

Appendix B. Security Resources

This appendix provides information on various security-related resources. Web site addresses can become invalid or outdated with little notice. For more information on the IBM policy regarding Web sites and non-IBM resources, refer to Appendix E, “Notices” on page 233.

Security Web Sites

AIX Virtual Private Networks: http://www-1.ibm.com/servers/aix/products/ibmsw/security/vpn/index.html

CERIAS (Center for Education and Research in Information Assurance and Security): http://www.cerias.purdue.edu/

CERT (Computer Emergency Response Team, at Carnegie Mellon University): http://www.cert.org

CIAC (Computer Incident Advisory Capability): http://ciac.llnl.gov

Computer Security Resource Clearinghouse: http://csrc.ncsl.nist.gov/

FIRST (Forum of Incident Response and Security Teams): http://www.first.org/

IBM eServer Security Planner: http://www-1.ibm.com/servers/security/planner/

IBM Security Solutions: http://www-3.ibm.com/security/index.shtml

OpenSSH: http://www.openssh.org/

Security Mailing Lists

CERT: http://www.cert.org/contact_cert/certmaillist.html

IBM Software Technical Mailings: http://techsupport.services.ibm.com/server/listserv

comp.security.unix: news:comp.security.unix

Security Online References

Common Criteria Concepts FAQ: http://www.radium.ncsc.mil/tpep/process/faq-sect3.html

Rainbow Series Library: http://www.radium.ncsc.mil/tpep/library/rainbow/

faqs: org: http://www.faqs.org/faqs/computer-security/

IBM eServer pSeries Information Center: http://publib16.boulder.ibm.com/pseries/en_US/infocenter/base

Appendix C. Summary of Common AIX System Services

The following table lists the more common system services within AIX. Use this table to recognize a starting point for securing your system.

Before you proceed with securing your system, back up all your original configuration files, especially:

- **/etc/inetd.conf**
- **/etc/inittab**
- **/etc/rc.nfs**
- **/etc/rc.tcpip**

Service	Daemon	Started by	Function	Comments
inetd/bootps	inetd	/etc/inetd.conf	bootp services to diskless clients	• Necessary for Network Installation Management (NIM) and remote booting of systems • Works concurrently with tftp • Disable in most cases
inetd/chargen	inetd	/etc/inetd.conf	character generator (testing only)	• Available as a TCP and UDP service • Provides opportunity for Denial of Service attacks • Disable unless you are testing your network
inetd/cmsd	inetd	/etc/inetd.conf	calendar service (as used by CDE)	• Runs as root, therefore a security concern • Disable unless you require this service with CDE • Disable on back room database servers
inetd/comsat	inetd	/etc/inetd.conf	Notifies incoming electronic mail	• Runs as root, therefore a security concern • Seldom required • Disable
inetd/daytime	inetd	/etc/inetd.conf	obsolete time service (testing only)	• Runs as root • Available as a TCP and UDP service • Provides opportunity for a Denial of Service PING attacks • Service is obsolete and used for testing only • Disable
inetd/discard	inetd	/etc/inetd.conf	/dev/null service (testing only)	• Available as TCP and UDP service • Used in Denial of Service Attacks • Service is obsolete and used for testing only • Disable

Service	Daemon	Started by	Function	Comments
inetd/dtspc	inetd	/etc/inetd.conf	CDE Subprocess Control	• This service is started automatically by the **inetd** daemon in response to a CDE client requesting a process to be started on the daemon's host. This makes it vulnerable to attacks • Disable on back room servers with no CDE • CDE might be able to function without this service • Disable unless absolutely needed
inetd/echo	inetd	etc/inetd.conf	echo service (testing only)	• Available as UDP and TCP service • Could be used in Denial of Service or Smurf attacks • Used to echo at someone else to get through a firewall or start a datastorm • Disable
inetd/exec	inetd	/etc/inetd.conf	remote execution service	• Runs as root and therefore is dangerous • Requires that you enter a user ID and password, which are passed unprotected • This service is highly susceptible to being snooped • Disable
inetd/finger	inetd	/etc/inetd.conf	finger peeking at users	• Runs as root and therefore is dangerous • Gives out information about your systems and users • Disable
inetd/ftp	inetd	/etc/inetd.conf	file transfer protocol	• Runs as root user • User id and password are transferred unprotected, thus allowing them to be snooped • Disable this service and use a public domain secure shell suite
inetd/imap2	inetd	/etc/inetd.conf	Internet Mail Access Protocol	• Ensure that you are using the latest version of this server • Only necessary if you are running a mail server. Otherwise, disable • User ID and password are passed unprotected
inetd/klogin	inetd	/etc/inetd.conf	Kerberos login	• Enabled if your site uses Kerberos authentication

Service	Daemon	Started by	Function	Comments
inetd/kshell	inetd	/etc/inetd.conf	Kerberos shell	• Enabled if your site uses Kerberos authentication
inetd/login	inetd	/etc/inetd.conf	rlogin service	• Susceptible to IP spoofing, DNS spoofing • Data, including User IDs and passwords, is passed unprotected • Runs as root and is therefore dangerous • Use a secure shell instead of this service
inetd/netstat	inetd	/etc/inetd.conf	reporting of current network status	• Could potentially give network information to hackers if run on your system • Disable
inetd/ntalk	inetd	/etc/inetd.conf	Allows users to talk with each other	• Runs as root and is therefore dangerous • Not required on production or back room servers • Disable unless absolutely needed
inetd/pcnfsd	inetd	/etc/inetd.conf	PC NFS file services	• Disable service if not currently in use • If you need a service similar to this, consider Samba, as the pcnfsd daemon predates Microsoft's release of SMB specifications
inetd/pop3	inetd	/etc/linetd.conf	Post Office Protocol	• User IDs and passwords are sent unprotected • Only needed if your system is a mail server and you have clients who are using applications that only support POP3 • If your clients use IMAP, use that instead, or use the POP3s service. This service has a Secure Socket Layer (SSL) tunnel • Disable if you are not running a mail server or have clients who need POP services
inetd/rexd	inetd	/etc/inetd.conf	remote execution	• Runs as root and therefore is dangerous • Peers with the **on** command • Disable service • Use **rsh** and **rshd** instead

Service	Daemon	Started by	Function	Comments
inetd/quotad	inetd	/etc/inetd.conf	reports of file quotas (for NFS clients)	• Only needed if you are running NFS file services • Disable this service unless required to provide an answer for the **quota** command • If you need to use this service, keep all patches and fixes for this service up to date
inetd/rstatd	inetd	/etc/inetd.conf	Kernel Statistics Server	• If you need to monitor systems, use SNMP and disable this service • Required for use of the **rup** command
inetd/rusersd	inetd	/etc/inetd.conf	info about user logged in	• This is not an essential service. Disable • Runs as root and therefore is dangerous • Gives out a list of current users on your system and peers with **rusers**
inetd/rwalld	inetd	/etc/inetd.conf	write to all users	• Runs as root user and therefore is dangerous • If your systems have interactive users, you might need to keep this service • If your systems are production or database servers, this is not needed • Disable
inetd/shell	inetd	/etc/inetd.conf	rsh service	• Disable this service if possible. Use Secure Shell instead • If you must use this service, use the TCP Wrapper to stop spoofing and limit exposures • Required for **xhier**
inetd/sprayd	inetd	/etc/inetd.conf	RPC spray tests	• Runs as root user and therefore is dangerous • Might be required for diagnosis of NFS network problems • Disable if you are not running NFS
inetd/systat	inetd	/etc/inted.conf	″ps -ef″ status report	• Allows for remote sites to see the process status on your system • This service is disabled by default. You must check periodically to ensure that the service has not been enabled

Service	Daemon	Started by	Function	Comments
inetd/talk	inetd	/etc/inetd.conf	establish split screen between 2 users on the net	• Not a required service • Used with the **talk** command • Provides UDP service at Port 517 • Disable unless you need multiple interactive chat sessions for UNIX user
inetd/ntalk	inetd	/etc/inetd.conf	″new talk″ establish split screen between 2 users on the net	• Not a required service • Used with the **talk** command • Provides UDP service at Port 517 • Disable unless you need multiple interactive chat sessions for UNIX user
inetd/telnet	inetd	/etc/inetd.conf	telnet service	• Supports remote login sessions, but the password and ID are passed unprotected • If possible, disable this service and use Secure Shell for remote access instead
inetd/tftp	inetd	/etc/inetd.conf	trivial file transfer	• Provides UDP service at port 69 • Runs as root user and might be compromised • Used by NIM • Disable unless you are using NIM or have to boot a diskless workstation
inetd/time	inetd	/etc/inetd.conf	obsolete time service	• Internal function of **inetd** that is used by **rdate** command. • Available as TCP and UDP service • Sometimes used to synchronize clocks at boot time • Service is outdated. Use **ntpdate** instead • Disable this only after you have tested your systems (boot/reboot) with this service disabled and have observed no problems
inetd/ttdbserver	inetd	/etc/inetd.conf	tool-talk database server (for CDE)	• The **rpc.ttdbserverd** runs as root user and might be compromised • Stated as a required service for CDE, but CDE is able to work without it • Should not be run on back room servers or any systems where security is a concern

Service	Daemon	Started by	Function	Comments
inetd/uucp	inetd	/etc/inetd.conf	UUCP network	• Disable unless you have an application that uses UUCP
inittab/dt	init	/etc/rc.dt script in the /etc/inittab	desktop login to CDE environment	• Starts the X11 server on the console • Supports the X11 Display Manager Control Protocol (xdcmp) so that other X11 stations can log into the same machine • Service should be used on personal workstations only. Avoid using it for back room systems
inittab/dt_nogb	init	/etc/inittab	desktop login to CDE environment (NO graphic boot)	• No graphical display until the system is up fully • Same concerns as **inittab/dt**
inittab/httpdlite	init	/etc/inittab	web server for the **docsearch** command	• Default web server for the docsearch engine • Disable unless your machine is a documentation server
inittab/i4ls	init	/etc/inittab	license manager servers	• Enable for development machines • Disable for production machines • Enable for back room database machines that have license requirements • Provides support for compilers, database software, or any other licensed products
inittab/imnss	init	/etc/inittab	search engine for ″docsearch″	• Part of the default web server for the docsearch engine • Disable unless your machine is a documentation server
inittab/imqss	init	/etc/inittab	search engine for ″docsearch″	• Part of the default web server for the docsearch engine • Disable unless your machine is a documentation server
inittab/lpd	init	/etc/inittab	BSD line printer interface	• Accepts print jobs from other systems • You can disable this service and still send jobs to the print server • Disable this after you confirm that printing is not affected

Service	Daemon	Started by	Function	Comments
inittab/nfs	init	/etc/inittab	Network File System/Net Information Services	• NFS and NIS services based which were built on UDP/RPC • Authentication is minimal • Back room database servers should have no need for this • Disable this for back room machines
inittab/piobe	init	/etc/inittab	printer IO Back End (for printing)	• Handles the scheduling, spooling and printing of jobs submitted by the **qdaemon** • Disable if you are not printing from your system because you are sending print job to a server
inittab/qdaemon	init	/etc/inittab	queue daemon (for printing	• Submits print jobs to the **piobe** daemon • If you are not printing from your system, then disable
inittab/uprintfd	init	/etc/inittab	kernel messages	• Generally not required • Disable
inittab/writesrv	init	/etc/inittab	writing notes to ttys	• Only used by interactive UNIX workstation users • Disable this service for servers, back room databases, and development machines • Enable this service for workstations
inittab/xdm	init	/etc/inittab	traditional X11 Display Management	• Do not run on back room production or database servers • Do not run on development systems unless X11 display management is needed • Acceptable to run on workstations if graphics are needed
rc.nfs/automountd		/etc/rc.nfs	automatic file systems	• If you use NFS, enable this for workstations • Do not use the automounter for development or back room servers
rc.nfs/biod		/etc/rc.nfs	Block IO Daemon (required for NFS server)	• Enabled for NFS server only • If not an NFS server, then disable this along with **nfsd** and **rpc.mountd**
rc.nfs/keyserv		/etc/rc.nfs	Secure RPC Key server	• Manages the keys required for secure RPC • Important for NIS+ • Disable this if you are *not* using NFS and NIS and NIS+

Service	Daemon	Started by	Function	Comments
rc.nfs/nfsd		/etc/rc.nfs	NFS Services (required for NFS Server)	• Authentication is weak • Can lend itself to stack frame crashing • Enable if on NFS file servers • If you disable this, then disable **biod**, **nfsd**, and **rpc.mountd** as well
rc.nfs/rpc.lockd		/etc/rc.nfs	NFS file locks	• Disable if you are not using NFS • Disable this if you are not using file locks across the network • **lockd** daemon is mentioned in the SANS Top Ten Security Threats
rc.nfs/rpc.mountd		/etc/rc.nfs	NFS file mounts (required for NFS Server)	• Authentication is weak • Can lend itself to stack frame crashing • Should be enabled only on NFS file servers • If you disable this, then disable **biod** and **nfsd** as well
rc.nfs/rpc.statd		/etc/rc.nfs	NFS file locks (to recover them)	• Implements file locks across NFS • Disable unless you are using NFS
rc.nfs/rpc.yppasswdd		/etc/rc.nfs	NIS password daemon (for NIS master)	• Used to manipulate the local password file • Only required when the machine in question is the NIS master; disable in all other cases
rc.nfs/ypupdated		/etc/rc.nfs	NIS Update daemon (for NIS slave)	• Receives NIS database maps pushed from the NIS Master • Only required when the machine in question is a NIS slave to a Master NIS Server
rc.tcpip/autoconf6		/etc/rc.tcpip	IPv6 interfaces	• Disable unless you are running IPV6
rc.tcpip/dhcpcd		/etc/rc.tcpip	Dynamic Host Configure Protocol (client)	• Back room servers should not rely on DHCP. Disable this service • If your host is not using DHCP, disable
rc.tcpip/dhcprd		/etc/rc.tcpip	Dynamic Host Configure Protocol (relay	• Grabs DHCP broadcasts and sends them to a server on another network • Duplicate of a service found on routers • Disable this if you are not using DHCP or rely on passing information between networks

Service	Daemon	Started by	Function	Comments
rc.tcpip/dhcpsd		/etc/rc.tcpip	Dynamic Host Configure Protocol (server	• Answers DHCP requests from clients at boot time; gives client information, such as IP name, number, netmask, router, and broadcast address • Disable this if you are not using DHCP • Disabled on production and back room servers along with hosts not using DHCP
rc.tcpip/dpid2		/etc/rc.tcpip	outdated SNMP service	• Disable unless you need SNMP
rc.tcpip/gated		/etc.rc.tcpip	gated routing between interfaces	• Emulates router function • Disable this service and use RIP or a router instead
rc.tcpip/inetd		/etc/rc.tcpip	inetd services	• A thoroughly secured system should have this disabled, but is often not practical • Disabling this will disable remote shell services which are required for some mail and web servers
rc.tcpip/mrouted		/etc/rc.tcpip	multi-cast routing	• Emulates router function of sending multi-cast packets between network segments • Disable this service. Use a router instead
rc.tcpip/names		/etc/rc.tcpip	DNS name server	• Use this only if your machine is a DNS name server • Disable for workstation, development and production machines
rc.tcpip/ndp-host		/etc/rc.tcpip	IPv6 host	• Disable unless you use IPV6
rc.tcpip/ndp-router		/etc/rc.tcpip	IPv6 routing	• Disable this unless you use IPV6. Consider using a router instead of IPv6
rc.tcpip/portmap		/etc/rc.tcpip	RPC services	• Required service • RPC servers register with **portmap** daemon. Clients who need to locate RPC services ask the **portmap** daemon to tell them where a particular service is located • Disable only if you have managed to reduce RPC service so that the only one remaining is **portmap**
rc.tcpip/routed		/etc/rc.tcpip	RIP routing between interfaces	• Emulates router function • Disable if you have a router for packets between networks

Service	Daemon	Started by	Function	Comments
rc.tcpip/rwhod		/etc/rc.tcpip	Remote ″who″ daemon	• Collects and broadcasts data to peer servers on the same network • Disable this service
rc.tcpip/sendmail		/etc/rc.tcpip	mail services	• Runs as root user and therefore is dangerous • Has a long history of security breaches • Disable this service unless the machine is used as a mail server • If disabled, then do one of the following: – Place an entry in crontab to clear the queue. Use the **/usr/lib/sendmail -q** command – Configure DNS services so that the mail for your server is delivered to some other system
rc.tcpip/snmpd		/etc/rc.tcpip	Simple Network Management Protocol	• Disable if you are not monitoring the system via SNMP tools • SNMP may be required on critical servers, but not likely on workstations
rc.tcpip/syslogd		/etc/rc.tcpip	system log of events	• Never disable this service • Prone to denial of service attacks • Required in any system
rc.tcpip/timed		/etc/rc.tcpip	Old Time Daemon	• Disable this service and use xntp instead
rc.tcpip/xntpd		/etc/rc.tcpip	New Time Daemon	• Keeps clocks on systems in sync • Disable this service. • Configure other systems as time servers and let other systems synchronize to them with a cron job that calls ntpdate
dt login		/usr/dt/config/Xaccess	unrestricted CDE	• If you are not providing CDE login to a group of X11 stations, you can restrict dtlogin to the console.

Service	Daemon	Started by	Function	Comments
anonymous FTP service		user rmuser -p <username>	anonymous ftp	• Anonymous FTP ability prevents you from tracing FTP usage to a specific user • Remove user ftp if that user account exists, as follows: **rmuser -p ftp** • Further security can be obtained by populating the **/etc/ftpusers** file with a list of those who should not be able to ftp to your system
anonymous FTP writes			anonymous ftp uploads	• No file should belong to ftp. • FTP anonymous uploads allow the potential for misbehaving code to be placed on your system. • Put the names of those users you want to disallow into the **/etc/ftpusers** file • Some examples of system-created users you might want to disallow from anonymously uploading via FTP to your system are: root, daemon, bin.sys, admin.uucp, guest, nobody, lpd, nuucp, ladp, imnadm • Change the owner and group rights to the **ftpusers** files as follows: **chown root:system /etc/ftpusers** • Change the permissions to the **ftpusers** files to a stricter setting as follows: **chmod 644 /etc/ftpusers**
ftp.restrict			ftp to system accounts	• No user from the outside should be allowed to replace root files via **ftpusers** file
root.access		/etc/security/user	rlogin/telnet to root account	• Set the rlogin option in the **etc/security/user** file to false • Anyone logging in as root should first log in under their own name and then **su** to root; this provides an audit trail
snmpd.readWrite		/etc/snmpd.conf	SNMP readWrite communities	• If you are *not* using SNMP, disable the SNMP daemon. • Disable community private and community system in the **/etc/snmpd.conf** file • Restrict 'public' community to those IP addresses that are monitoring your system

Service	Daemon	Started by	Function	Comments
syslog.conf			configure syslogd	• If you have not configured **/etc/syslog.conf**, then disable this daemon • If you are using **syslog.conf** to log system messages, then keep enabled

Appendix D. Summary of Network Service Options

To achieve a higher level of system security, there are several network options that you can change using `0` to disable and `1` to enable. The following list identifies these parameters you can use with the **no** command.

Parameter	Command	Purpose
bcastping	/usr/sbin/no -o bcastping=0	Allows response to ICMP echo packets to the broadcast address. Disabling this prevents Smurf attacks.
clean_partial_conns	/usr/sbin/no -o clean_partial_conns=1	Specifies whether or not SYN (synchronizes the sequence number) attacks are being avoided.
directed_broadcast	/usr/sbin/no -o directed_broadcast=0	Specifies whether to allow a directed broadcast to a gateway. Setting to 0 helps prevent directed packets from reaching a remote network.
icmpaddressmask	/usr/sbin/no -o icmpaddressmask=0	Specifies whether the system responds to an ICMP address mask request. Disabling this prevents access through source routing attacks.
ipforwarding	/usr/sbin/no -o ipforwarding=0	Specifies whether the kernel should forward packets. Disabling this prevents redirected packets from reaching remote network.
ipignoreredirects	/usr/sbin/no -o ipignoreredirects=1	Specifies whether to process redirects that are received.
ipsendredirects	/usr/sbin/no -o ipsendredirects=0	Specifies whether the kernel should send redirect signals. Disabling this prevents redirected packets from reaching remote network.
ip6srcrouteforward	/usr/sbin/no -o ip6srcrouteforward=0	Specifies whether the system forwards source-routed IPv6 packets. Disabling this prevents access through source routing attacks.
ipsrcrouteforward	/usr/sbin/no -o ipsrcrouteforward=0	Specifies whether the system forwards source-routed packets. Disabling this prevents access through source routing attacks.
ipsrcrouterecv	/usr/sbin/no -o ipsrcrouterecv=0	Specifies whether the system accepts source-routed packets. Disabling this prevents access through source routing attacks
ipsrcroutesend	/usr/sbin/no -o ipsrcroutesend=0	Specifies whether applications can send source-routed packets. Disabling this prevents access through source routing attacks.

Parameter	Command	Purpose
nonlocsroute	/usr/sbin/no -o nonlocsrcroute=0	Tells the Internet Protocol that strictly source-routed packets may be addressed to hosts outside the local network. Disabling this prevents access through source routing attacks.
tcp_pmtu_discover	/usr/sbin/no -o tcp_pmtu_discover=0	Disabling this prevents access through source routing attacks.
udp_pmtu_discover	/usr/sbin/no -o udp_pmtu_discover=0	Enables or disables path MTU discovery for TCP applications. Disabling this prevents access through source routing attacks.

For more information about network-tunable options, see *AIX 5L Version 5.2 Performance Management Guide*.

Appendix E. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Dept. LRAS/Bldg. 003
11400 Burnet Road
Austin, TX 78758-3498
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

AIX
AIX 5L
DB2
SecureWay
IBM
RS/6000

Lotus Notes is a registered trademark of the Lotus Development Corporation and/or IBM Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft is a trademark of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Index

Special characters

A

B

C

D

E

F

G

I

K

L

M

N

U

V

Readers' Comments — We'd Like to Hear from You

AIX 5L Version 5.2
Security Guide

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name _____ Address _____

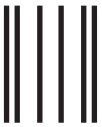
Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Information Development
Department H6DS-905-6C006
11501 Burnet Road
Austin, TX
78758-3493

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line