HEWLETT PACKARD

HP 9000 Computers

HP-UX System Security

HP-UX System Security

HP Part No. B2355-90045
Printed in USA August 1992

E0892

Legal Notices

The information in this document is subject to change without notice.

Hewlett-Packard makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Warranty. A copy of the specific warranty terms applicable to your Hewlett-Packard product and replacement parts can be obtained from your local Sales and Service Office.

HEWLETT-PACKARD COMPANY
3000 Hanover Street
Palo Alto, California 94304 U.S.A.

Printing History

The manual printing date and part number indicate its current edition. The printing date will change when a new edition is printed. Minor changes may be made at reprint without changing the printing date. The manual part number will change when extensive changes are made.

Manual updates may be issued between editions to correct errors or document product changes. To ensure that you receive the updated or new editions, you should subscribe to the appropriate product support service. See your HP sales representative for details.

First Edition—October 1989

Second Printing—January 1991

Second Edition—August 1992

Contents

1. Using this Book

Organization of Chapters 1-3
Using the System Administration Manager (SAM) 1-4
Invoking SAM 1-4
Definition of Terms 1-5

2. Setting up and Maintaining your Secure System

Setting Up your Secure System 2-1
Plan Prior to Conversion 2-1
Install the System from Tape 2-3
Convert to a Secure (Trusted) System 2-4
Prerequisites 2-4
Procedure 2-6
Enable Auditing 2-7
Maintaining your Secure System 2-11
Creating Product Description Files (pdf) 2-11
Example of a pdf entry 2-11
Verifying File System Consistency 2-12
Using pdf for Customized Filesets 2-12
Other Security Checks 2-13

3. Auditing Tasks

Set Audit Monitor and Audit Log Parameters 3-2
Prerequisite 3-2
Procedure 3-2
Turn Auditing On/Off 3-4
Prerequisite 3-4
Procedure 3-4
Select Users to be Audited 3-5
Prerequisite 3-5

Procedure 3-5
Select Events to be Audited 3-7
Prerequisite 3-7
Procedures 3-7
Select System Calls to be Audited 3-9
Prerequisite 3-9
Procedure 3-9
View Audit Logs 3-11
Procedure 3-11
Examples 3-12
Interpreting the Audit Log Data 3-13

4. Controlling User Access to Directories and Files
Assigning File Permissions to Specific Users and Groups. 4-1
Listing File Permissions with the lsacl Command 4-2
An Example Using the lsacl Command 4-3
Changing permissions with the chacl command 4-3
How the chmod and chacl Commands Interact 4-4
Using the chacl Command 4-4
Examples Using the chacl Command 4-5
Protecting Directories from File Tampering 4-7
Default Permissions 4-8

5. Protecting your Operating System and Files
Looking for setuid and setgid Programs 5-2
Verifying Password File Security 5-4
Protecting User Accounts 5-5
Safeguarding Network Security 5-7
Identifying Administrative Domain 5-7
Verifying Permission Settings on Network Control Files . . . 5-10
Controlling File System Export 5-10
Maintaining Device File Security 5-11
What to Do if You Discover a Security Breach 5-12

6. Planning for Security

Security Policy 6-2
Physical Security Policy 6-2
Procedural Security Policy 6-3
System Security Policy 6-3
Educating Users 6-4
Trusted Computing Base (TCB) 6-5
Discretionary Access Control (DAC) 6-5
Subjects 6-5
Objects 6-6
Least Privilege 6-6
Accountability 6-6
Typical Software Threats 6-7
Key Security Personnel 6-9
System Security Officer Tasks 6-9
System Administrator Tasks 6-9
Operator Tasks 6-10
Systems Programmer Tasks 6-10

7. Understanding Auditing

Event Types Which are Selectable for Auditing 7-2
Streamlining Audit Log Data 7-5
Guidelines for Writing Programs to Streamline Auditing Data . 7-6
Example of Self-Auditing Program 7-7
The Audit Record 7-9
Audit Log Files 7-10
Managing Audit Log Resources 7-16
Guidelines for Administering Your Auditing System 7-17
Performance Considerations 7-17
Using Auditing in a Diskless Environment 7-18

8. Planning User Access to System and Files

Password Security 8-1
Password Responsibilities 8-2
Criteria of a Good Password 8-3
Password Encryption 8-3
Two Password Files 8-4
/etc/passwd 8-4

Changing /etc/passwd Fields 8-4
/.secure/etc/passwd 8-5
Eliminating Pseudo-Accounts 8-6
Manipulating the Password Files 8-6
File Permissions 8-7
Root 8-7
Root Use Guidelines 8-7
System Files and Directories 8-8
Protecting Key Subsystems 8-9
Criteria for Modes 8-10
Security Considerations for Device Files 8-11
Protecting Disk Partitions 8-12
System Access by Modem 8-13
Managing User Accounts 8-14
Guidelines for Adding a User Account 8-14
Password Aging on User Accounts 8-16
Guidelines for Deactivating an Account 8-17
Guidelines for Reactivating a User Account 8-17
Guidelines for Removing an Account 8-18
Guidelines for Moving a User Account 8-19
Guidelines for Adding a Group 8-20
Guidelines for Removing a Group 8-21
Guidelines for Modifying a Group 8-21
Controlling File Access Selectively 8-22
Access Control Entries Defined 8-22
Comparing ACLs and File Permissions 8-23
Base ACL Entries = File Mode Permissions 8-24
Granting Selective Access with Optional ACLs 8-24
Access Check Algorithm 8-25
ACL Uniqueness 8-26
How to Use ACL Notation 8-26
Operator Form of ACLs (input only) 8-27
Using chacl to Set ACL Entries (operator form) 8-29
Short Form of ACLs (input and output) 8-30
Examples of ACL Entries in Short Form 8-31
Long Form of ACLs (output only) 8-32
Example of ACL Long Form 8-32
ACL Patterns 8-33

Examples of ACL Patterns 8-34
Working with ACL Functionality 8-35
New ACL Commands and Programs 8-35
Commands 8-35
System Calls 8-35
Library Routines 8-35
Existing HP-UX Core Programs and ACLs 8-36
General-Purpose Commands and System Calls 8-36
File Archive Commands 8-37
Configuration Control Commands 8-37
File System Maintenance Commands 8-37
ACLs in a Network Environment 8-38

9. Recognizing Risks to Your HP-UX Operating System
Overall Risk Management Guidelines 9-1
Set User ID (setuid) and Set Group ID (setgid) Programs . . . 9-2
How IDs are Set 9-2
Why setuid Programs Can Be Risky 9-3
Kinds of Attacks 9-3
Guidelines for Limiting setuid Power 9-4
Guidelines for Writing setuid Programs 9-5
Secure System Initialization 9-6
Exempted System Programs 9-7
Backup and Recovery Guidelines in a Secure Environment . . . 9-8
Backup Security Practices 9-8
Recovery Security Practices 9-10
Guidelines for Mounting and Unmounting a File System 9-12
Guidelines for Shutting Down a System Securely 9-14
Security Precautions 9-14
Guidelines for Handling Network Security Breaches 9-15

10. Trusted Networking
Controlling Administrative Domain 10-2
Understanding Network Services 10-3
Protecting Passwords when using RFA 10-4
Using inetd.sec to Restrict Outside Access 10-4
Denying Access with /etc/ftpusers 10-5
Files Mounted in an NFS Environment 10-5

Server Vulnerability 10-6
Client Vulnerability 10-6
How to Safeguard NFS-mounted Files 10-7
Link-Level Access 10-7

Index

Figures

7-1. Self-Auditing Process 7-5
7-2. The Primary Audit Log Approaches AFS 7-11
7-3. Primary Audit Log Reaches AFS 7-12
7-4. Primary Audit Log Approaches FSS 7-13
7-5. Primary Audit Log Reaches FSS 7-14
7-6. Primary and Auxiliary Audit Log Files 7-15
10-1. Examples of Administrative Domain 10-2

Tables

1-1. Typographical Conventions 1-2
2-1. Audit Monitor and Log Parameters 2-8
7-1. Event Types 7-3

1

Using this Book

HP-UX System Security is a supplementary reference book of procedures and guidelines essential to administering the HP-UX series 300 or 800 computer as a **trusted system**. According to the industry-standard *Department of Defense Trusted Computer System Evaluation Criteria*, a trusted system is one that "employs sufficient hardware and software integrity measures to allow its use for processing simultaneously a range of sensitive or classified information."

You should read *HP-UX System Security* if you are the person responsible for the security of an HP-UX system. For overall system admininistration, see the *System Administration Tasks* manual specific to your system.

Turn the page for conventions and terms used in this book, a "roadmap" through *HP-UX System Security* and a description of the SAM user interface you will use to access the auditing features.

Table 1-1. Typographical Conventions

If You See This	It Means
boldface text	**Boldface text** is used when introducing a term.
`computer text`	`Computer text` is used in several ways: ■ To represent literally what is displayed by the computer. For example: `logout`. ■ To show exact user input. For example: `setenv` ■ To denote command or file names in text. For example: `/etc/config`. ■ To denote text which appears in object lists, as seen in the System Administration Manager (SAM). For example: `Auditing and Security ->`.
italic text	*Italic text* represents a user-supplied variable. For example: in **cp** *filename*, **cp** is typed in exactly, and *filename* represents the name of the file you choose. *Italic text* is also used to show emphasis and annotation.
`function key`	A shaded key, such as `Select Item`, represents a key label seen on the computer screen. Pressing the corresponding keycap, (i.e. f1) activates the key's label. This typography is also used when referring to menu items, such as `Add...` as seen in the System Administration Manager (SAM).

Organization of Chapters

The first 4 chapters of *HP-UX System Security* provide instructions on performing specific security-related tasks. Later chapters explain the concepts underlying security concerns.

Tasks:

Chapter 2	Setting Up and Maintaining Your Secure System. Explains the procedure for implementing security features for the first time, and then maintaining them.
Chapter 3	Auditing Tasks. Provides detailed instructions to modify auditing parameters.
Chapter 4	Controlling User Access to Directories and Files. Guides the system administrator on manipulating file permissions using access control lists (ACLs).
Chapter 5	Protecting Your Operating System and Files. Helps locate possible security breaches.

Concepts:

Chapter 6	Planning for Security. Addresses overall security policy and defines security terms.
Chapter 7	Understanding Auditing. Examines the concepts underlying auditing functionality.
Chapter 8	Planning User Access to System and Files. Describes password security, file permissions, and access control lists.
Chapter 9	Recognizing Risks to Your HP-UX Operating System. Discusses `setuid` programs, environmental safeguards, and best practices for backup, recovery, file system mounting and unmounting, and system shutdown.
Chapter 10	Trusted Networking. Provides guidelines for improving security in a networked environment.

Using the System Administration Manager (SAM)

Use the **System Administration Manager** (SAM) whenever possible to perform security-related system administration tasks. SAM, a window environment reserved for users with superuser capabilities, queries you through each step, focuses choices, and protects you from corrupting critical files. It avoids introducing mistakes or compromises that might breach security.

The following security-related system administration tasks can be performed with SAM:

- Turning auditing on/off
- Setting the audit monitor and log parameters
- Viewing audit logs
- Viewing and modifying audit options for users, events, and system calls
- Converting to a trusted system
- Managing user accounts

Invoking SAM

- Invoke SAM by typing:

  ```
  /usr/bin/sam
  ```

- Select the area you wish to work in:
  - In the X Window System interface, double-click on the area title.
  - In the text terminal interface), move the highlight through the SAM list items using the arrow (▲, ▼) keys and press Return.

For details on the differences between the X Window System interface and the text terminal interface, consult Chapter 1, "Introduction to System Administration" in the *System Administration Tasks* manual. This introduction will give you help in interpreting the meanings of the terms **highlight** and **choose**, which indicate different actions within the two interfaces.

See specific tasks in this manual for more detailed information on SAM usage.

Definition of Terms

administrative domain	A group of systems connected by network services that allow users to access one another without password verification. See Chapter 10, for more information.
audomon	The auditing system monitor daemon. See Chapter 7 for more information.
Audit ID	A number, assigned by the system and unique to each user, used to identify users for auditing purposes. See Chapter 7 for more information.
Auxiliary log file	A back-up file used for collecting auditing data. See Chapter 7 for more information.
Audit File Switch point (AFS)	The point of saturation of the audit log file when auditing attempts to switch the recording of log data to a back-up file. See Chapter 7 for more information.
Discretionary access control (DAC)	A means of restricting access to objects based on file permission modes, which can be set by the owner.
Event type	System activities with like behavior that have been grouped together for use by auditing. See Chapter 7 for more information.
File System Switch point (FSS)	The point of saturation of the file system where the audit log file resides when auditing attempts to switch the recording of log data to a back-up file. See Chapter 7 for more information.
least privilege	Users receive the lowest level of privilege necessary to perform their tasks. See Chapter 8 for more information.
min-free	A system defined saturation point for file systems. See Chapter 7 for more information.

password	A private character string used to authenticate an identity.
Primary log file	The current file used by auditing for collecting data. See Chapter 7 for more information.
SAM	The System Administration Manager. A menu-driven package that simplifies some of the most commonly performed system administration tasks.
Self-auditing programs	Programs that can suspend low-level auditing of certain processes. See Chapter 7 for more information.
setgid program	A program whose group ID is set to grant a user privileges equivalent to that of the program group.
setuid program	A program whose user ID is set to grant a user privileges equivalent to that of the program owner.
Trap door	A hidden software or hardware mechanism that circumvents system security. See Chapter 6 for more information.
Trojan horse	A computer program containing additional functionality that exploits the program's capabilities for destructive ends. See Chapter 6 for more information.
Trusted computing base (TCB)	All protection mechanisms within a computer system (including hardware, firmware, and software) responsible for enforcing security policy. Security effectiveness is based on both the TCB mechanisms and its correct implementation by system administrative personnel.
trusted system	A system that employs sufficient hardware and software security measures to allow its use for processing sensitive material.

Virus	Code segments that replicate themselves through a system destructively. See Chapter 6 for more information.
Worm	A program that migrates through a system for harmful purposes. See Chapter 6 for more information.

2

Setting up and Maintaining your Secure System

This chapter provides a strategic roadmap for:

- Setting up a secure system
- Enabling Auditing
- Maintaining your system after having implemented security features

Setting Up your Secure System

The procedures presented here cover all of the tasks required to implement a secure (trusted) system. Follow the steps in "Plan Prior to Conversion", "Install the System from Tape", "Convert to a Secure (Trusted) System", and "Enable Auditing" to set up your secure system.

Plan Prior to Conversion

1. Assess your auditing needs.
   a. Read Chapter 7 for an overview of the auditing subsystem.
   b. Determine the size of your audit log files.
   c. Decide which events and users to audit.
   d. Determine how often to evaluate your audit log.

2. Establish an overall security policy appropriate to your worksite:

   a. Read Chapter 6 to familiarize yourself with the issues.
   b. Identify the security requirements of your own worksite.
   c. Establish written guidelines at both administrative and user levels. Successful guidelines reflect the realistic needs of the worksite.
   d. Inform all personnel—administrators as well as individual users—of their security responsibilities.
   e. Keep security guidelines updated and all users informed as requirements change.

3. Inspect all existing files on your system for security risks, and remedy them. This is mandatory the first time you install a secure (trusted) system. Thereafter, examine your files regularly, or when you suspect a security breach.

   Perform the commands found in Chapter 5. If you find an unsafe situation, correct it by:

   a. Removing the file
   b. Changing the file permissions
   c. Moving the file

   Read Chapter 9 for an overview.

Once you are confident that no security breaches exist you may proceed to the next section.

Install the System from Tape

Caution You may *not* update the system, you must install. The effectiveness of security features may be compromised if your system files are altered.

1. Back up your file system for later recovery of user files. Be sure to follow the procedures in the preceeding section on inspecting your file system for security risks before backing up.

   You can use any of the backup and recovery programs provided by HP-UX for your *initial* backup and recovery. Once security features are implemented, however, use only *fbackup*(1M) and *frecover*(1M). Read "Backup and Recovery Guidelines in a Secure Environment" in Chapter 9 for details.

2. Install the HP-UX operating system from tape. Choose *all* filesets when installing.

   Refer to *Installing and Updating HP-UX*.

Note The first time the system is booted, after install, you will receive the following message:

```
/etc/auditrc: This file must be edited to activate audit subsystem
```

This step is not necessary when using SAM to activate the auditing subsystem

3. Recover user files from your backup media.
4. Create **Product Description Files (pdfs)** for each product fileset installed on your system. Use these as a reference when checking for security breaches later. See "Maintaining your Secure System" later in this chapter for instructions.
5. Proceed directly to the conversion task that follows. No other operations should occur before conversion.

Convert to a Secure (Trusted) System

When you convert to a secure (trusted) system the conversion program:

- Creates a new, secure password file called **/.secure/etc/passwd**
- Copies the **/etc/passwd** file into **/etc/passwd.old.sav**, a file with restrictive permissions
- Moves encrypted passwords from the **/etc/passwd** file to the **/.secure/etc/passwd** file and replaces the password field in **/etc/passwd** with a *
- Forces all users to use passwords
- Creates an audit ID number for each user
- Sets the audit flag on for all existing users
- Converts the **at**, **batch** and **crontab** files to use the submitter's audit ID

Prerequisites

Before running this conversion program:

1. Be sure that the directory **/etc/filesets** contains the **AUDIT** fileset. To check for this fileset, enter:

   ```
   ls /etc/filesets | grep AUDIT
   ```

   If the system returns the message

   ```
   AUDIT
   ```

   the fileset is there and you can proceed. If nothing is returned it probably means that during installation you did not choose all filesets. To add the **AUDIT** fileset, see the update instructions in the *System Administration Tasks* manual.

2. Verify that the **/etc/newconfig/auditrc** file has been copied to **/etc/auditrc** by typing:

   ```
   ls /etc/auditrc
   ```

   If you receive the message

   ```
   /etc/auditrc not found
   ```

   use the following command string to copy the file:

   ```
   cp /etc/newconfig/auditrc /etc/auditrc
   ```

3. Verify that the **/etc/rc** file calls the **/etc/auditrc** file by typing:

   ```
   more /etc/rc
   ```

   Locate the text **Here is the heart of the rc script:** in the **/etc/rc** file and verify that the following lines appear before it:

   ```
   audit_start()
   {
           # Start up the auditing subsystem
           if [ -x /etc/auditrc ]  & /etc/auditrc
           then
                   echo "Audit subsystem started"
           fi
   }
   ```

   Insert these lines if they are not there and insert the subroutine call **audit_start** at the end of the list of calls in the **standalone** section and in the **localroot** section of this file.

2

Procedure

To convert to a secure (trusted) system:

1. Invoke SAM by typing:

   ```
   /usr/bin/sam
   ```

2. Highlight **Auditing and Security** and activate (Open).

3. On the "Auditing and Security" window, highlight **Users** and activate (Open).

Note If your installation is already converted to a trusted system, the "Users" window will appear. It contains a list of users and their audit status. You may proceed with normal auditing tasks.

If your installation is *not* converted to a trusted system, a "Confirmation" window will appear. It contains a message describing the actions taken in the conversion process.

---

4. Press (Y) to begin the conversion process. As the system is converted, you will receive messages concering the progress of the conversion.

You are now ready to enable your auditing subsystem.

Enable Auditing

The system supplies default auditing parameters at installation. Some of these defaults are activated automatically, some have to be enabled. Before turning auditing on go through the steps below.

From the "Auditing and Security" window you may choose to change the auditing status of users, events, or system calls. To make changes, enter the appropriate functional area by highlighting either **Users**, **Events**, or **System Calls**.

1. By default, the audit status for all users is set to **y**. To retain this state simply exit the screen. However, if you wish to modify this screen follow the procedures in Chapter 3 in the section titled "Select Users to be Audited."

2. The event types **admin**, **logon**, and **moddac** are selected as defaults by the system. Both **Audit Success** and **Audit Failure** are set to **y**. This is the minimum event type selection recommended for running a secure system. To activate these default event types, enter the "Events" functional area, highlight these events, and choose **Audit both success and failure** from the "Actions" menu.

   See Chapter 7 the section titled "Event Types which are Selectable for Auditing" for more information about event types and their associated processes. See Chapter 3 the section titled "Select Events to be Audited" for detailed procedures about changing your event type selection.

3. The system calls associated with the event types selected are automatically enabled. If you wish to alter system call selection, see Chapter 3 the section titled "Select System Calls to be Audited."

4. To enter the "Set Audit Monitor and Log Parameters" window, choose `Log/Monitor Params` from the "Actions" menu in any of the "Auditing and Security" functional areas ("Users," "Events," or "System Calls"). The following table describes the parameters set here and shows the default values supplied automatically at installation.

Table 2-1. Audit Monitor and Log Parameters

SAM Screen Entry	Default	What is Being Set Here
Primary log file path name	`/.secure/etc/auditlog_1`	The full path name of the file set to collect auditing data initially.
Primary log file switch size (AFS)	5,000 kbytes	The Audit File Switch size for the primary audit log file.
Auxiliary log file path name	`/.secure/etc/auditlog_2`	The full path name of the backup file for collecting auditing data.
Auxiliary log file switch size (AFS)	1,000 kbytes	The Audit File Switch size for the backup audit log file.
Monitor wake up interval	1 minute	How frequently you want audomon to check the state of the auditing system when it is approaching either AFS or FSS.
Allowable free space minimum (FSS)	20%	The FSS value is the File System Switch point, the minimum amount of file space allowed on the file system before a switch to a backup file is attempted.
Percentage of log size limit to trigger warnings	90%	When either the AFS or FSS point is nearing this percentage, **audomon** will send out its initial warning message.

Chapter 7 discusses the audit file interaction in more detail.

5. Assess the size of your file systems using the **bdf** command. Use the **Shell** escape from SAM by pressing the **Shell** key and execute the **bdf** command at the shell prompt by typing:

   ```
   bdf
   ```

   This provides you with the following information about your file systems:

   - Size in kbytes
   - Amount of space used in kbytes
   - Amount of space available in kbytes
   - Capacity by percentage
   - Mounted on

   Here is an example of possible output of the **bdf** command:

   ```
   Filesystem          kbytes    used   avail capacity Mounted on
   /dev/dsk/c2000d0s0   23191   19388    1483    93%   /
   /dev/dsk/c2000d0s5  207267  184224    2316    99%   /mnt
   /dev/dsk/c3d0s2     120942   13374   95473    12%   /mnt/oth
   /dev/dsk/c2d0s10    121771   48273   61320    44%   /usr/tool
   ```

6. Choose a file system with adequate space for your audit log files. For example, using the system supplied default for the primary audit log file would mean that:

   a. The **/.secure/etc** filesystem must have more than 5000 kbytes available for the primary audit log file, and
   b. It must have more than 20% of its file space available.

   The following errors can occur if file system space is inadequate:

   a. If the primary audit log file resides in a file system with less than 20% file space available, the system immediately switches to the auxiliary audit log file when auditing is invoked.

b. If the file system chosen has insufficient space to handle the indicated audit file switch size (i.e., 5000 kbytes), the system issues the following warning:

```
May not have completed task . . .

current audit file /.secure/etc/auditlog_1
insufficient space available on audit file filesystem,
specify a different audit file or select a smaller AFS
auditing system unchanged
```

7. Provide a new path name for the auxiliary audit log file. We recommend that the primary and auxiliary audit log files reside on separate file systems. Since each installation of HP-UX is different it is not known which file systems are available at your installation. Hence, the default situation has both the primary and auxiliary log files residing on the same file system, `/.secure/etc`.

8. You can now enable these parameters and turn auditing on. Leave the default answer of `y` at the question `Turn auditing ON?` and leave the default of (`y`) at the query `Make changes permanent (live beyond reboot)?` if you wish these parameters to become the new default parameters. Press the Perform Task key.

You are now ready for normal operation as a secure system.

Maintaining your Secure System

Once your secure system is up and running you should verify file system security and check for security breaches periodically. The follow sections describe a set of programs to use as a tool for running such security checks.

Creating Product Description Files (pdf)

You will want to create **Product Description Files** (`pdfs`) for each of the product filesets installed on your system to use as a basis for later comparison.

The `pdf` files you create will contain a single-line entry for each file having the following information:

field	comments
pathname	Absolute
owner	Either symbolic or numeric ID
group	Either symbolic or numeric ID
mode	Symbolic representation as displayed by the `ls -l` command
size	Size of the file in bytes. Major and minor numbers are listed for device special files.
links	Number of hard links to pathname.
version	Numeric value, reported by *what*(1).
checksum	File contents computed by a checksum algorithm. This field reflects the slightest change to a file, even a single character.
linked_to	Indicates whether the file has symbolic or hard links.

Fields are separated by colons (:).
Comment lines begin with percent signs (%).

Example of a pdf entry

Here is a sample pdf entry:

```
% Product Description File
% UX_CORE fileset, Release 7.0
/bin/ls:bin:bin:-r-xr-xr-x:94208:6:64.1:3245:
/bin/ll:bin:bin:-r-xr-xr-x:94208:6:64.1:3245:/bin/ls
% total size is 154712 bytes.
% total size is 153 blocks.
```

Producing pdf files is a simple task involving the use of the *mkpdf*(1M) command. *Mkpdf*(1M) finds each file, gathers statistics on it, and tabulates those statistics in the format shown above.

The following example shows how to produce pdfs for all filesets currently installed using the *sh*(1) or *ksh*(1) login shell. You must be superuser to execute these commands.

```
release='uname -a'
date='date'
ls /etc/filesets |
while read fileset
do
        comment="Fileset $fileset, Release $release, $date"
        echo processing $comment
        mkpdf -c "$comment" /etc/filesets/$fileset /system/$fileset/pdf
done
```

The resulting pdfs will reside in files named **/system/$fileset/pdf**, for example **/system/UX_CORE/pdf**.

Verifying File System Consistency

To verify that system files have not been altered, use the *pdfck*(1M) command to compare current file statistics against the HP-UX release file system statistics you collected and stored in **/system/$fileset/pdf**.

1. Run the *pdfck*(1M) command. **Pdfck** does not produce output unless it finds discrepancies.
2. Examine the results, paying particular attention to changes in:
   - Mode permission bits
   - Owner ID and group ID
   - Checksum discrepancies

Using pdf for Customized Filesets

Use the same procedures as above to verify file consistency for your customized system.

1. Create a prototype file list and run the *mkpdf*(1M) command on that list to produce a pdf.
2. Archive that pdf somewhere safe.

3. To verify the consistency of the listed files, run the **pdfck** command using the archived **pdf** file as baseline. **Pdfck** will read each entry in the file, gather the current statistics, compare it to baseline, and report any discrepancies.

Other Security Checks

In addition to the fileset verifications, you might want to run regularly the checks provided in Chapter 5.

3

Auditing Tasks

This chapter provides:

- Information regarding the default auditing system parameters supplied at installation, and
- Procedures for:
  - Setting audit monitor and audit log parameters
  - Turning auditing on/off
  - Selecting users to be audited
  - Selecting events to be audited
  - Selecting system calls to be audited
  - Viewing audit log files

Using SAM to perform all auditing tasks is recommended as it focuses choices and helps to avoid mistakes that might cause a breach in security. However, all auditing tasks can be done manually using the following commands:

audsys(1M)	Starts or halts the auditing system and sets or displays audit file information
audusr(1M)	Selects users to be audited
audevent(1M)	Changes or displays event or system call status
audomon(1M)	Sets the audit file monitoring and size parameters
audisp(1)	Displays the audit record

See *HP-UX Reference* for more details on these commands.

Set Audit Monitor and Audit Log Parameters

The following default values are supplied by the system automatically and will be retained until new information is supplied using SAM:

3

- Primary log file path name = `/.secure/etc/auditlog_1`
- Primary log file switch size (AFS) = 5,000 kbytes
- Auxiliary log file path name = `/.secure/etc/auditlog_2`
- Auxiliary log file switch size (AFS) = 1,000 kbytes
- Monitor wake up interval = 1 minute
- Allowable free space minimum (FSS) = 20%
- Start sending warning messages when log reaches = 90%

See Chapter 7, for more information about audit log files and auditing parameters and Chapter 2 for guidelines about setting these parameters at your first invocation of auditing.

Prerequisite

- Set up your secure system according to the procedures in Chapter 2.

Procedure

To alter any of the above parameters:

1. Invoke SAM by typing:

   ```
   /usr/bin/sam
   ```

2. Choose `Auditing and Security->`.
3. Choose `Users`.
4. From the "Actions" menu, choose `Log/Monitor Params...`
5. Tab to the field(s) you wish to change and type your changes.

   When specifying a new path name for an audit log file, that file must be empty or nonexistent.

   Any changes to this screen will be retained as new defaults at reboot.

Note If auditing is currently off, it will be turned on when activating your changes.

6. Activate OK.

Turn Auditing On/Off

When auditing is currently on, the audit status message in any of the functional area windows ("Users," "Events," or "System Calls") reads `Auditing is currently ON`, and when auditing is currently off, this message reads `Auditing is currently OFF`.

3

Prerequisite

- Set up your secure system according to the procedures in Chapter 2.

Procedure

1. Invoke SAM by typing:

   ```
   /usr/bin/sam
   ```

2. Choose `Auditing and Security->`.
3. Choose `Users`.
4. Take one of the following actions:
   a. To turn auditing *on*, from the "Actions" menu, choose `Turn Auditing ON`.
   b. To turn auditing *off*, from the "Actions" menu, choose `Turn Auditing OFF`.
5. You are informed by a message box of the change you have requested. Activate [OK] to continue.
6. The "User Audit Status" window now indicates the change you requested.

Note Use this menu item to turn auditing on and off when audit log file and monitor parameters stay the same. When changing audit log file and monitor parameters, choose the `Log/Monitor Params...` menu item to make the changes and turn auditing on or off.

Select Users to be Audited

An audit flag is set to on for all existing users at initial conversion to a trusted system. To change the selection of audited users on your system follow the procedure below.

Prerequisite

- Set up your secure system according to the procedures in Chapter 2.

Procedure

1. Invoke SAM by typing:

   ```
   /usr/bin/sam
   ```

2. Choose `Auditing and Security->`.
3. Choose `Users`.
4. Highlight the users whose auditing status is to be changed.
5. To change the auditing status of the highlighted users, choose one of the following from the "Actions" menu:
   a. `Audit Users`
   b. `Don't Audit Users`
6. You will receive a confirmation message from the system. Activate [OK] to continue.

Note The y or n next to the name of each highlighted user will be changed to reflect your choice.

New users added to the system are automatically audited. You must enter this screen to turn auditing off for any new user if you do not wish him audited.

Note Changes to this option take effect at next login. For example, if you add **user1** to the list of users being audited, and **user1**

is currently logged on, his actions are not audited until he has logged off and back on.

Select Events to be Audited

The following event types are recommended as a minimal set of auditable events. They are supplied by the system as default event types and become activated when the Perform Task key is pressed.

- Login
- Moddac
- Admin

All system calls associated with a specific event type are audited automatically when that event type is selected for auditing. See Chapter 7 for a list of event types and their associated system calls and commands.

Note

An auditing record is written when:

- The event type being performed is selected to be audited, and
- The user initiating the event has been selected to be audited.

The `login` event is an exception to these conditions. Once selected, this event will be recorded whether or not the user logging in has been selected for auditing.

Prerequisite

- Set up your secure system according to the procedures in Chapter 2.

Procedures

1. Invoke SAM by typing:

   ```
   /usr/bin/sam
   ```

2. Choose `Auditing and Security->`.
3. Choose `Events`.
4. Highlight the events whose auditing status is to be changed.
5. To change the auditing status of the highlighted events, choose one of the following items from the "Actions" menu:

a. `Audit for success only`
b. `Audit for failure only`
c. `Audit for both success and failure`
d. `Audit for neither success nor failure`

6. You will receive a confirmation message from the system. Activate OK to continue.

Note The y or n in the appropriate column next to the name of each highlighted event will be changed to reflect your choice.

Select System Calls to be Audited

All system calls associated with selected event types are audited automatically. The following is a list of the system calls and commands associated with the system supplied default event types.

`login`	*login*(1), *init*(1M)
`moddac`	*chmod*(2), *chown*(2), *umask*(2), *fchown*(2), *fchmod*(2), *setacl*(2), *fsetacl*(2)
`admin`	*stime*(2), *cluster*(2), *swapon*(2), *settimeofday*(2), *sethostid*(2), *privgrp*(2), *setevent*(2), *setaudproc*(2), *audswitch*(2), *setaudid*(2), *setdomainname*(2), *reboot*(2)

To select additional system calls to be audited, or to remove system calls that have been selected due to their association with a selected event type, follow the procedure below.

Prerequisite

- Set up your secure system according to the procedures in Chapter 2.

Procedure

1. Invoke SAM by typing

   ```
   /usr/bin/sam
   ```

2. Choose `Auditing and Security->`.
3. Choose `System Calls`.
4. Highlight the system calls whose auditing status is to be changed.
5. To change the auditing status of the highlighted system calls, choose one of the following items from the "Actions" menu:
   a. Audit for success only
   b. Audit for failure only
   c. Audit for both success and failure
   d. Audit for neither success nor failure

6. You will receive a confirmation message from the system. Activate [OK] to continue.

Note The y or n in the appropriate column next to the name of each highlighted system call will be changed to reflect your choice.

View Audit Logs

Auditing accumulates a lot of data. SAM gives you the opportunity to select the data you want to view.

Procedure

1. Invoke SAM by typing:

   ```
   /usr/bin/sam
   ```

2. Choose **Auditing and Security->**.
3. Choose **Users**.
4. From the "Actions" menu, choose **View Audit Log...**
5. Tab to the field(s) you wish to change and type your changes.

   You may change the following items:

   a. Whether the log output is directed to the screen or to a file.
   b. The name of the file to which log output is to be directed.
   c. Whether you wish to view the records of successful events, failed events, both, or neither.
   d. Which log file you wish to read.
   e. Which user login you wish to view.
   f. Which **tty** you wish to view.
   g. Which events or system calls you wish to select for viewing.

   When specifying a new path name for an audit log file, that file must be empty or nonexistent.

6. Use the default settings on this screen or alter them to suit your needs.
7. Activate [OK].

Note It may take a few minutes to prepare the record for viewing when working with large audit logs.

Examples

The following sample record from an audit log file shows a failed attempt to open the secure password file:

3

```
users and aids:
hesh
69
Selected the following events:
5
All ttys are selected.
Selecting successful & failed events.
TIME              PID E  EVENT   PPID    AID   RUID   RGID   EUID   EGID TTY

----------------------------------------------------------------ing ---------------
890620 14:31:30   570 F      5    567     69     69     11     69     11 ttysx
[ Event=open; User=hesh; Real Grp=topaz; Eff.Grp=topaz; Origin=hpkang;  ]

        RETURN_VALUE 1 = -1;
        PARAM #1 (file path) = 1 (cnode);
                               0x000e0000 (dev);
                               6418 (inode);
                 (path) = /.secure/etc/passwd
        PARAM #2 (int) = 0
        PARAM #3 (int) = 438
----------------------------------------------------------------ing ---------------
```

The initial lines identify information for which the audit log file was searched. Following in tabular form the record shows:

- The year, month and day (in this case 1989, June, 20th)
- Time of day (in this case 1400 hours, 31 minutes, 30 seconds)
- Process ID (in this case 570)
- Whether the action failed or succeeded (in this case F for failed)
- Event—a number identified with the event type (in this case 5)
- Parent Process ID (in this case 567)
- Audit ID of the user (in this case 69)
- Real User ID (in this case 69)
- Real Group ID (in this case 11)
- Effective User ID (in this case 69)
- Effective Group ID (in this case 11)
- The tty of the audited event (in this case **ttysx**)

Additional data recorded for the selected system calls corresponds to code in the synopsis of the *open*(2) manual page in *HP-UX Reference*:

```
int open (path, oflag [, mode])
```

Self-auditing commands conform to the same tabular format. For example,


```
------------------------------------------------------------ing ---------------
890621 14:12:36  1063 S   9218   1062     76      0     11      0     11 ttyu0
[ Event=login; User=nlb; Real Grp=topaz; Eff.Grp=topaz; Origin=hpkang;  ]

SELF-AUDITING TEXT: User= nlb uid=2419 audid=76 Successful login
------------------------------------------------------------ing ----------------
```

Interpreting the Audit Log Data

Since auditing produces large amounts of data be discriminating in the selection of event types and users to audit. Selection of all events and all users for auditing could result in an overwhelming amount of data as well as a very rapid filling of disk space. Use of self-auditing programs where appropriate for your operation can help significantly. See Chapter 7 for more information about self-auditing programs and for guidelines on managing audit log resources.

When viewing your audit data be aware of the following anomalies:

- Audit data may be inaccurate when programs that call auditable system calls supply incorrect parameters. For example, calling the *kill*(2) system call with no parameters (i.e. `kill()`) produces unpredictable values in the parameter section of the audit record.
- System calls that take filename arguments may not have device and inode information properly recorded. The values will be zero if the call does not complete successfully.
- Auditing of the superuser while using the SAM interface to change event or system call parameters will result in a long audit record. For example, when you add an event type to be audited in SAM a record will be produced for each event type and system call being audited, for both success and failure if both are turned on, *not just* for the new event type being added.

4

Controlling User Access to Directories and Files

This chapter shows

- How to list and change directory and file permissions.
- How to restrict access to directories and files using access control lists (ACLs).
- How to protect directories from file tampering.

Assigning File Permissions to Specific Users and Groups.

In a traditional HP-UX system, you use the `ls -l` command to see a full listing of a file's permissions and `ls -ld` to list a directory's permissions. The *chmod*(1) command enables you to change the mode permissions of directories and files.

On a secure system, to assign permissions to specific users and groups, you can use the *lsacl*(1) command to see what permissions are associated with a given file, and the *chacl*(1) command to change the access control lists of the file.

Listing File Permissions with the lsacl Command

To see exactly who can access certain files and directories and what permissions are set on them, use the `lsacl` command. (Refer to *HP-UX Reference*, *lsacl*(1) and *acl*(5) manual pages, for detailed specifications and information on access control lists.) The general form of the `lsacl` command is as follows:

```
$ lsacl file_name
```

The system will respond with a listing in this general form:

4

(user.group,mode) . . . file_name

where *(user.group,mode)* is an ACL entry (there will always be at least three ACL entries), and *file_name* is the name of the file or directory for which you want a listing. The following table describes what each element of an ACL entry means:

This element ...	Means ...
user	The user's login name; a percent sign (%) in this position means all users.
group	The user's group; a percent sign (%) in this position means all groups.
mode	The permissions allowed: read (`r`), write (`w`), execute/search (`x`). A dash (-) means a permission is denied (for example, `rw-` means that read and write permissions are allowed, but execute/search permission is denied).

An Example Using the lsacl Command

Suppose you run the `lsacl` command on `myfile`:

```
$ lsacl myfile
(sally.adm,rw-) (leslie.%,r--) (%.mtg,r--) (%.%,---) myfile
```

Interpret the above listing as follows:

`(sally.adm,rw-)`	The user **sally** while in the group **adm** has read and write permissions (`rw-`) on **myfile**.
`(leslie.%,r--)`	The user **leslie** while in any group (%) has read permission (`r--`) on **myfile**.
`(%.mtg,r--)`	Any user (%) in the group **mtg** has read permission (`r--`) on **myfile**.
`(%.%,---)`	No other user (%) from any other group (%) has read, write, or execute permissions (`---`) on **myfile**.

The following section explains how you can change file and directory permissions using ACLs.

Changing permissions with the chacl command

Like the *chmod*(1) command, which enables you to change general permissions on your files and directories, the *chacl*(1) command allows you to grant or restrict file access to or from specific users or specific groups of users. (See the *chacl*(1) manual page in *HP-UX Reference* for detailed specifications.)

For example, you may want a particular file or directory to be accessible to only one other person. To do so, you can use the `chacl` (*ch*ange *acl*) command to grant or restrict file access to or from specific users or groups of users.

How the chmod and chacl Commands Interact

Since you can use both the **chmod** and the **chacl** commands to change access permissions, you need to be aware of how the two commands interact.

- The **chacl** command is a superset of the **chmod** command. Any specific permissions you assign with the **chacl** command are added to the more general permissions assigned with the **chmod** command. For example, suppose you use the **chmod** command to allow only yourself write permission to **myfile**. You can use the **chacl** command to make an exception and allow your manager write permission to **myfile** also. Users other than yourself and your manager will still be denied write permission as previously specified by the **chmod** command.

4

- Use **chmod** with the -A option when working with files that have additional permissions assigned with the **chacl** command. The additional permissions will be deleted if you fail to use the -A option with **chmod**. For example, to keep ACL permissions on the file **myfile**, enter

```
$ chmod -A  number  myfile
```

Using the chacl Command

The general form for the **chacl** command is as follows:

```
$ chacl 'user.group operator mode' file_name
```

user — Indicates the user's login name; a percent sign (**%**) in this position means all users.

group — Indicates the user's group; a percent sign (**%**) in this position means all groups.

operator — Indicates whether you are adding or denying permissions to existing ACL entries or whether you are adding new ACL entries. A plus sign (**+**) adds permissions; a minus sign (**−**) denies permissions; an equals sign (**=**) means "this permission exactly," and usually is used when adding new ACL entries.

mode Indicates the permissions allowed. Possible modes are read (`r`), write (`w`), and execute/search (`x`). An *operator* (explained above) immediately precedes the mode (for example, `+rw` adds read and write permissions; `-rw` denies read and write permissions).

file_name Indicates the name of the file for which you want to specify access.

Examples Using the chacl Command

Suppose your user name is `leslie`, and you are in the group `users`. If you need to limit access to `myfile` so anyone in your group can read the file, but *only* you and your manager (`arnie`) can both read from and write to the file, you might follow these steps:

- First, use the `chmod` command to protect `myfile` so your group can read it, but only you can both read from and write to the file. (*Note:* If you have any previously set ACL entries, `chmod` deletes them unless you use the `-A` option):

```
$ chmod 640 myfile
```

- To view the permissions you just set, run the `ll` command on `myfile`:

```
$ ll myfile
-rw-r-----   1 leslie   users          236 Dec  8 14:04 myfile
```

  The `ll` command shows that *owner* (`leslie`) has read and write permission on `myfile`; *group* (`users`) has only read permission; *other* has no access to the file.

- Now use the `lsacl` command to compare the long listing above with the ACL entries:

```
$ lsacl myfile
(leslie.%,rw-)(%.users,r--)(%.%,---) myfile
```

  The `lsacl` command shows that `leslie` in any group (`%`) has read and write permission to `myfile`; anyone (`%`) in the group `users` has read permission to `myfile`; and everyone else has no access to `myfile`.

- To specify that your manager, who is in a different group (**mtg**), should have read and write access to **myfile**, use the **chacl** command to create a new ACL entry for **myfile**:

```
$ chacl 'arnie.mtg=rw' myfile
```

- Now run the **ll** command on **myfile**:

```
$ ll myfile
-rw-r------+  1 leslie   users          236 Dec  8 14:04 myfile
```

4 The plus sign (+) at the end of the permissions string means that additional permissions (in the form of optional ACL entries) exist for **myfile**.

- Run the **lsacl** command to view these optional ACL entries:

```
$ lsacl myfile
(arnie.mtg,rw-)(leslie.%,rw-)(%.users,r--)(%.%,---) myfile
```

The **lsacl** command now shows that, in addition to the previous ACL entries, **arnie** of the group **mtg** has read and write permission to **myfile**.

Since directories are also files, the same technique of using **lsacl** and **chacl** can be used to set selective access to directories.

The table below further illustrates ways you can use the **chacl** command:

If you want to ...	Use this command ...
Create a new ACL entry allowing the user **cyc** in any group (%) read and write (**=rw**) access to **myfile**.	`$ chacl 'cyc.%=rw' myfile`
Modify an existing ACL entry allowing all users (%) in all groups (%) read (**+r**) access to **foofile**.	`$ chacl '%.%+r' foofile`
Modify an existing ACL entry denying all users (%) in the **adm** group write (**−w**) access to **afile**.	`$ chacl '%.adm−w' afile`
Create a new ACL entry denying user **jon** in the **mkt** group read, write, or execute/search access to **olddir**.	`$ chacl 'jon.mkt=' olddir`

Protecting Directories from File Tampering

If a directory is writable, anyone can remove its files, regardless of the permissions on the files themselves. The only way to ensure that no files can be removed from a directory is to remove write permission from that directory.

For maximum protection this technique can be applied to the directory of a user account.

1. Create your confidential directory. If you want to even hide its name from routine view, use a dot (.) as the first character of its name.

   ```
   $ mkdir .secret
   ```

2. Move sensitive files into the directory.

   ```
   $ mv report1 report2 .secret
   ```

3. Use `ll -d` to view the permissions set on the directory.

   ```
   $ ll -d .secret
   drwxrwxr-x 1 leslie   users 236  Dec 8 14:04 .secret
   ```

4. Use the *chmod*(1) command to change the permission modes of the directory to restrict access to the owner only.

   ```
   $ chmod g=,o= .secret
   ```

   The symbolic notation denies read, write, and execute access to members of your group (g) and all other (o) users.

5. List the permissions on the directory.

   ```
   $ ll -d .secret
   drwx------ 1 leslie   users 236  Dec 8 14:04 .secret
   ```

6. To allow another person to access that directory also, use the *chacl*(1) and *lsacl*(1) commands.

   ```
   $ chacl 'arnie.%=rwx' .secret
   $ ll -d .secret
   drwx------+ 1 leslie   users 236  Dec 8 14:04 .secret
   ```

   The plus sign (+) after the permission modes indicates that ACLs are set on the directory.

7. Use the **lsacl** command to list the ACLs.

```
$ lsacl .secret
(leslie.%,rwx)(%.users,---)(%.%,---)(arnie.%,rwx) .secret
```

In this example, only the users **leslie** and **arnie** can access the directory **.secret**.

Default Permissions

4

The default **umask** setting in a secure (trusted) system is should be set to **022**. This means that all directories created will have a default permission mode of **755**, granting access of **drwxr-xr-x**. All files created will have the default permission mode of 644, granting access of **rw-r--r--**.

5

Protecting your Operating System and Files

This chapter provides tools that you can use to:

- Assess your system files for potential security risks before converting to a secure (trusted) system
- Review your system files for routine security maintenance
- Locate suspicious files in case of security breach

Before you install the new release of HP-UX with security features, you *must* inspect your system for files that pose a potential risk to security.

Thereafter, on a regularly scheduled and ongoing basis, and whenever you suspect any breach of security, examine your system to locate problem files.

To better understand HP-UX risk factors (especially **setuid** programs), read Chapter 9.

Looking for setuid and setgid Programs

Since they pose the greatest security liability to your system,

- Note which programs are **setuid** and **setgid**
- Stay vigilant of any changes to them
- Investigate further any programs that appear to be needlessly **setuid**
- Change the permission of any unnecessarily setuid program to **setgid**

The long form of the `ls` command (`ll` or `ls -l`) shows **setuid** programs by listing **s** instead of **-** or **x** for the owner-execute permission. It shows **setgid** programs by listing **s** instead of **-** or **x** for the group-execute permission.

- To locate **setuid** or **setgid** programs in your hierarchy, list the files returned by the `find` command:

```
$ find $HOME \( -perm -4000 -o -perm -2000 \) \
-exec ls -ld {} \;
```

- Periodically, find and list all **setuid** programs:

```
find / -hidden -perm -4000 -exec ls -ld {} \;
```

  Or, list **setuid** and **setgid** programs in system directories:

```
find [directories] \( -perm -4000 -o -perm -2000 \) \
-exec ls -ld {} \;
```

- Review the output for unexpected results.

  You can expect to find system files, but they should display the same permissions as shown in the `pdf` files provided, unless you have customized them. (See the section, "Maintaining your Secure System," in Chapter 2 for discussion of `pdf` files.)

  Setuid-to-root programs are the most significant.

  Setuid to other users are less significant, but might warrant concern.

  Users normally should not have **setuid** programs, especially **setuid** to users other than themselves. Do not run a user's personal **setuid** program, since you may not know what that program may be doing.

- Examine the code of all programs imported from external sources for destructive programs known as "Trojan Horses."
- Never restore a **setuid** program for which you have no source to examine. If you suspect trouble, delete the file.
- Most locally written programs reside in **/usr/local/bin**, and may be restored if you know exactly what they do.

Verifying Password File Security

Before conversion, passwords are located in **/etc/passwd**; in a trusted system, passwords are located in **/.secure/etc/passwd**.

Do not permit any empty password fields in either password file. The conversion tools replace any empty password fields in **/etc/passwd** with **,..** to force the user to select a password upon first login. However, even this leaves a potential for security breach, by allowing any user to set the password for that account.

- Before converting to a trusted system, look for empty password fields or fields that force the user to set a password by typing

```
awk -F: '$2 == "" || $2 == ",.."' < /etc/passwd
```

- Once the system has been converted to a trusted system, periodically look for password fields or fields that force the user to set a password by typing

```
awk -F: '$2 == "" || $2 == ",.."' < /.secure/etc/passwd
```

- On a trusted system, the password fields of the **/etc/passwd** file should contain an asterisk. To find deviations, type

```
awk -F: '$2 != "*"' /etc/passwd
```

- Every user should have a unique user ID (**uid**), and should have the same **uid** on each computer system for which he or she has an account.

  To locate duplicate user IDs, type

```
sort -t: +2n /etc/passwd | \
awk -F: '{if (duplicate == $3) print; duplicate = $3}'
```

  Only **root** and users with root privileges (user ID of **0**) should be found. Inspect those accounts to make sure that the users warrant root privilege. Assign new user IDs to any ordinary users with duplicate IDs.

Protecting User Accounts

- Home directories should not be writable, because they allow any user to add and remove files from them. To find home directories that are writable by others:

```
find 'awk -F: '{print $6}' /etc/passwd' \
-prune -perm  -02 -exec ls -ld '{}' \;
```

Here is an example of output:

```
drwxrwxrwx  6 cwagner  topaz  2048 Feb 16 07:18 /mnt/topaz/cwagner
drwxrwxrwx  2 ekr      topaz  1024 Aug 10  1988 /mnt/topaz/ekr
drwxrwxrwx 10 achoy    topaz  2048 Jun  8 12:49 /mnt/topaz/achoy
drwxrwxrwx  2 gnuss    topaz  1024 May  5 15:12 /mnt/topaz/gnuss
drwxrwxrwx 15 jacques  topaz  3072 Jun 13 17:35 /mnt/topaz/jacques
```


Set directory permissions to **drwxrwxr-x**, to grant write permission only to your group members:

```
chmod 775 [directory]
```

If confidentiality requires that even your group members be denied write access to your directory, set the permissions to **drwxr-xr-x**:

```
chmod 755 [directory]
```

- Users' **.profile**, **.cshrc**, and **.login** files should not be writable by anyone other than the account owner. To find those that are writable by group members or others, run:

```
find 'awk -F: '{printf "%s/.profile\n", $6}' /etc/passwd' \
-prune -perm  -022 -exec ls -l '{}' \;

find 'awk -F: '{printf "%s/.cshrc\n", $6}' /etc/passwd' \
-prune -perm  -022 -exec ls -l '{}' \;

find 'awk -F: '{printf "%s/.login\n", $6}' /etc/passwd' \
-prune -perm  -022 -exec ls -l '{}' \;
```

The message "find: cannot stat ... " means that the user does not have the file listed. This is a warning only and can be ignored.

- A user's `.rhosts` file should not be readable or writable by anybody other than the owner. This precaution prevents users from guessing what other accounts you have, as well as preventing anyone from editing your `.rhosts` file to gain access to those systems.

  To find readable or writable `.rhosts` files:

  ```
  find 'awk -F: '{printf "%s/.rhosts\n", $6}' /etc/passwd' \
  -prune -perm -066 -exec ls -l '{}' \;
  ```

  In this command, `awk` prints the sixth field (the home directory) of every entry in `/etc/passwd`, followed by the string `.rhosts` and a carriage return (`\n`). This output is then evaluated by the options of `find`, which searches for and displays a long listing (`ls -l`) of any entries that grant read or write permissions to group or other (`-066`).

  The message "find: cannot stat ... " is returned if you do not have a `.rhosts` file. This is a warning only and can be ignored.

- Use of a `.netrc` file is discouraged, since it bypasses `.login` authentication for remote login and even contains the user's unencrypted password. If used at all, `.netrc` must not be readable or writable by anyone other than its owner.

  To find readable or writable `.netrc` files:

  ```
  find 'awk -F: '{printf "%s/.netrc\n", $6}' /etc/passwd' \
  -prune -perm -066 -exec ls -l '{}' \;
  ```

- Some systems maintain an `/etc/securetty` file, which should not be writable. If it exists on your system, use

  ```
  ls -l /etc/securetty
  ```

  to verify that its modes are correct.

Safeguarding Network Security

This section provides techniques for checking security of network control files.

Identifying Administrative Domain

Following are steps to identify and control an administrative domain. (See Chapter 10 for a full discussion of administrative domains.) The example used in these steps assumes that the computers you can reach on your network are named for corporate departments, such as **finance**, **sales**, and **mktg**, and that you are working on a machine named **hq**.

1. List the nodes to which you export file systems.

   ```
   cat /etc/exports
   ```

   The **/etc/exports** file lists entries that consist of the path name of a file system followed by a series of names of computers and names of groups of computers.

   Any entry consisting of only a path name without being followed by a computer name is a file system available to every computer on the network. All path names should be associated with specific computers.

   The **/etc/exports** entries might contain names of groups of computers. You can find out what individual machines are included in a group by checking **/etc/netgroup**.

   If the output of the command **cat /etc/exports** is:

   ```
   /u1 mfg mktg
   /u2 personnel
   /u3 mfg design quality
   ```

   you can conclude that **hq**, **mfg**, **mktg**, **design**, **personnel** and **quality** are part of one administrative domain.

2. List the nodes that have equivalent password data bases.

   ```
   cat /etc/hosts.equiv
   ```

   The `/etc/hosts.equiv` file lists the names of computers with equivalent password files. If the output of this command is:

   ```
   legal
   mfg
   mktg
   quality
   ```

   you can conclude that **hq**, **mfg**, **mktg**, **design**, **personnel**, **quality** and **legal** are part of one administrative domain.

3. Verify that each node in the administrative domain does not extend privileges to any unincluded nodes.

   5

   You must repeat Steps 1, 2 and 3 on each of these machines: **mfg**, **mktg**, **design**, **personnel**, **quality** and **legal**.

   For example, suppose that most of these machines extend privileges only to computers within the administrative domain, but **mktg** also exports file systems to **sales**. Repeat Steps 1, 2 and 3 for the machine called **sales**.

4. Control root and local security on every node in your administrative domain. A user with superuser privileges on any machine in the domain can acquire those privileges on every machine in the domain. Follow the system security procedures described elsewhere in this manual to protect root privilege on each machine.

5. Maintain consistency of user name, **uid**, and **gid** among password files in your administrative domain. To verify consistency of systems in the domain, compare their password files.

   For example, if you are working on system **hq** and wish to check consistency with **mfg**, whose **/** file system is remotely mounted to **hq** as **/nfs/mfg/**, type the following sequence of commands:

   ```
   % awk -F: '{printf "%s %s %s\n", $1, $3, $4}' \
   /etc/passwd > /tmp/hq-pwd
   % awk -F: '{printf "%s %s %s\n", $1, $3, $4}' \
   /nfs/xyz/etc/passwd > /tmp/mfg-pwd
   % diff /tmp/hq-pwd /tmp/mfg-pwd
   ```

The **awk** commands print the values of the first, third, and fourth fields of **/etc/passwd** (user name, **uid**, and **gid**, respectively) to temporary files. Then, any output produced by running the **diff** command on the temporary files indicates inconsistency between your two **/etc/passwd** files. Edit one or both **/etc/passwd** files to make them consistent.

The **/etc/passwd** files of a local (**hq**) and remote (**mfg**) system can be compared by using *rcp*(1) instead of NFS. You might choose to use these commands if you do not have NFS and **mfg** is listed in the **/etc/hosts.equiv** file of **hq**, making it a member of **hq**'s administrative domain:

```
% rcp mfg:/etc/passwd /tmp/mfg-passwd
% awk -F: '{printf "%s %s %s\n", $1, $3, $4}' \
/etc/passwd > /tmp/hq-pwd
% awk -F: '{printf "%s %s %s\n", $1, $3, $4}' \
/tmp/mfg-passwd > /tmp/mfg-pwd
% diff /tmp/hq-pwd /tmp/mfg-pwd
```

6. Maintain consistency among any group files on all nodes in your administrative domain.

   If you are working on system **hq** and you wish to check consistency with system **mfg**, and **mfg**'s **/** file system is remotely mounted to **hq** as **/nfs/mfg/**,

   ```
   % diff /etc/group /nfs/mfg/etc/group
   ```

   If you see any output, your two **/etc/group** files are inconsistent. The manual entry on *diff*(1) will explain how to interpret the output. Edit one or both of the files, making them the same.

   For a non-NFS method:

   ```
   % rcp mfg:/etc/group /tmp/mfg-group
   % diff /etc/group /tmp/mfg-group
   ```

   In both cases, if you see no output, the two **/etc/group** files are consistent, and you are done.

Verifying Permission Settings on Network Control Files

Modes, owners, and groups on all system files are set carefully. Their correct values are recorded in Product Description File (`pdf`) databases, which are provided with all shipped HP-UX filesets (see Chapter 2 for a discussion of `pdf`). All deviations from these values should be noted and corrected.

Pay particular attention to network control files, which reside in `/etc`, and are notable targets because they provide access to the network itself. Network control files should never be writable by the public. Among them are:

`networks`	Network names and their addresses
`hosts`	Network hosts and their addresses
`hosts.equiv`	Remote hosts allowed access equivalent to the local host
`services`	Services name database
`exports`	List of file systems being exported to NFS clients
`protocols`	Protocol name database
`inetd.conf`	Internet configuration file
`netgroup`	List of network-wide groups

5

Controlling File System Export

The `/etc/exports` file defines which file systems can be exported to other systems. Each one-line entry should have at least two fields: the first is the name of the file system being exported, the second and subsequent name the systems to which the file system can be exported. If fewer than two fields are present, the file system can be shipped anywhere in the world.

Verify that no file system can be universally exported:

```
sed -e '/^\#/d' -e '/^[space, tab]*$/d' /etc/exports | awk 'NF < 2'
```

This command examines `/etc/exports`, removes all comment lines, removes all null lines (lines containing only spaces or tabs), then searches the file for lines with fewer than two fields.

Maintaining Device File Security

To ensure the integrity of your devices, check the device special files for potential security hazards.

- Character and block special devices should reside only in **/dev**. To find and list files of a particular type (such as **b** for block special files, **c** for character special files, or **d** for directories), type

  ```
  find / -hidden -name /dev -prune -o -type b -exec ls -l {} \;
  ```

  Devices found anywhere other than **/dev** should be removed unless they have been added for a specific reason. No individual user should own a device file.

- Before mounting disks or other mountable devices of unknown origin, first check its files for special files and **setuid** programs:

  ```
  ncheck -s /dev/dsk/[device name]
  ```

  This program, which takes several moments to run, provides only a first impression of the unmounted device. It returns inode and file names relative to slash (/). It cannot give a long listing of the unmounted files' permissions.

  If you decide that any of the listed **setuid** files might pose risk, you can mount the device special file with its **setuid** bit inhibited, as follows:

  ```
  /etc/mount -o nosuid /dev/dsk/[device name] [mount point]
  ```

  Consider this only a temporary measure, and be sure to investigate the problem promptly. Delete the file if it threatens system security.

What to Do if You Discover a Security Breach

A security breach can present itself in many different ways:

- Someone might report unexpected or destructive behavior by a common program.
- You might notice a sudden increase in your system's load average, causing the computer not to respond well.
- Read/write permissions or ownership might be changed from what you expect.
- The byte count of any system files changes unexpectedly.

Anything that seems to deviate from normal system behavior might suggest tampering. If you suspect a security breach, such as a virus or worm, handle it by limiting its immediate impact.

5

1. Shut down the system. If you feel you can give users a warning, use the more courteous **shutdown** command:

   Series 800:

   ```
   /etc/shutdown [-r|-h] [grace_period]
   ```

   or

   Series 300:

   ```
   /etc/shutdown [-r|-h] [-d device] [-f lif_file] [grace_period]
   ```

2. Bring your system to a sudden halt by using

   ```
   /etc/reboot -q
   ```

   if:

   An infiltrating program is actively corrupting the system.

   Allowing for a grace period by using **shutdown** might allow more time for further corruption.

   The system might not respond to system load.

Note Once rebooted, a series 800 system asks to autoboot from the primary boot path enabled. (The series 300 returns without asking.) Press any key within 10 seconds to override this option. See your system administrator's manual, and follow procedures for rebooting your system in single-user state.

3. You want to bring the system up in a single-user state, its barest minimum. This limits the impact, subject to symptoms. From a single-user state, analyze the problem and clean it up.

   This brings up the system without any local file systems being mounted. By killing all processes, you have also stopped a virus or worm from causing more damage.

4. Mount all file systems, using

   ```
   /etc/mount -a
   ```

   Until their integrity has been verified, set restrictive directory permissions (`drwx------`) to prevent users from accessing the questionable files. This is a short-term solution only.

   What do you think went wrong? Examine the audit trail, password file, and login files for clues.

5. Run the `pdf` programs (see Chapter 2), `ncheck` and `find` commands (given earlier in this chapter) to compare file size from the previously backed-up system to the current one. Examine the date files were last written, byte count, inodes, ownership. Suspect any files whose sizes differ unexpectedly. Remember, however, that some system files, especially network files, might have been customized, and therefore differ from the default system software.

6. Copy contaminated files to tape to save as evidence.

7. Under some circumstances, you might not be able to reboot, or you might not trust the reboot program (`/etc/init`) itself. If so, you must reinstall your system.

8. If you are uncertain of the scope of damage, we recommend that you *reinstall* HP-UX from the system tapes.

Even if you believe you have found the offending file and understand the architecture of the problem, the possibility still exists that the worm, virus, or Trojan Horse might have propagated itself. Reinstall the system completely.

9. After reinstalling, you must decide if you have corrupted any user files, or other files not reinstalled from tape.
10. Mount users' home directories and run the **find** and **ncheck** commands described earlier in this chapter to uncover any additional compromised files.
11. If the breach was an unauthorized access of your machine, under most circumstances the point of entry will be apparent. Disable those accounts, replacing the password entries with an asterisk. The root user then has to change the password at the console, by hand.
12. Inform all system users of a security breach and ask them to check their accounts for anything unusual. Instruct users to run **ls -lt** to look for unexpected changes to files, such as time of last modification for file creation or mode change, which might suggest tampering.
13. Analyze evidence to determine how the breach occurred and what can be done to prevent recurrences.

6

Planning for Security

This chapter

- Discusses the need for a comprehensive security policy.
- Defines the Trusted Computing Base, and concepts of discretionary access control, and least privilege.
- Describes typical software threats.
- Advocates distributing tasks to various personnel, to implement and maintain security policy.

The U.S. Computer Security Act of 1987 casts new urgency on computer security in all business segments. It stipulates that if financial loss occurs due to computer fraud or abuse, the company, not the perpetrator, is liable for damages. Thus, ultimate responsibility for safeguarding information lies with individual businesses themselves.

Systems processing sensitive or proprietary data are inherently targets for mischief or damage. Damage can be intentional or accidental, physical, or result from malicious use of software. Since UNIX was developed for an environment of cooperating users, not all security flaws have been corrected programmatically. Unauthorized persons gaining access to sensitive material might wreak considerable havoc to the system.

To protect system and data integrity, HP recommends that you establish a comprehensive security policy to govern computer use.

Security Policy

A computer security policy is a statement of rules that govern the behavior of users to ensure system and data integrity.

Some basic principles govern good security:

- Commit management to security.
- Control physical equipment.
- Educate employees to know what is expected of them.
- Design administrative procedures to increase security.
- Segregate and compartmentalize data.
- Disconnect unused terminals and mass storage devices.
- Never perform any task as superuser that can be performed with lesser privilege.
- Do not trust what others can alter.
- Require users to be on the system purposefully, on a "need-to-know" basis.
- Have users report any unusual or irresponsible activities to authorities. These activities might include unaccounted-for programs or unexpected software behavior.

6 Besides software features, administrative support is essential for achieving a workable security policy. When drafting a security policy, be sure you address the following questions:

- What facilities require protection?
- Which data warrants protection?
- Who is allowed access to your system and under what circumstances?
- What permissions and protections are required to maintain security?
- How can you enforce system security policy by physical, procedural, and system mechanisms?

Physical Security Policy

Physical security safeguards system hardware from damage. It protects software from corruption due to environmental conditions and assures that unauthorized personnel are denied access to areas containing system equipment. Hardware includes the central processing unit (CPU), system console, terminals, and other peripherals such as printers, disk drives, and tape drives. Software includes the operating system, programs, and data.

- Restrict physical access to areas containing system equipment by
  - Using perimeter controls, such as locked computer rooms, fenced building sites, and guards at building entrances.
  - Using antitheft protection designed for desktop computers.
  - Issuing keys and ID badges.
  - Physically securing access to terminal wiring and network cables.
- Safeguard sensitive or proprietary data by
  - Keeping media archived offsite in a locked facility.
  - Erasing obsolete data.
  - Shredding or securely disposing of console logs or printouts.

Procedural Security Policy

Although practices may differ depending on the type of computer involved, procedural security policy should govern the following:

- Use of equipment and systems operation
- Management of software and data, including the following:
  - How computer-processed information can be accessed, manipulated, and monitored to maintain system safeguards.
  - Information handling throughout its life cycle.
  - Use of system security features, including frequency of audit review and analysis.
  - Physical storage of data.

Common security practices include the following:

- Restrict login access to software to those with legitimate need.
- Have users log off or use the `lock` command when not using their terminals.
- Decentralize computer duties by rotating operators.
- Store backup tapes at bonded, offsite depositories.

System Security Policy

System security tasks such as auditing should be performed by authorized security personnel only. However, users might use security features such as ACLs, which are applicable to user files.

Maintaining system security involves:

Identification of Users	All users must have a unique login identity, or ID, which identifies them as legitimate system users. An ID consists of an account name and password, and without it, a user cannot log into the system.
Authentication of Users	When a user logs in, the system authenticates his or her password by checking for its existence in **/etc/passwd** or **/.secure/etc/passwd**.
Authorization of Users	At a system level, HP-UX provides two kinds of authorized computer use—regular and superuser. Individual users also may be granted or restricted access to system files through traditional file permissions and access control lists.
Auditing of Users	HP-UX enables you to audit computer usage by user, system call, and event. See Chapter 2 for setting up auditing, Chapter 3 for auditing tasks, and Chapter 7 for auditing concepts.

6

Educating Users

All users are responsible for security. A security policy is effective only if your users are informed of its contents and trained in its use.

- Centralize security responsibilities with a clearly defined security officer, and make sure that all users know whom to call if a security breach is suspected.
- Prepare a set of security guidelines, and distribute it to all computer users.
- Have your security guidelines reviewed by management to establish compliance at all levels.
- Review and update your guidelines periodically as your technology and needs change.
- Disseminate information about the policy changes promptly.
- Consider installing a file of security guidelines in every new user's account.
- Consider factors of human nature in working to gain acceptance of a security policy.

- Emphasize the benefits derived by a sound security policy.
- Keep security policy consistent with the "corporate culture" of your organization.
- Do not make the system any more restrictive than necessary. Poorly chosen or excessively rigid security measures often force users to develop loopholes to maintain productivity.

Trusted Computing Base (TCB)

The Trusted Computing Base (TCB) contains all the elements of the computer system (hardware, firmware, and software) necessary to create a protected environment, and allow for its use in processing a range of sensitive or classified information. The TCB consists of the kernel and all key operating system components, including **login**, **getty**, **backup**, **restore**, **sam**, and other utilities essential to run and support the system.

The TCB enforces security policy by controlling access to code and data on which the protection is based and by providing auditing capability.

Discretionary Access Control (DAC)

The *Department of Defense Trusted Computer System Evaluation Criteria* defines discretionary access control (DAC) as "a means of restricting access to objects based on the identity of subjects and/or groups to which they belong. The controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission on to any other subject."

Traditional HP-UX DAC provides sets of permissions for user, group, and others. Access control lists (ACLs) extend this mechanism by allowing users to add sets of permissions or restriction for specific users and groups.

Subjects

Subjects are active entities—persons, processes, or devices—that cause information to flow among objects or that change the system state. All subjects are a potential target of auditing on a trusted system.

Objects

Objects are passive entities—files, directory trees, programs, bits, bytes, fields, processors, stack registers, video displays, keyboards, clocks, printers, network nodes—anything that contains or receives information on a system. Since access to an object implies access to the information the object contains, objects require protection. Certain controlled objects require special attention:

- Root directory
- Sensitive files such as `.rhosts` and password files
- Configuration files such as `inetd.sec`
- Public directories
- Log files

To ensure security, set **umask** as restrictively as possible and assign access control lists (ACLs) as needed to all files whose access you intend to limit. See Chapters 4 and 8 for ACLs and *umask*(1) in the *HP-UX Reference*.

Least Privilege

The principle of least privilege requires each subject in a system to be granted only as much privilege as is needed to perform authorized tasks. Users should be able to access information based only on valid "need to know." These criteria help to limit damage resulting from accident, error, or unauthorized use.

Accountability

To ensure that individual users are held accountable for their activities online, the conversion program to a trusted system creates an audit ID (**aid**) that identifies every user uniquely, and associates that user with every process invoked. The **aid** and HP-UX auditing functionality enable authorized personnel to evaluate auditable events—actions potentially capable of allowing access to, generating, or releasing sensitive information. See Chapters 3 and 7, for information on auditing and audit IDs.

Typical Software Threats

Since HP-UX is a relatively open system, once having logged in, the user has access to virtually all available functionality.

Although HP-UX has numerous built-in software restrictions (including passwords, account and group management, and access control), it is vulnerable to penetration, leading to destruction or compromise of material or data. Be aware of the following penetration techniques:

Spoofing A technique used to masquerade a hostile program as a system program. A classic example prompts the user to log in, thereby capturing the person's login and password. Instruct users to report any unexpected system query for their password once logged in.

Time bombs A program that remains inactive until triggered by a program such as an *at*(1) job, scheduled to run at a set time. A time bomb might be as harmless as a program that displays blinking Christmas trees on everyone's terminal screen on Christmas eve or as harmful as a program set to reformat the root disk at 8:00 pm October twelfth.

Trap door A hidden software or hardware mechanism that circumvents system protections by acting differently from documented behavior. A trap door is often activated subtly, such as by "random" key sequence at a terminal, or it might be built into a program by the software developer.

Trojan Horse A seemingly useful computer program containing additional (hidden) functionality that exploits the program's capabilities to the detriment of security. For example, a program might copy a sensitive file for the creator of the Trojan Horse.

Trojan Horses are frequently located in a directory searched before the system directories in a user's PATH. Trojan Horses also might arrive online as a "gift" with some "undocumented features."

- Never begin PATH with the user's current (.) directory.

- Be vigilant against Trojan Horses when unpacking archives of software. The **shar** format is especially dangerous. Never unpack archives as **root**.
- Never blindly compile and execute source code without first examining it for software threats.
- Be wary of using programs that require privilege to operate, especially if you have the binaries without the source. If you must use them, back up the system completely before loading the code, and watch for changes to the password file and other sensitive areas.

Viruses Viruses are covert code segments programmed to reproduce rapidly, debilitate hosts, and spread through a variety of carriers.

Worms Worms are independent programs that, like viruses, migrate through a system for harmful purposes. The 1988 Internet Worm invaded 6000 UNIX systems, including 308 government-affiliated sites.

6

Caution

- Do not run or copy software whose origin you do not know. Games and pirated software are especially suspect.
- Know where you got your code!
- Use, and encourage all users to use, the HP-UX security features provided to the fullest practical extent.

Key Security Personnel

One technique for increasing accountability in system administration is to distribute security-related responsibilities among different users. In the following model, recommended by the National Computer Security Center, the System Security Officer is responsible for overall system security, while the System Administrator keeps the system running and works with the System Security Officer to plan the system's overall hardware and software needs.

System Security Officer Tasks

- Initiates and monitors auditing policy.
- Determines which users and events are audited.
- Maintains the secure password system.
- Initializes DAC privileges on public files.
- Authorizes new user accounts.
- Checks file systems for **suid/sgid** (set user ID/set group ID) programs.

System Administrator Tasks

- Implements auditing procedures.
- Inspects and analyzes audit logs.
- Administers group and user accounts.
- Repairs damaged user files and volumes.
- Updates system software.
- Sets system configuration parameters.
- Collects various system statistics.
- Disables and deletes accounts.
- Makes periodic system checks.
- Monitors repeated login attempts.
- Periodically scans file permissions.
- Deals with invalid **su** (substitute user, or superuser) attempts and invalid network requests.

Operator Tasks

- Installs security-relevant software.
- Performs routine maintenance, such as backups.
- Performs on-line terminal and device tests.
- Responds to user requests for routine system maintenance.

Systems Programmer Tasks

- Installs system upgrades.
- Performs dump analysis.
- Writes programs that conform to security criteria.

7

Understanding Auditing

Auditing allows you to record user access to objects. The resulting record can show such things as repeated attempts by a user to assume a level of privilege that exceeds his approved level (i.e., superuser). By checking the audit record you can monitor system activities.

Once you follow the procedures outlined in this book for installing your new operating system and converting to a secure (trusted) system, you are ready to begin auditing. The auditing subsystem allows you to audit selected users performing selected types of activities on your system.

Users are audited through their association with their **audit ID** numbers, assigned automatically by the system. The audit ID (a number ranging from 0 to 60,000) is kept in the `/.secure/etc/passwd` file, which can be read only by superusers. When an audited user logs in, any action (also selected for auditing) performed by that user is traceable to him through his audit ID.

You must select actions you wish to monitor, such as file deletions. You may choose to audit any action you select:

1. When the action is attempted and either succeeds or fails
2. Only when the action is attempted and fails
3. Only when the action is attempted and succeeds

The types of activities you may want to monitor are:

- Use of identification and authentication mechanisms, such as login
- File manipulations such as opens, closes, deletions and creations
- Security-relevant actions such as turning the auditing system on and off, and changing the password file
- Printing

Event Types Which are Selectable for Auditing

To simplify the selection of actions to be audited, system activities with like behavior are grouped together in categories called **event types.** Selecting one of these event types to be audited automatically turns auditing on for all processes in that category. However, you can select system calls for auditing without selecting the event type that they belong to. Also, system calls selected for auditing due to their association with a particular event type can be removed. See Chapter 3 for details on how to select events and/or system calls to be audited.

The following table shows the event types (and the processes associated with them) that you can select for auditing:

Table 7-1. Event Types

Event Type	Description of Action	Associated Processes
`create`	Log all creations of objects (files, directories, other file objects)	*creat*(2), *mknod*(2), *mkdir*(2), *semget*(2), *msgget*(2), *shmget*(2), *shmat*(2), *pipe*(2)
`delete`	Log all deletions of objects (files, directories, other file objects)	*rmdir*(2), *semctl*(2), *msgctl*(2)
`moddac`	Log all modifications of object's Discretionary Access Controls	*chmod*(2), *chown*(2), *umask*(2), *fchown*(2), *fchmod*(2), *setacl*(2), *fsetacl*(2)
`modaccess`	Log all access modifications other than Discretionary Access Controls	*link*(2), *unlink*(2), *chdir*(2), *setuid*(2), *setgid*(2), *chroot*(2), *setgroups*(2), *setresuid*(2), *setresgid*(2), *rename*(2), *shmctl*(2), *shmdt*(2), *newgrp*(1)
`open`	Log all openings of objects (file open, other objects open)	*open*(2), *execv*(2), *ptrace*(2), *execve*(2), *truncate*(2), *ftruncate*(2), *lpsched*(1M)
`close`	Log all closings of objects (file close, other objects close)	*close*(2)
`process`	Log all operations on processes	*exit*(2), *fork*(2), *vfork*(2), *kill*(2)

Table 7-1. Event Types (continued)

Event Type	Description of Action	Associated Processes
`removable`	Log all removable media events (mounting and unmounting events)	*smount*(2), *umount*(2), *vfsmount*(2), *rfa_netunam*(2)
`login`	Log all logins and logouts	*login*(1), *init*(1M)
`admin`	Log all administrative and privileged events	*stime*(2), *cluster*(2), *swapon*(2), *settimeofday*(2), *sethostid*(2), *privgrp*(2), *setevent*(2), *setaudproc*(2), *audswitch*(2), *setaudid*(2), *setdomainname*(2), *reboot*(2), *sam*(1M), *audisp*(1M), *audevent*(1M), *audsys*(1M), *audusr*(1M), *chfn*(1), *chsh*(1), *passwd*(1), *pwck*(1M), *init*(1M)
`ipccreat`	Log all `ipc create` events	*socket*(2), *bind*(2), *ipccreate*(2), *ipcdest*(2)
`ipcopen`	Log all `ipc open` events	*connect*(2), *accept*(2), *ipclookup*(2), *ipcconnect*(2), *ipcrecvcn*(2)
`ipcclose`	Log all `ipc close` events	*shutdown*(2), *ipcshutdown*(2)
`ipcdgram`	Log ipc datagram transactions	*udp*(7) user datagram
`uevent1`, `uevent2`	Log user-defined events	See "Streamlining Audit Log Data", next page

Streamlining Audit Log Data

Some processes invoke a series of auditable actions. To reduce the amount of audit log data collected and to provide for more meaningful notations in the audit log files, some of these processes are programmed to suspend auditing of the actions they invoke and produce one audit log entry describing the process that occurred. Processes programmed in this way are called **self-auditing programs.**

For example, when the `login` program is invoked it opens, reads and closes several files to gather the necessary information for logging a user into the system. Each of these opens, reads and closes may be auditable actions. However, since `login` is a self-auditng program, the auditing subsystem does not record these subsequent actions. Instead it makes one entry in the audit log file to record that **UserX** has logged into the system.

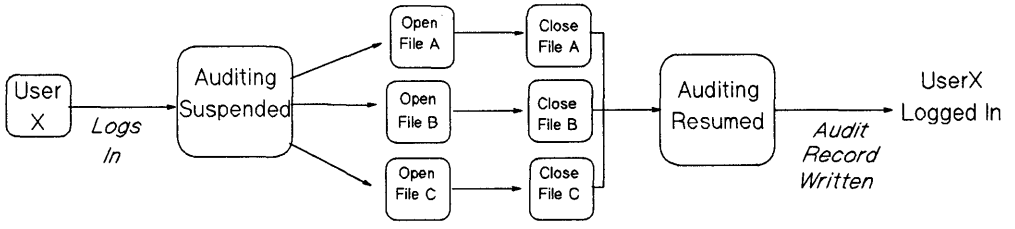


Figure 7-1. Self-Auditing Process

The following processes have self-auditing capabilities:

chfn(1)	Change finger entry
chsh(1)	Change login shell
login(1)	The login utility
newgrp(1)	Change effective group
passwd(1)	Change password
audevent(1M)	Select events to be audited
audisp(1M)	Display the audit data
audsys(1M)	Start or halt the auditing system
audusr(1M)	Select users to be audited
init(1M)	Change run levels, users logging off.
lpsched(1M)	Schedule line printer requests
pwck(1M)	Password/group file checker
sam(1M)	The System Administration Manager performs self-auditing for certain security relevant tasks.

Self-auditing programs are useful for streamlining the audit data collected on your system. Therefore, two event types, UEVENT1 and UEVENT2, are reserved for self-auditing programs you may want to write. See the following section for guidelines on how to write self-auditing programs.

7 Guidelines for Writing Programs to Streamline Auditing Data

Users with superuser permission may wish to write their own programs to streamline auditing data. Use the **audswitch** and **audwrite** system calls to suspend process-by-process auditing of a program and generate a summary audit record. You can suspend auditing (`audswitch(AUD_SUSPEND)`), choose key points in the program to generate an auditing record; (**audwrite**); and then resume regular auditing (`audswitch(AUD_RESUME)`).

Example of Self-Auditing Program

```
#include <stdio.h>
#include <sys/audit.h>

void writerec();

struct self_audit_rec audrec;

main(){

    char *errmesg, *wr_string;
    int fd;

    /* suspend auditing for this self auditing process */
    if (audswitch(AUD_SUSPEND)) {
        fprintf(stderr,"You do not have privilege
        to access this command\n");
        exit(1);
    };

    /* do the work that isn't to be audited */
    /* open the test file */
    if((fd=creat("test_file", 0666))== -1) {
        errmesg="could not create test file\n";
        fprintf(stderr,errmesg);
        writerec(errmesg,3);
    }

    /* write "that's all folks" to the test file */
    wr_string="that's all folks\n";
    if(write(fd,wr_string,strlen(wr_string))== -1) {
        errmesg="could not write to test file\n";
        fprint(stderr,errmesg);
        writerec(errmesg,3);
    }

    /* tell the user what we did */
```

```
    errmesg="successfully created *that's all folks* file\n";
    printf(errmesg);

    /* write *successfully* audit record */
    writerec(errmesg,0);

}


/* routine to write an audit record and exit */
static void
writerec(mesg,err)
    char *mesg;
    int err;
{
    extern struct self_audit_rec audrec;

    /* copy output message int audit record body - aud_body.text */
    strncpy (audrec.aud_body.text, mesg, MAX_AUD_TEXT);

    /* initialize the audit record header */
    /* 1. set ah_error to return value */
    /* 2. set ah_event to specify the type of event */
    /* 3. set ah_len to specify the length of the aud_body.text */
    audrec.aud_head.ah_error = err;
    audrec.aud_head.ah_event = EN_UEVENT1;
    audrec.aud_head.ah_len = strlen(mesg);

    /* resume auditing */
    audswitch(AUD_RESUME);

    /* write the audit record */
    audwrite(&audrec);

    exit(err);

}
```

7

Note If the auditing system is turned off at the time your program is run, **audwrite** returns successfully, but no auditing record is written.

Refer to the **audwrite** manual page in section 2 of *HP-UX Reference* for more information on how to write self-auditing programs.

The Audit Record

For each event audited, the following information is recorded in the audit log file:

- Date and time of event
- Audit ID of the user generating the event
- Subject (user/process)
- Type of event
- Success and/or failure of event
- Origin of request (tty) for identification/authentication events
- Name of an object introduced to or deleted from a user's address space
- Description of modifications made by the system administrator to the user/system security databases
- Other information relevant to the event

See the section entitled "Viewing Audit Logs" in Chapter 3 for examples of typical audit records.

Audit Log Files

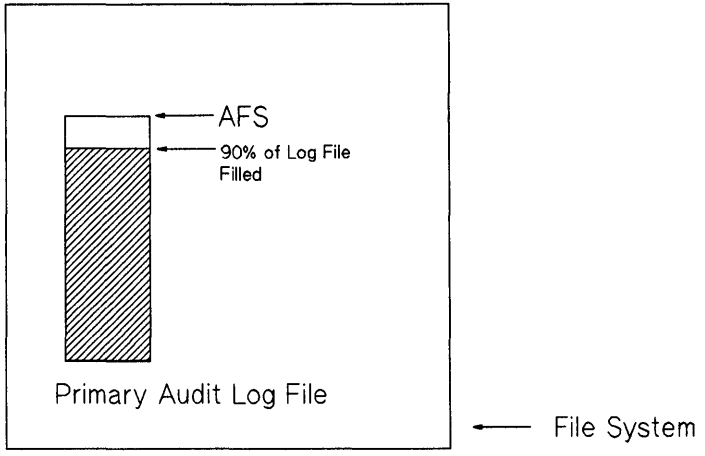
All auditing data is written to an audit log file. You can specify two files to collect auditing data, the **primary log file** and the (optional) **auxiliary log file.** These files should reside on two different file systems. The growth of these files (and the file systems on which they reside) is closely monitored by the audit overflow monitor daemon, **audomon**, to insure that no audit data is lost.

The primary log file is where audit records begin to be collected. When this file approaches a predefined capacity (its **Audit File Switch (AFS)** size), or when the file system on which it resides approaches a predefined capacity (its **File Space Switch (FSS)** size), the auditing subsystem issues a warning. When either the AFS or the FSS of the primary log file is reached, the auditing subsystem attempts to switch to the auxiliary log file for recording audit data. If no auxiliary log file is specified, the primary log file continues to grow. The following diagrams show what happens as this file grows.

The example assumes that:

- Only the primary audit log file has been specified
- It resides on a file system with no other user files competing for space

STAGE 1

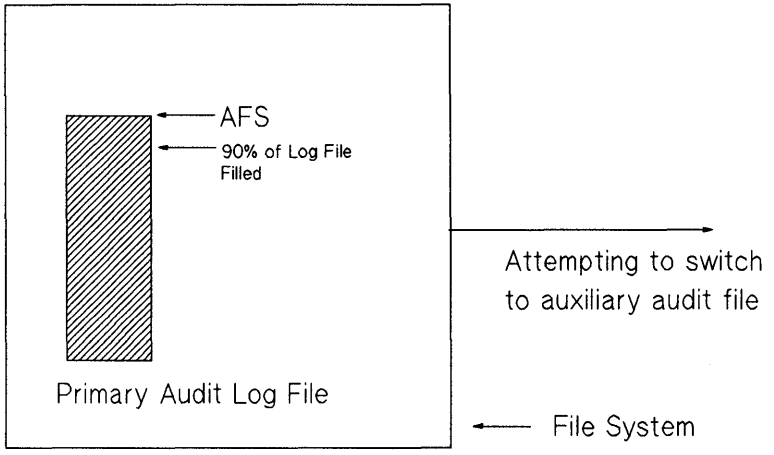


Message: "Audit system approaching Audit File Switch Point. Current audit file size = _____ kilobytes"

Figure 7-2. The Primary Audit Log Approaches AFS

The primary audit log has reached 90% of its AFS size. Audomon, which is monitoring the state of the auditing system, issues the warning message shown to the system console.

STAGE 2



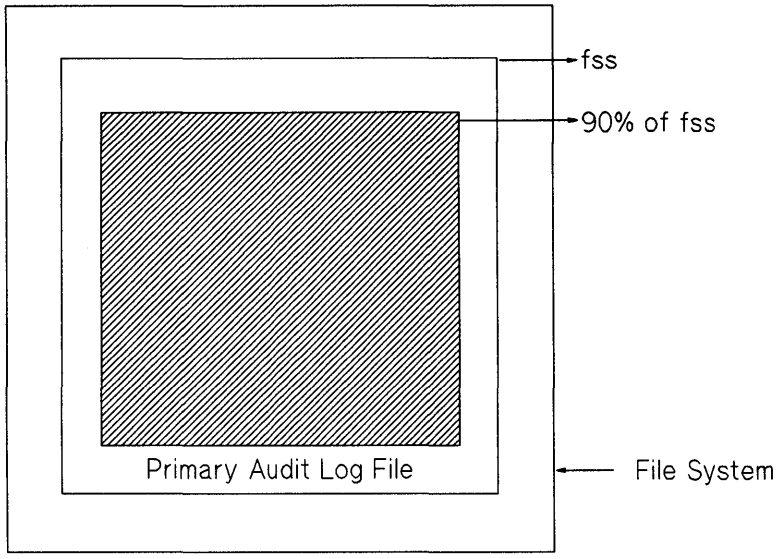
Message: "Current audit file size is ____ kbytes. An attempt to switch to the backup file failed."

Figure 7-3. Primary Audit Log Reaches AFS

7

The primary audit log has passed the first warning point and reached the AFS size. The system attempts to switch to an auxiliary audit log file, but finding none issues the indicated message, periodically, to the system console.

STAGE 3

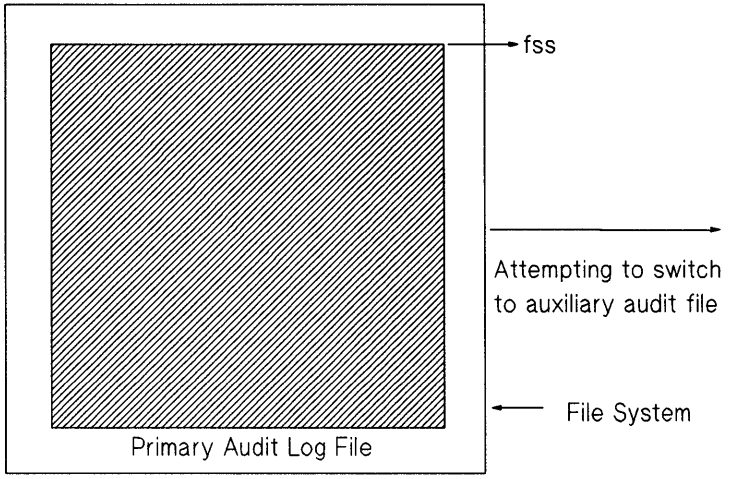


Message: "Audit system approaching file space switch point. Free space left on file system _____%"

Figure 7-4. Primary Audit Log Approaches FSS

The primary audit log has grown past its AFS size and reached 90% of the space allocated to it on the file system. The message sent indicates that the audit file system is approaching capacity.

STAGE 4



Message: "File system of audit file has _____% free space left. An attempt to switch to the backup file failed."

Figure 7-5. Primary Audit Log Reaches FSS

7

The primary log file has reached FSS. The message shown is sent periodically to the system console.

If other activities consume space on the file system, or the file system chosen has insufficient space for the AFS size chosen, the File System Switch point could be reached before the Audit File Switch point.

Caution If the primary audit log continues to grow past the FSS point, a system defined parameter, **min_free**, could be reached. *All auditable actions are suspended for regular users* at this point. Restore the system to operation by archiving the audit data, or specifying a new audit log file on a file system with space.

TWO LOG FILES ON
SEPARATE FILE SYSTEMS

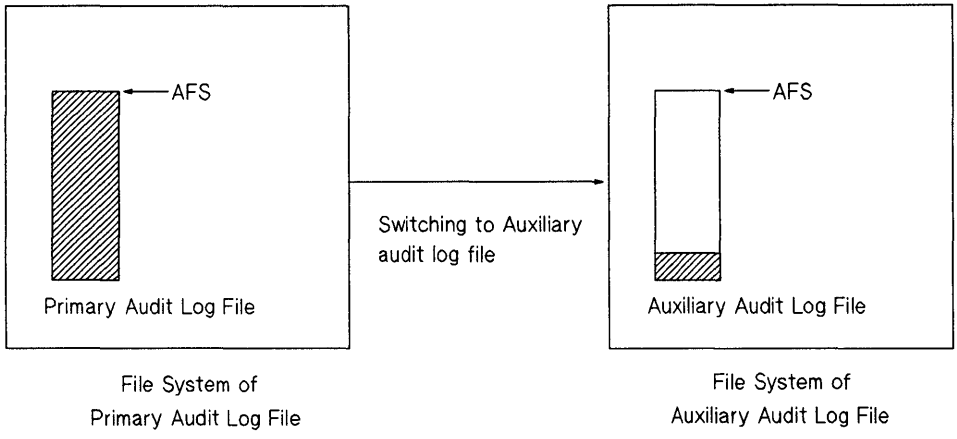


Figure 7-6. Primary and Auxiliary Audit Log Files

If, as shown, an auxiliary log file is specified on a different file system from the primary log file, when the primary log file reaches either AFS or FSS, the system switches to the auxiliary log file. From this point the auxiliary log file becomes the new primary log file and the growth stages proceed as in the first example.

Note Specify both a primary and an auxiliary log file that reside on different file systems for best results.

See Chapter 2 and Chapter 3 for information about defining your auditing system parameters.

Managing Audit Log Resources

Because auditing produces large amounts of data, you need to manage auditing resources carefully. The following guidelines should prove helpful:

- Be selective when choosing events and users for auditing.
- Once an audit file is full, immediately respecify the destination file for audit log entries.
- Archive the full audit file onto tape to free up disk space.

An on-line audit file should be retained for at least 24 hours and all audit records stored off-line should be retained for a minimum of 30 days.

Guidelines for Administering Your Auditing System

We recommend using the following guidelines when administering your system:

1. Check the audit logs once a day at a minimum.
2. Review the audit log for unusual activities, such as:
   - Late hours login
   - Login failures
   - Failed access to system files
   - Failed attempts to perform security-relevant tasks
3. Quickly remove users who no longer have access to the system.
4. Prevent overflow of audit file by archiving daily.
5. Revise current selectable events periodically.
6. Revise audited users periodically.
7. Do not follow any pattern or schedule for event or user selection.
8. Set site guidelines. Involve users and management in determining these guidelines.

Performance Considerations

Auditing increases the system overhead. When performance is a concern (such as in a real-time environment), the system administrator has to weigh security versus performance. Being selective about what events and users are audited can help reduce the impact of auditing to an acceptable level.

Using Auditing in a Diskless Environment

Audit log files are **context dependent files** (**cdf**s). On a diskless cluster with server and clients, each cluster node (cnode) records its own audit data. All cnode records are merged into a single audit file viewable using SAM. This merged audit record is what you see when you use the "View Audit Files" window in SAM. To view a specific sub-element specify the cdf wanted. For example, type `/.secure/etc/auditlog_1+/`*cnode_name*.

8

Planning User Access to System and Files

This chapter covers basic information on

- Password security
- System and user file permissions
- File access control using access control list entries (ACLs)

Password Security

The password is the most important individual user identification symbol. With it, the system authenticates a user to allow access to the system. Since they are vulnerable to compromise when used, stored, or even known, passwords must be kept secret at all times.

The System Security Officer and every user on the system must share responsibility for password security. Security policy should be based on the following assumptions:

- A password is assigned when a user is added to the system.
- A user's password should be changed periodically.
- The system must maintain a password database.
- Users must remember their passwords and keep them secret.
- Users must enter their passwords at authentication time.

Password Responsibilities

The System Security Officer performs the following security tasks:

- Assigns initial system passwords.
- Maintains proper permissions on the `/etc/passwd` and `/.secure/etc/passwd` files.
- Assigns initial passwords to all new users.
- Establishes password aging.
- Deletes/nullifies expired passwords, and user IDs and passwords of users no longer eligible to access the system.

Every user must observe the following rules:

- Keep his password secret at all times.
- Change the initial password immediately.
- Remember the password (never write it down).
- Report any changes in status and any suspected security violations.
- Change his password periodically.
- Make sure no one is watching when entering the password.
- Choose a different password for each machine on which there is an account.

Criteria of a Good Password

Observe the following guidelines when choosing a password:

- A password should have at least six characters. It must contain at least two alphabetic and one numeric or special character. Special characters can include control characters and symbols such as asterisks and slashes.
- Do not choose a word found in a dictionary, even if you spell it backwards. Software programs exist that can find and match it.
- Do not choose a password easily associated with you, such as a family or pet name, or a hobby.
- Do not use simple keyboard sequences, such as **asdfghjkl**, or repetitions of your login (e.g. login is ann; password is annann).
- Misspelled words or combined syllables from two unrelated words make suitable passwords.
- Consider using a password generator that combines syllables to make pronounceable gibberish.

Note Keep your password private. It is a security violation for users to share passwords.

Password Encryption

All passwords are encrypted immediately after entry, and stored in the `/.secure/etc/passwd` file. Only the encrypted password is used in comparisons.

Two Password Files

A trusted system maintains two password files: **/etc/passwd** and **/.secure/etc/passwd**. Every system user has entries in both files, and **login** looks at both entries to authenticate login requests.

/etc/passwd

The **/etc/passwd** file is used to authenticate a user at login time. The file contains descriptions of every account on the HP-UX system. Each entry consists of seven fields, separated by colons. When a system is converted to a trusted system, the encrypted password, normally held in the second field of **/etc/passwd**, is moved to the **/.secure/etc/passwd** file and an asterisk holds its place in the **/etc/passwd** file. A typical entry of **/etc/passwd** in a trusted system looks like this:

```
robin:*:102:99:RobinTewes:/mnt/robin:/bin/csh
```

The fields contain the following information (listed in order):

1. User (login) name, consisting of up to 8 characters.
2. Encrypted password field, held by an asterisk instead of an actual password.
3. User ID (**uid**), an integer less than 60000.
4. Group ID (**gid**), taken from the **/etc/group** file. The **gid** must be an integer less than 60000.
5. Comment field, used for identifying information such as the user's full name.
6. Home directory, the user's initial login directory.
7. Login program path name, executed when the user logs in.

Changing /etc/passwd Fields

8

The user can change the encrypted password field (second field) by invoking the *passwd*(1) command, the comment field (fifth field) with the *chfn*(1) command, and the login program path name (seventh field) with the *chsh*(1) command (see *HP-UX Reference*). The system administrator sets the remaining fields, including the user and group IDs (**uid** and **gid**) and audit ID (**aid**). The **aid** must be unique; the **uid** should be unique. (For discussion of the **aid** see Chapter 7.)

/.secure/etc/passwd

Key security elements are held in the **/.secure/etc/passwd** file, accessible only to superusers. Each entry of **/.secure/etc/passwd** consists of four fields, separated by colons. A typical entry looks like this:

```
robin:a9xOEmtQsK6qc:101:1
```

The four fields of the **/.secure/etc/passwd** file contain the following information (listed in order):

1. User (login) name, consisting of up to eight characters.
2. Encrypted password.
3. Audit ID
4. Audit flag: 1=on, 0=off

General users cannot alter any fields in **/.secure/etc/passwd**.

Eliminating Pseudo-Accounts

By tradition, the **/etc/passwd** file contains numerous "pseudo-accounts"—entries not associated with individual users and which do not have true interactive login shells. Sometimes these entries have a star in their password field; sometimes they have a password.

Some of these entries, such as **date**, **who**, **sync**, and **tty**, have evolved strictly for user convenience. To tighten security, HP is eliminating them in **/etc/passwd** so that these programs can be performed only by a user who is logged in. This also prevents a cracker who knows the password of a pseudo-account from establishing a remote file access connection to the system without a legitimate login.

Other such entries remain in **/etc/passwd** because they are owners of files. Among them are **bin**, **daemon**, **adm**, **uucp**, **lp**, and **hpdb**.

Manipulating the Password Files

The following library routines can be used to access information in the password files:

getpwent	Get password entries from **/etc/passwd** and **/.secure/etc/passwd**
getspwent	Get password entries from **/.secure/etc/passwd**
putpwent	Write password file entries to **/etc/passwd**
putspwent	Write password file entries to **/.secure/etc/passwd**

Refer to section 3 of the *HP-UX Reference* for detailed specifications.

File Permissions

HP-UX controls all privileges through user accounts. To the system, there are only two kinds of accounts—root and all others. The security system administrator must consider many factors, such as program use and how the programs function, to set appropriate file permissions that help ensure system security without losing functionality.

Consider the principle of "least privilege" and whether a user legitimately needs file access when you set permissions.

Root

Root, or the "superuser," is a user whose effective user ID is zero. Superusers can access and modify any file. In a trusted system, the superuser is also the user with the most responsibility for maintaining the security and integrity of the system.

Many commands and system calls can be executed successfully only by a superuser. Superusers can perform tasks such as the following:

- Invoke any executable command in the system.
- Override any protection placed on user files.
- Add and remove system users.
- Perform other system functions.

Root Use Guidelines

Commands and system calls used only by the system administrator are reserved for the superuser. To protect the system, observe the following:

- Restrict knowledge of the root password to the barest minimum number of people—one, if possible. The root password should be held in strictest secrecy and changed periodically.
- All root accounts should have `PATH` set (in `.profile` or `.login`) to some default that does not contain the current directory ("dot"). The following `PATH` is recommended:

  ```
  /bin:/usr/bin:/etc
  ```

- Most system administration tasks should be performed by invoking SAM, because its menus restrict choices and thus reduce damage potential.

- If the root user forgets the root password, reboot the system in single-user state, and reassign the password.
- Superusers should construct **at** and **cron** jobs carefully. When **at** and **cron** are executed, the system searches the path set by root.
- Set your file creation mask with a **umask** of 077 before creating a file. This restricts read and write permissions to the file owner by default.
- Do not leave executables where they were developed. Restrict access to executables under development.

System Files and Directories

Permissions on essential HP-UX programs should be set as restrictively as possible without losing functionality. Most should be set to prevent users from writing to them. These include:

- directories and commands contained in **/**, **/bin**, **/dev**, **/etc**, **/usr**, **/usr/bin**, **/usr/lib**, and **/usr/spool**
- **/hp-ux**
- **/etc/rc**
- **/etc/inittab**

Only **root** should be able to write to **/etc/passwd**.

Only **root** should be able to read from and write to **/etc/.secure/passwd**.

Protecting Key Subsystems

Programs with owners such as **bin**, **uucp**, **adm**, **lp**, and **daemon** encompass entire subsystems, and represent a special case. Since they grant access to files they protect or use, these programs must be allowed to function as "pseudo-accounts", with entries listed in **/etc/passwd**.

```
root:*:0:3/:/bin/sh
daemon:*:1:5/:/bin/sh
bin:*:2:2/bin:/bin/sh
adm:*:4:4/usr/adm:/bin/sh
uucp:*:5:3/usr/spool/uucppublic:/usr/lib/uucp/uucico
lp:*:9:7/usr/spool/lp:/bin/sh
```

Key to the privileged status of these subsystems is their ability to grant access to programs under their jurisdiction, without granting a user ID of 0. Instead, the **setuid** bit is set and the **uid** bit is set equal to the specified subsystem (for example, **uid** is set to **lp**).

Once set, security mediation of that subsystem enforces the security of all programs encompassed by the subsystem, not the entire TCB. However, the subsystem's vulnerability to breach of security is also limited to the subsystem files only. Breaches cannot affect the programs under different subsystems (for example, programs under **lp** do not affect those under **daemon**).

Criteria for Modes

The permission bits of HP-UX programs are set according to the principle of "least privilege," which allows access to any object based on "need to know/use" only. The number of **setuid-to-root** files has been reduced to minimize risk of Trojan Horses or other security breaches. Wherever possible, all **setuid** programs have been changed to **setgid**. By default, both owner and group are designated **bin**. Some **setuid** programs require a different owner. For example, **passwd** must be owned by **root**, but it belongs to group **bin**.

Directories that do not change often (static directories) should not have any write permission whatsoever. Their modes are set to 555; for example,

/bin	555	dr-xr-xr-x	bin	bin
/dev	555	dr-xr-xr-x	bin	bin
/lib	555	dr-xr-xr-x	bin	bin

Directories to which files are added or deleted often (dynamic directories) need write permission; for example,

/usr/spool	755	drwxr-xr-x	bin	bin

Only execute and read bits of standard binaries (both **setuid** and **setgid**) should be set:

/bin/sh	555	r-xr-xr-x	bin	bin
/bin/su	4555	r-sr-xr-x	root	bin
/bin/ps	2555	r-xr-sr-x	bin	sys

The same guidelines for static and dynamic directories are applicable to executables, scripts, and databases (e.g. **/etc/utmp**, **/etc/wtmp**).

Security Considerations for Device Files

Access to all devices in your system is controlled by device special files, which enable programs to be device independent. These files have been shipped with permission settings that enable proper use and maximal security.

If you install any other special files, please refer to the **insf** command manual entry (for series 800 systems) or the **/etc/newconfig/mkdev** file (for series 300 systems) for the correct permission settings.

Since device special files can be as vulnerable to tampering as any other file, observe the following precautions:

- Use only HP-supplied device drivers in your kernel. If you write your own device driver, you invalidate the Trusted Computing Base.
- Protect the memory and swap files, **mem**, **kmem**, and **swap**, from casual access, since these files contain potentially sabotageable user information. For example, a program that watches memory for an invocation of the **login** program might copy the password from **login**'s buffers when a user types it in.
- All device files should be kept in **/dev**.
- Write-protect all disk special files from general users, to prevent inadvertent data corruption.
- Read-protect disk special files to prevent disclosure.
- Terminal ports on UNIX systems may be writable by anyone, if you are allowing users to communicate by using the **write** or **talk** programs. Only the owner, however, should have read permission.
- Individual users should never own a device file other than a terminal device or personal printer.

Protecting Disk Partitions

Although only series 800 disks can be divided into partitions, the concepts described here are applicable to series 300 disks, by considering the entire series 300 disk as a partition.

- Disk partitions should be readable only by root.
- Since ownership and permissions are stored in the inode, anyone with write permission to a mounted partition can set the user ID for any file in that partition, regardless of the owner, bypassing the `chmod()` system call and other security checks.
- If a program, such as a database, requires direct access to the partition, that partition should be reserved exclusively for the program and never mounted. Program users should be informed that the file's security is enforced by its permission settings, rather than by the UNIX file system.

System Access by Modem

System access by modem poses security problems similar to those of device special files. To protect against system penetration, observe the following precautions:

- Require use of a hardware dial-back system for all interactive modems, rather than allowing users to dial in directly, so that the system can authenticate authorization. This will help prevent break-ins by "trial and error".
- Require an additional password from modem users, by adding an entry for the modem device in the `/etc/dialups` and possibly `/etc/d_passwd` files.
- Have users renew their dial-in accounts frequently, to ensure that access lists are kept current.
- Cancel modem access promptly when a user is no longer in company employ.
- Establish a regular audit schedule (such as four times/year) to review remote usage.
- Maintain adequate protections on the modem access to your system. Connect the modems and dial-back equipment to a single HP-UX CPU, and allow the use of network services to reach the destination CPU from that point.
- Exceptions to dial-back must be made for UUCP access. Additional restrictions are possible through proper UUCP configuration. Another potential exception is file transfer via `kermit`.
- If a security breach with unknown factors occurs, shut down the network and telephone access to the computer and inform the network administrator. Open the computer to external access only after you identify and remedy the breach.
- To maximize security when configuring a dial-back modem system, dedicate the dial-out mechanism to the dial-out function only. It should not be configured to accept dial-in. Use another modem on another telephone line for your dial-in service.

Managing User Accounts

Every user account can potentially enforce or break system security. The organization of your accounts can advance or detract from your security goals.

Ideally, users should have easy access to the files, directories, and programs needed for their work and should not be able to use resources they do not need. Through selective privileges, the user can acquire access to resources as needed. In keeping with the concept of "least privilege," users should receive the lowest privilege level needed to perform a given task and only for the time needed. (See Chapter 6 for further information.)

Note For basic instructions on managing user accounts, refer to the System Administration Tasks manual.

- Use SAM to administer user accounts.
- Group users according to task, security level, and needs. Create specific groups for people working on the projects whose work is confidential.
- The `/etc/profile` file should deny write permission to general users. Use `/etc/profile` for mandatory system settings, such as `ulimit`, instead of placing them in the individual user's `.profile` file. Further protect `/etc/profile` by using `trap` to intercept signals sent to the shell from the same user at another terminal.

  See the *sh*(1) or *ksh*(1) manual pages in *HP-UX Reference* for information on using the special commands `ulimit` and `trap`.
- Maintain a list of what systems your users have accounts on, for accountability and to assist in thoroughly removing a user once the person leaves the organization.

8

Guidelines for Adding a User Account

- Use SAM to add a user account.
- Include the user's full name and a work-related identifier (such as phone number) in the fifth field of `/etc/passwd`. Do not include confidential information, since anyone can read this file.

- Develop a `.profile` (or `.cshrc` for C-shell users) file with suitable security settings, including well-defined PATH variables and restrictive **umask**. HP-UX default PATH variables are satisfactory. For basic security, set the **umask** at 022, to create default file permissions of `-rw-r--r--`, which allow only the file owner to write a file.
- Place system directories (written as absolute path names) in the user's search path before the home directory.
- Do not include in a user's path variable any open or temporary directories such as `/usr/tmp`. This will make it less likely that the user will be trapped by a penetrator's bogus program whose name matches a standard program.
- Use a separate login name and user ID number for each user to promote accountability.
- Do not assign login names beginning with a number. **Chown** interprets such strings as user IDs.
- Do not create generic guest accounts. They increase the risk of system penetration because typically they are not well maintained. Even temporary accounts should have sole ownership.
- Place a security policy file in the home directory of a new user's account, to call attention to security policy.
- If appropriate, set password aging (see next page).

Password Aging on User Accounts

When creating a new user account with the SAM interface, you can initiate password aging by placing the necessary information in the password field.

For example, you can add a new user, force that user to enter a password at initial login and establish password aging by entering the following in SAM's highlighted field "Enter Password":

```
,81..
```

This series of characters—comma, followed by digits defined on the *passwd*(4) manual page, and two dots—causes the user to receive the message **`Your password has expired`** at initial login, and be required to enter a password. Once the password is established, the user is required to change the password every 10 weeks and no sooner than 3 weeks.

See *passwd*(4) in *HP-UX Reference* for full details on establishing password aging.

Guidelines for Deactivating an Account

Sometimes it is necessary to deactivate an account temporarily, or pending removal. Deactivate the account as soon as you establish that the user no longer needs access. The administrative task involves the following procedure:

- Before deactivating the account, invoke **who** to see whether the user is logged in. If so, communicate via **write** or other means, to ask the user to log out.
- Verify that the user has no active processes by typing:

  ```
  ps -fu <user>
  ```

  then kill any current processes before deactivating the account.
- Inform other users in the same group that you intend to remove the account. They may then copy needed files, or adjust absolute path names and environment variables.
- Invoke SAM to deactivate the account.

Guidelines for Reactivating a User Account

- Use SAM to reactivate a user account.
- To allow the user to set the password, respond to the SAM query for a password with `,..` and upon logging in, the user is prompted to set a new password.

Guidelines for Removing an Account

Remove an account as soon as a user leaves your organization or no longer requires computer access, to reduce the chance of system penetration. To remove an account, follow these steps:

- Make a backup copy of the user's directory tree, so that the account can be reconstructed if necessary.
- Remove all the files and other objects in the user's hierarchy by using the recursive form of the **rm** command:

```
rm -rf <home_directory>
```

- Locate any files owned by the user, but residing in other users' hierarchies:

```
find / -user <user> -print
```

  Consider who should own them. Remove any trivial files. Transfer ownership of necessary files to the logical owner, by using the *chown*(1) command.

- Search the system for files owned by the user after you have removed the home directory structure. Look for a file in **/usr/spool/cron/crontabs**. Remove reference to the user in **cron.allow** and **cron.deny**.

```
rm /usr/spool/cron/crontabs/<user>
```

- Look for the user's **at** jobs, which by default are found in **/usr/spool/cron/atjobs**. To remove them, type the following commands:

```
at -l
at -r <job_number>
```

  Remove reference to the user in **at.allow** and **at.deny**.

- Remove the user's mailbox from **/usr/mail**.
- Use the *find*(1) command to locate all files in which the user is explicitly included in an ACL entry, as follows:

```
find <user> -hidden -acl '<user>.%' -print
```

  If appropriate, notify the file owner, and remove the ACL entry.

- Remove reference to the user in **/usr/lib/aliases** or redirect the user's mail, if appropriate.

- A user might have accounts on other systems that you do not administer. Inform other system administrators to remove the user.
- Use SAM to remove the account.

Guidelines for Moving a User Account

Moving a user account from one system to another is trickier than it might at first seem.

- Use SAM to add the user to the new system.
- Make sure the user's `uid` and `gid` do not already exist on the new system. If either does, the user must be reassigned a new one for the new system, and the `uid` and/or `gid` of all of the user's files must be changed. Do so from the user's home directory, referred to as an absolute pathname:

  ```
  find <directory> -user <old_uid> -exec chown <new_uid> {} \;
  ```

  or

  ```
  find <directory> -group <old_gid> -exec chgrp <new_gid> {} \;
  ```

- Copy the user's files from the old to the new system.
- Remove or deactivate the user from the old system.
- Move the user's `/usr/mail/<user>` file.
- If you are acquiring a user from a system you do not administer, or the user is moving from a less to more secure environment, check the user's files carefully for `setuid/setgid` programs. Remove any `setuid/setgid` programs that might compromise security.

Guidelines for Adding a Group

Since teams of employees typically require access to common files and directories, define groups of users in the **/etc/group**, and link it to **/etc/logingroup**. All members of a group acquire the group ID (**gid**) when logging in. Users can belong to more than one group; access to resources can be governed by group membership. The administrator should have sole access to **/etc/group** and **/etc/logingroup**, which are manipulated using the **newgrp** and **chgrp** commands (for **/etc/group**) or through the **login** command or **initgroups** library routine (for the **/etc/logingroup** file). (Refer to the system administration manual for your specific system for details on their use.)

Groups may be used to deny access by individuals to objects such as shared memory segments and message queues. To deny access using groups, you can create a group that contains all users except the individual denied access. For files, use ACLs to set selective access.

When adding a group:

- Use SAM to add a group.
- Place users with similar needs or in the same project in the same group. This enables users to make links to each other's data, or conveniently access it rather than having to copy files or directories unnecessarily.
- Add a group any time a new project joins the user base.
- Do not assign group names beginning with numbers, because the **chgrp** command treats such names as a **gid**.
- Group and user IDs in **/etc/group** and **/etc/passwd** should be consistent, although no tool exists to verify the consistency. Make sure that both contain the same group ID whenever a change is made.
- Create a new group for a user who needs privacy for development or other highly confidential work.

Guidelines for Removing a Group

- Groups without members should be removed, but first find any files or directories with the group's group ID. Conduct your search from a directory where you would expect to find the **gid** and refer to the directory as an absolute pathname. Invoke the following command:

  ```
  find <directory> -group <group_name> -print
  ```

- Either assign new group IDs to these files or remove them.
- Use SAM to remove a group.

Guidelines for Modifying a Group

- Use SAM to modify a group.

Controlling File Access Selectively

HP-UX already enables non-privileged users or processes to set access to files and other objects, through the user and/or group identity (see *passwd*(4) and *group*(4) in the *HP-UX Reference*). This level of control is accomplished by setting or changing a file's permission bits to grant or restrict access by owner, group, and others (see *chmod*(2) in the *HP-UX Reference*).

Access Control Entries Defined

Access control lists (**ACLs**) are a key enforcement mechanism of discretionary access control (DAC), for specifying access to objects by users and groups more selectively than traditional HP-UX mechanisms allow, based on the user's legitimate need for access.

ACLs offer a greater degree of selectivity than permission bits by allowing the file owner or superuser to set (permit or deny) access to individual users or groups.

An ACL consists of sets of entries associated with a file to specify permissions. Each entry specifies for one user-ID/group-ID combination a set of read, write, and execute/search access permissions, and is represented in the syntax `(user.group, mode)`.

ACLs are supported for files only.

Comparing ACLs and File Permissions

To understand the relationship between access control lists and traditional file permissions, consider the following file and its permissions:

-rwxr-xr--	karen	admin	datafile

The file owner is user **karen**.
The file owner's group is **admin**.
The name of the file is **datafile**.
The file owner permissions are **rwx**.
The file group permissions are **r-x**.
The file other permissions are **r--**.

In an ACL, user and group IDs can be represented by names or numbers, found in **/etc/passwd**. The following special symbols can also be used:

%	No specific user or group
@	Current file owner or group

Base ACL Entries = File Mode Permissions

When a file is created, three base access control list entries are mapped from the file's access permission bits to match a file's owner and group and its traditional permission bits. Base ACL entries can be changed by the `chmod()` and `setacl()` system calls.

(`uid.%,mode`)	Base ACL entry for the file's owner
(`%.gid,mode`)	Base ACL entry for the file's group
(`%.%,mode`)	Base entry for other users

(Except where noted, examples are represented in short form notation. See ACL Notation, below.)

Granting Selective Access with Optional ACLs

Optional access control list entries contain additional access control information, which the user can set with the `setacl()` system call to further allow or deny file access. Up to thirteen additional user-group combinations can be specified.

For example, the following optional access control list entries can be associated with our file:

(`mary.admin, rwx`)	Grant read, write, and execute access to user `mary` in group `admin`.
(`george.%, ---`)	Deny any access to user `george` in any group.

Access Check Algorithm

ACL entries can be categorized by four levels of specificity, based on their user and group IDs. In access checking, ACL entries are compared by effective user and group IDs in the following order:

`(u.g, rwx)`	Specific user, specific group
`(u.%, rwx)`	Specific user, any group
`(%.g, rwx)`	Any user, specific group
`(%.%, rwx)`	Any user, any group

Once an ACL entry is matched, only other entries at the same level of specificity are checked. More specific entries that match take precedence over any less specific matches.

In the Berkeley model, a process might have more than one group ID, in which case more than one `(u.g, mode)` or `(%.g, mode)` entry might apply for that process. (See *setgroups*(2) in *HP-UX Reference*.) Under these circumstances, the access modes in all matching entries (of the same level of specificity, `u.g` or `%.g`) are OR'd together. Access is granted if the resulting mode bits permit. Since entries are unique, their order in each entry type is insignificant.

Because traditional UNIX permission bits are mapped into ACLs as base ACL entries, they are included in access checks.

If a request is made for more than one type of access, such as opening a file for both reading and writing, access is granted only if the process is allowed all requested types of access. Note that access can be granted if the process has two groups in its groups list, one of which is only allowed read access, and the other of which is only allowed write access. Even if the requested access is not granted by any one entry, it may be granted by a combination of entries due to the process belonging to several groups.

ACL Uniqueness

All ACL entries must be unique. For every pair of u and g values, there can be only one (u.g, mode) entry; one (u.%, mode) entry for a given value of u; one (%.g, mode) entry for a given value of g; and one (%.%, mode) entry for each file. Thus, an ACL can have a (23.14, mode) entry and a (23.%, mode) entry, but not two (23.14, mode) entries or two (23.%, mode) entries.

How to Use ACL Notation

Supported library calls and commands that manage ACLs recognize three different symbolic representations:

operator form	Used to input entire ACLs and modify existing ACLs, in a syntax similar to that used by the *chmod*(1) command.
short form	Easier to read, intended primarily for output. The *chacl*(1) command accepts this form as input, to interpret output from the *lsacl*(1) command.
long form	A multi-line format easiest to read, but supported only for output.

The base ACL entries of our example file are represented in the three notations as follows:

Operator form
```
karen.% = rwx, %.admin = rx, %.% = r
```
Short form
```
(karen.%,rwx) (%.admin,r-x) (%.%,r--)
```
Long form
```
rwx karen.%
r-x %.admin
r-- %.%
```

Some library calls and commands use a variant format, known as ACL Patterns (described later in this chapter).

Operator Form of ACLs (input only)

Each entry consists of a **user** identifier and **group** identifier, followed by one or more operators and mode characters, as in the mode syntax accepted by the *chmod*(1) command. Multiple entries are separated by commas.

```
user . group operator mode [ operator mode ]... , ...
```

The entire ACL must be a single argument, and thus should be quoted to the shell if it contains spaces or special characters. Spaces are ignored except within names. A null ACL is legitimate, and means either "no access" or "no changes," depending on context.

Each user or group ID may be represented by:

name	Valid **user** or **group** name.
number	Valid numeric ID value.
%	Any **user** or **group**, as appropriate.
@	Current file owner or group, as appropriate; useful for referring to a file's **u.%** and **%.g** base ACL entries.

An operator is required in each entry. Operators are:

- **=** Set all bits in the entry to the given mode value.
- **+** Set the indicated mode bits in the entry.
- **-** Clear the indicated mode bits in the entry.

The mode is an octal value of **0** through **7** or any combination of **r**, **w**, and **x**. A null mode denies access if the operator is **=**, or represents "no change" if the operator is **+** or **-**.

Multiple entries and multiple operator-mode parts in an entry are applied in the order specified. If more than one entry or operator for a user and group are specified, the last specified entry or operator takes effect. Entries need not appear in any particular order.

Note that the *chmod*(1) command allows only **u**, **g**, **o**, or **a** to refer symbolically to the file owner, group, other, or all users, respectively. Since ACLs work with arbitrary **user** and **group** identifiers, **@** is provided as a convenience.

The exact syntax is:

```
acl   ::=  [entry[,entry]...]
entry  ::=  id . id op mode [op mode]...
id    ::=  name | number | % | @
op    ::=  = | + | -
mode  ::=  0..7 | [char[char]...]
char  ::=  r | w | x
```

Using chacl to Set ACL Entries (operator form)

Note For basic instructions on setting access control list entries, see the information on the *chacl*(1) command in Chapter 4.

The following example sets the **%.%** entry to restrict **other** users to only reading **myfile**.

```
chacl '%.% = r' myfile
```

The following allows user **bill** in any group to write the file, assuming that no restrictive entry is more specific than the **bill.%** entry (for example, a **bill.adm** entry that denies writing). (For information on the order in which ACLs evaluate access, see Access Check Algorithm.)

```
chacl 'bill.% +w' myfile
```

The following ACL contains two entries. The first one deletes write- and adds read-capability to the entry for user 12, group 4. The second entry denies access for any unspecified user in any unspecified group. The two entries are separated by a comma and a required space.

```
chacl '12.4-w+r, %.% =' myfile
```

The following pair of entries sets the current user entry for the file's owner to allow both read and execute and results in adding write and execute capabilities for **other** users (the **%.%** entry). Note that a **mode** character (**x**) is repeated for illustration.

```
chacl '@.% = 5, %.% + xwx' myfile
```

Short Form of ACLs (input and output)

Short form differs from operator form in several ways:

- Entries are separated by parentheses rather than commas.
- Each entry specifies the `mode`, including all mode bits. It is not possible to change the mode value with `+` and `-` operators. However, the comma functions like the `=` operator in operator form.
- For clarity, hyphens represent unset permission bits in the output of the mode field and are allowed in input. This resembles the mode output style used by the `ls -l` command.

Multiple entries are concatenated. For consistency with operator form, a dot (`.`) is used to separate `user` and `group` identifiers.

On output, no spaces are printed except in names (if any). Identifier numbers are printed if no matching names are known. Either identifier can be printed as `%` for "any user or group." The mode is always represented by three characters—`r`, `w`, and `x`—and padded with hyphens for unset mode bits. If the ACL is read from the system, entries are ordered by specificity, then by numeric values of identifier parts. (See Access Check Algorithm, earlier in this chapter.)

On input, the entire ACL must be delimited by quotation marks to retain its quality as a single argument, since it might contain spaces or special characters such as parentheses. Spaces are ignored except within names. A null ACL is legitimate, and means either "no access" or "no changes," depending on context.

`User` and `group` identifiers are represented as in operator form.

The **mode** is represented by an octal value of 0 through 7 or any combination of **r**, **w**, **x**. A null mode denies access.

Redundancy does not result in error; the last entry for any **user.group** combination takes effect. Entries need not appear in any particular order.

The exact syntax is:

```
acl   ::=  [entry[entry]...]
entry  ::=  (id.id,mode)
id    ::=  name | number | % | @
mode  ::=  0..7 | [char[char]...]
char  ::=  r | w | x | -
```

Examples of ACL Entries in Short Form

The following is a sample ACL as it might be printed. It allows user **jpc** to read or execute the file while in group **adm**; it denies user **ajs** access to the file while in group **trux**; it allows user **jpc** in any group (except **adm**) to only read the file; any other user in group **bin** may read or execute the file; and any other user may only read the file.

```
(jpc.adm,r-x)(ajs.trux,---)(jpc.%,r--)(%.bin,r-x)(%.%,r--)
```

On input, the following example allows **other** users to only read **myfile**.

```
chacl '(%.%,r)' myfile
```

The following sets write-only access for user **bill** in any group.

```
chacl '(bill.%,-w-)' myfile
```

The following sets the entry for user 12 in group 4 to allow read and write.

```
chacl '(12.4,wr)' myfile
```

The following sets the base ACL entry for the file's owner to allow both read and execute, and sets write and execute capabilities for **other** (**%.%**) users.

```
chacl '(@.%, 5) (%.%, xwx)' myfile
```

Long Form of ACLs (output only)

Each entry occupies a single line of output. The mode appears first in a fixed-width field, using hyphens (unset **mode** bits) for easy vertical scanning. Each **user** and **group** identifier is shown as a name, if known, a number, if unknown, or **%** for "any user or group." Entries are ordered from most to least specific, then by numeric values of the identifiers.

Note that every ACL printed has at least three entries, the base ACL entries (that is, **uid.%**, **%.gid**, and **%.%**).

The exact syntax is:

```
acl   =  entry[<newline>entry] ...
entry  =  mode<space>id.id
mode =  <r|-><w|-><x|->
id    =  name | number | %
```

Example of ACL Long Form

Here is the same ACL as in an earlier example, printed in long form.

```
r-x  jpc.adm
---  ajs.trux
r--  jpc.%
r-x  %.bin
r--  %.%
```

ACL Patterns

Some library calls and commands recognize and use ACL patterns instead of exact ACLs. This allows operations on all entries that match the patterns. ACL syntax is extended in the following ways:

wildcard user and group IDs	A user or group name of * (asterisk) matches the **user** or **group** ID in any entry, including **%** (any user or group).
mode bits on, off, or ignored	For operator-form input, the operators **=**, **+**, and **-** are applied as follows: **=** Entry mode value matches this mode value exactly **+** These bits turned on in entry mode value **-** These bits turned off in entry mode value When only **+** and **-** operators are used, commands ignore the values of unspecified mode bits. Short-form patterns treat the mode identically to the **=** operator in operator form.
Wildcard mode values	A mode of * (asterisk) in operator or short form input (for example, **ajs.%=*** or **(ajs.%,*)**) matches any mode value, provided no other is given in an operator-form entry. Also, the mode part of an entry can be omitted for the same effect.
Entries not combined	Entries with matching **user** and **group** ID values are not combined. Each entry specified is applied separately by commands that accept patterns.

Examples of ACL Patterns

The following command locates files whose ACLs contain an entry that allows read access and denies write access to any **user.group** combination.

```
find / -acl '*.*+r-w' -print
```

The following matches entries for any user in group **bin** and for user **tammy** in any group, regardless of the entries's mode values. Matching optional ACL entries are deleted and mode values in matching base ACL entries are set to zero:

```
chacl -d '%.bin, tammy.*=*' myfile
```

The following matches all entries, deleting optional entries and setting mode values of base ACL entries to zero:

```
chacl -d '(*.*,*)' myfile
```

Working with ACL Functionality

New ACL Commands and Programs

This section describes the new programs available to manipulate a file's access control list information. For the detailed specifications, refer to the *HP-UX Reference*.

Commands

chacl(1)	Change (add, modify, delete, copy) access control lists of files, restrict access to files, convert specified ACL information into base permission bits.
getaccess(1)	List access rights to files.
lsacl(1)	List access control lists of files.

System Calls

getaccess(2)	Get a user's effective access rights to a file.
`getacl, fgetacl(2)`	Get access control list information.
`setacl, fsetacl(2)`	Set access control list information.

Library Routines

`acltostr(3c)`	Convert access control list (ACL) structure to string form.
`chownacl(3c)`	Change owner and/or group represented in a file's access control list.
`cpacl, fcpacl(3c)`	Copy the access control list and mode bits from one file to another.
`setaclentry, fsetaclentry(3c)`	Add, modify, or delete an entry in a file's access control list.
`strtoacl(3c)`	Parse and convert access control list (ACL) structure to string form
`strtoaclpatt(3c)`	Parse and convert ACL pattern strings to arrays.

Existing HP-UX Core Programs and ACLs

Optional ACL entries are affected by numerous HP-UX commands, system calls, and subroutine libraries—sometimes in unexpected ways. This section identifies issues critical to using familiar HP-UX programs on a system in which access control lists are implemented. For the detailed specifications, refer to the *HP-UX Reference* of each manual entry.

General-Purpose Commands and System Calls

chmod(1)	Use the `-A` option to retain ACLs.
chmod(2)	All optional ACL entries are deleted when `chmod()` is executed. Use `getacl()` and `setacl()` to save and restore the permission bits of ACL entries.
cpset(1)	`Cpset` does not set a file's optional ACL entries.
find(1)	New functionality enables `find` to identify files whose ACL entries match or include specific ACL patterns.
ls(1)	In long form, `ls` indicates the existence of ACLs by displaying a + after the file's permission bits.
mailx(1)	`Mailx` does not support optional ACL entries on `/usr/mail/*` files.
compact(1) *compress*(1) *cp*(1) *ed*(1) *makecdf*(1) *pack*(1) *unpack*(1)	These programs copy optional ACL entries to the new files they create.

File Archive Commands

frecover(1M), *fbackup*(1M)	Use only these programs to selectively recover and back up files. However, use the `-A` option when backing up and recovering files for use on systems that do not implement ACLs.
ar(1), *cpio*(1), *ftio*(1), *shar*(1), *tar*(1), *dump*(1m), *restore*(1m)	These programs do not retain ACLs when archiving and restoring. They use the `st_mode` value returned by `stat()`.

Configuration Control Commands

`SCCS, RCS`	The commands in these packages do not support ACLs. As a general practice, do not place optional ACL entries on system software. They are not preserved across updates.

File System Maintenance Commands

Access control lists use additional, "continuation inodes" when creating new file systems. Consider them when using the following programs:

fsck(1M)	`Fsck` returns the number of files with optional ACL entries as a value for *icont*. Use the `-p` option to clear unreferenced continuation inodes.
diskusg(1m), *ncheck*(1m)	These commands ignore continuation inodes.
mkfs(1M)	Allow for continuation inodes on new disks.

ACLs in a Network Environment

ACLs are not visible on remote files by Remote File Access (RFA) or Network File Systems (NFS), although their control over access permissions remains effective. Individual manual entries specify the behavior of various system calls, library calls, and commands under these circumstances. Use caution when transferring a file with optional entries over a network, or when manipulating a remote file, because optional entries may be deleted with no indication.

9

Recognizing Risks to Your HP-UX Operating System

This chapter addresses both preventative and emergency risk management, by

- Discussing the risks associated with **setuid** and **setgid** programs
- Describing how HP-UX safeguards the operating system environment
- Providing security guidelines for file-system management tasks, such as
  - Backup and recovery
  - Mounting and unmounting a file system
  - System shutdown

Overall Risk Management Guidelines

- Familiarize yourself with the elements of your system comprising the Trusted Computing Base (TCB). Read Chapter 6.
- Leave system file permissions set restrictively to maintain secure usage.
- Safeguard system programs by limiting the impact of **setuid** programs whenever possible.
- Use access control lists (ACLs) to restrict or permit access to data selectively.
- Both system software and user files pose security risks. Before converting to a trusted system, be sure that all existing user files are free of security hazards. To do so,
  - Review this chapter on **setuid** programs.
  - Run the commands given in "Protecting your Operating System and Files."

Once you are satisfied that the system is free of security hazards, you can install the new system software and convert to a trusted system.

Set User ID (setuid) and Set Group ID (setgid) Programs

Note These comments also apply to **setuid** and **setgid** shell scripts.

Most programs that run **setuid** can be run **setgid** just as successfully.

- A **setuid** program is one whose **setuid** bit is on.
- A **setgid** program is one whose **setgid** bit is on.
- The **setuid** and **setgid** bits are indicated by the **s** (or **S**) in the owner-execute (for **setuid**) or group-execute (for **setgid**) position of the file permission modes.

For example, in **/bin/passwd**, the **setuid** bit is set to its owner, **root**:

```
ll /bin/passwd
-r-sr-xr-x   1 root     bin        110592 Jul 21 12:55 /bin/passwd
```

In **/bin/ps**, the **setgid** bit is set to its group, **sys**:

```
ll /bin/ps
-r-xr-sr-x   1 bin      sys        116736 Jul 21 12:55 /bin/ps
```

How IDs are Set

- The **ruid** and **rgid** are inherited from the **login** process, which sets your **uid** and **gid**. The **uid** and **gid** are located in **/etc/passwd**.
- The **aid** is also set at **login** time and is located in **/.secure/etc/passwd**. The **aid** does not change when running **setuid** and **setgid** programs.
- The **login** command also changes the **ruid**, **euid**, **rgid**, and **egid**.
- The **su** command changes the **euid** and **ruid**.
- The **newgrp** command can change the **gid**.
- **Setuid** and **setgid** bits are set by using the **chmod()** system call or command. Use **chmod 4000** for the **setuid** bit, **chmod 2000** for the **setgid** bit.

Why setuid Programs Can Be Risky

Whenever any program is executed, it creates a process with four numbers—real and effective user ID (`ruid` and `euid`) and real and effective group ID (`rgid` and `egid`). Typically, these ID pairs are identical.

However, running a `setuid` or `setgid` program changes the `euid` or `egid` of the process from that associated with the owner to that of the object. The processes spawned acquire their attributes from the object, giving the user the same access rights as the program's owner and/or group.

- If the `setuid` bit is turned on, the privileges of the process are set to that of the owner of the file.
- If the `setgid` bit is turned on, the privileges of the process are set to that of the group of the file.
- If neither the `setuid` nor `setgid` bit is turned on, the privileges of the process are unchanged.
- If a program is `setuid-to-root`, the user gains all privileges available to root. This is dangerous, because the program can be used in a way to violate the TCB.

Kinds of Attacks

Most common "attacks" to the system are due to operator error! However, a system attacker can exploit `setuid` and `setgid` programs, most often in one of two ways:

- By having a `setuid` or `setgid` program execute commands defined by the attacker, either interactively or by script, or
- By substituting bogus data for the data created by a program.

A discussion of computer attacks is beyond the scope of this manual. Interested readers can refer to industry periodicals for articles on this subject. However, maintaining appropriately restrictive file permissions on system programs strengthens computer assurance.

Guidelines for Limiting setuid Power

A privileged program is more powerful than other programs, because it gives you the capability of doing something you are not otherwise allowed to do. If misused, such privilege can threaten system integrity.

Caution You must not add **setuid-to-root** programs to an existing trusted system. Adding a **setuid-to-root** program changes the system configuration, and might compromise your security.

Enforce restrictive use of privileged programs through the following suggestions:

- Use **setuid** only when absolutely necessary.
- Make sure that no **setuid** program is writable by others. Write permission allows attack by Trojan Horses.
- Whenever possible, use **setgid** instead of **setuid** to reduce the scope of damage that might result from coding flaws or breaches of security.
- Periodically search your file systems for new or modified **setuid** and **setgid** programs. If you suspect tampering, remove the **setuid** bit. Before reinstating the **setuid** bit, examine the program thoroughly, or rebuild it from trusted sources.
- Know exactly what your **setuid** and **setgid** programs do, and verify that they do only what is intended. Failing this, remove the program or its **setuid** attribute.
- If you must copy a **setuid** program, make sure that the modes are correct on the destination file.
- Write **setuid** programs so that they can be tested on non-critical data, without **setuid** or **setgid** attributes. Apply these attributes only after the code has been reviewed and all affected departments are satisfied that the new programs maintain security.

Guidelines for Writing setuid Programs

The following guidelines minimize risk when writing **setuid** and **setgid** programs:

- Do not write **setuid-to-root** programs. If possible, write the programs **setuid-to-user** (user other than root) or **setgid**.
- Make sure that a **setuid** program does not create files writable by anyone other than its intended user.
- Reset the **euid** before making an **exec** system call. Be aware that **exec** may be called within other library routines, and be wary of using routines (including **popen**, **system**, **execlp**, and **execvp**) that fork a shell to execute a program.
- When writing **setuid** programs, use *setresuid*(2) (see *HP-UX Reference* for specifications) around the pieces of code that require privileges, to reduce the window of vulnerability.
- Close all unnecessary file descriptors before calling **exec()**.
- Ensure that all variables (PATH, IFS, and **umask**) in the program's environment are sufficiently restrictive.
- Do not use the **creat()** system call to make a lock file. Use **lockf()** or **fcntl()** instead.

Secure System Initialization

HP-UX sets up a safe operating environment at the start of most **setuid-to-root** programs by calling a special library function to establish the following conditions:

- Environment variables are set to only those values necessary for the proper operation of **setuid** programs.

  Since Trojan Horses typically attack improperly set PATH and IFS variables, these are set to predetermined values: PATH is set to **/bin:/usr/bin**. IFS is set to space, tab, and newline. All other environment variables are deleted. (See *environ*(5) in *HP-UX Reference*.)

- All file descriptors other than standard input, standard output and standard error are closed. (See *close*(2) in *HP-UX Reference*.)

- All alarms are turned off. All interval timers are set to zero. (See *getitimer*(2) in *HP-UX Reference*.)

These safeguards increase assurance that known programs are executed in a known environment.

Exempted System Programs

The library function described on the previous page cannot be applied to some programs, because to do so would inhibit their specified behavior. Instead, the following programs have been carefully examined for flaws:

at(1) *crontab*(1)	Specified shell environment variables are retained when the commands are executed.
hpterm(1) *xterm*(1)	Some file descriptors other than standard input, standard output, and standard error remain open.
newgrp(1)	All exported variables retain their value.
su(1)	The environment is passed along unchanged.

Also, see Chapter 8 for guidelines on setting permissions on system files.

Backup and Recovery Guidelines in a Secure Environment

Since implementing HP-UX security features requires that you completely install (not update) HP-UX, you will need to back up and recover your entire file system. This section provides security guidance to supplement other information sources.

Backup Security Practices

Note For basic instructions on backing up system files, refer to the *System Administration Tasks* manual, and *fbackup*(1M) in the *HP-UX Reference*.

Orderly backup of files safeguards data against loss by archiving information that can be retrieved in case of system failure or user error. Be sure to establish a thorough set of procedures and adhere to them at all times.

- Use only *fbackup*(1M) and *frecover*(1M) to back up and recover files selectively. Before attempting these tasks, refer to the manual pages in *HP-UX Reference* for full information.
- Only *fbackup*(1M) and *frecover*(1M) retain access control lists (ACLs). However, use the `-A` option of these commands when backing up and recovering files for use on systems that do not implement ACLs.
- If you plan to recover the files to another system, be sure that the user's user name and group name on both systems (the system to which files are being backed up from and recovered to) are consistent.
- Remember that your backup media is sensitive material. Allow access to the media only on the basis of proven need.
- Label backup tapes and store them securely. Offsite storage provides maximum security. Keep archives for six months, then recycle the media.
- Since backup takes time and uses considerable media, set your schedule according to how much data you can afford to lose. Criteria include system use, available media, and storage space. Daily incremental and full weekly backups are recommended.

- Be mindful of the security risks posed by automated backup. Be sure that your tape holds the intended contents of your incremental backup. Safeguard your tape against tampering. Do not leave your tape unattended; retrieve it promptly when backup has completed.
- Be sure that tape is mounted on the correct output device.
- If backing up to reel tape, be sure that the write ring is in place.
- If backing up to cartridge tape, be sure that tape is write-enabled.
- Once system backup is complete, remove the write ring on the reel tape or write-disable the cartridge.
- If all files must be backed up on schedule, request that all users log off before performing the backup. However, *fbackup*(1M) does the best it can backing up active file systems without interrupting regular work. *Fbackup*(1M) warns you if a file is changing while the backup is being performed.
- Examine the logfile of latest backups to identify problems occurring during backup. The backup logfile should have restrictive permissions set.

Recovery Security Practices

Note For instructions on setting up a recovery system, refer to the *System Administration Tasks* manual.

For detailed specifications on recovering files that have been backed up, see *frecover*(1M) in *HP-UX Reference*.

Your ability to recover data after user error or system failure will endear you to coworkers and management alike. More importantly, recovery of current data is critical to protecting computerized information. Observe the following precautions:

- Use only the *frecover*(1M) command to recover files and retain access control list (ACL) information. However, use the `-A` option with *frecover*(1M) when recovering files for use on systems that do not implement ACLs.
- *Frecover*(1M) allows you to overwrite a file. However, the file retains the permissions and ACLs set when the file was backed up.
- When recovering files from another machine, you might have to execute the *chown*(1) command to set the user ID and group ID for the system on which they now reside, if the user and group do not exist on the new system. If files are recovered to a new system that does not have the specified group, the files will take on the group ownership of the person running *frecover*(1M). If owner and group names have different meanings on different systems, recovery results might be unexpected.
- Keep your recovery system tape locked up, or otherwise physically secured. Allow access to the archive only on the basis of proven need.
- Power failure should not cause file loss. However, if someone reports a lost file after a power failure, look for it in `/lost+found` before restoring it from a backup tape.
- To verify contents of the tape being recovered, use the `-I` option of *frecover*(1M) to preview the index of files on the tape. Note, however, that existing permissions of a file system are kept intact by the backup; *frecover*(1M) prevents you from reading the file if the permissions on the file forbid it.

- Examine the file listing for overly liberal permissions, **setuid**, **setgid**, or sticky bits. Change attributes if warranted, using the *chmod*(1) command with the **-A** option, since ACLs might be present. See *HP-UX Reference* for detailed specifications.
- Never recover in place any critical files such as **/etc/passwd**, or those in **/.secure**. Instead, restore the file to a **/tmp** directory, and give this directory permission **drwx------**, preventing anyone else from using it, even if others will be able to use the files after you verify their identities and move them to their final destinations. Compare the restored files with those to be replaced, to ensure that all current data is preserved. Make any necessary changes, then move the files into place.

  If this precaution is not followed, a user added to the system after you have backed up your system and possibly changed the **/etc/passwd** file would be unable to log in unless the current and archival files are reconciled.
- Never recover **/dev** files in place. If you do and then try to reboot, your system is likely to hang and you will be unable to reboot.

  Device files can be recovered in **/tmp**. You must then manually create any missing files, based on what is on the tape and recovered to **/tmp**.
- In another **/dev** recovery scenario, suppose your disk has crashed and you have no way to recover from your own system. A coworker might have a running system. Roll your disk over to your coworker's system and mount your disk in **/mysys**, owned by **root** and with permissions set to **drwx------**. Then you can **cd /mysys** and recover the **/dev** files in place. Retreive your disk and boot up.
- Auditing is not enabled automatically when you have recovered the system. Be sure to turn auditing on.

Guidelines for Mounting and Unmounting a File System

Mounting a file system can create security problems

- If not done carefully
- If the media being mounted contains compromising files
- In an NFS-configured computer environment

This section is intended to provide a security perspective on these tasks.

Note See the *mount*(1M) manual page in *HP-UX Reference* for detailed specifications.

The `/etc/mount`" command enables you to attach to an existing file tree

- Removable file systems
- Disks or disk partitions

The `mount` command uses a file called `/etc/mnttab`, which contains a list of available file systems and their permissions. The `/etc/mnttab` file should be writable only by root, but readable by others.

Observe the following precautions when mounting a file system or disk:

- Create a mount point directory (such as `/mnt`) on which to mount a new file system. Never mount a file system in the directory tree already containing files, because those files will become inaccessible.

  The mount point of a mounted file system acquires the permissions and ownership of the file system's root directory.

- Use base mode permissions and access control list entries on disk path names to control access to disks.
- Use the `-r` option of the `mount` command to maintain the write protection of tape drives and disks.

- When mounting a new or foreign file system, assume that the medium is insecure.
  - Create a directory restricted to root, by setting its permissions at 700 `drwx------`. Mount the foreign file system read-only at that location, for example, by loading the disk and typing:

    ```
    mount /dev/disk1 /securefile -r
    ```
  - Check all directories for special objects and privileged programs, and verify the identity of every program.
  - Run `ncheck -s` to scan for `setuid` and `setgid` programs and device files, and investigate any suspicious findings. Remove any unnecessary `setuid` and `setgid` permissions from files. These precautions are especially important if a user requests that you mount a personal file system.
  - Run the `fsck` program to verify that the file system is not technically corrupted.

  Only after performing these tests should you unmount the file system and remount it in its desired location.
- Be sure to unmount all mounted file systems of a user whose account you are disabling or removing.

For information on files mounted in an NFS environment, see Chapter 10.

Guidelines for Shutting Down a System Securely

Note This section gives a security overview of system shutdown. If you suspect that your system has been compromised, refer to "Shutdown due to Breach of Security," found in Chapter 5.

For basic instructions on shutting down a system, refer to the *System Administration Tasks* manual manual for your specific system, Chapter 3, "Starting and Stopping HP-UX".

`Shutdown` is used to halt the system in an orderly fashion for maintenance, installation, or powering down, without adversely affecting the file system. After a grace period, `shutdown`

- Kills all unnecessary processes
- Forces the contents of the file system's I/O buffers to be written to the disk (with the `sync` command)
- Places the system in single-user state (also known as system-administration mode, or run-level `s`)

`Shutdown` can also abruptly halt or reboot the system.

Since it is run only from the system console by a user logged in with root privileges, `shutdown` must be performed conscientiously to maintain system security.

Security Precautions

Observe the following security precautions when bringing down the system:

- Instruct users to log out before you start final shutdown procedures.
- When invoking the `shutdown` command, set a grace period to allow stragglers to log out and processes to complete.
- Always use `reboot` or `shutdown` to halt the CPU. If you simply pull the plug or push the reset button, all the processes halt and `syncer` cannot write the memory buffers onto disk.

- Never leave your system in the system-administration (**s**) run level any longer than necessary. **Shutdown** does not self-audit, and it turns auditing off.
- Do not physically write-protect a mounted file system, since this prevents **syncer** from updating the hard disk.
- Complete the **shutdown** before taking offline any disk drives or other peripherals. Do not take a disk offline without syncing and unmounting the file system on the disk.
- If the computer is halted and the last command involving output to the file system was not a **reboot** or **shutdown**, a superblock might be corrupted. The **fsck** program can be used to detect superblock inconsistency. See the *fsck*(1M) manual page in *HP-UX Reference*, Chapter 8, "HFS File System" in *How HP-UX Works: Concepts for the System Administrator*, and Chapter 6, "File System Problems" in *Solving HP-UX Problems*.

Guidelines for Handling Network Security Breaches

If a network security breach occurs due to unknown cause:

- Shut down the network and telephone access to the computer.
- Inform the network administrator immediately.
- Allow external access to the computer only after identifying and eliminating the problem.

Also, refer to "What to Do if You Discover a Security Breach," in Chapter 5 of this manual.

10

Trusted Networking

From the perspective of security, networked systems are more vulnerable than standalone systems. Networking increases system accessibility, but also greater risk of security violations.

An open system such as HP-UX allows extensive public access over networks to information in unencrypted form. While you cannot control security over the network, you can control security of each node on the network, to limit penetration risk without reducing the usefulness of the system or user productivity.

Following the principle of least privilege, all network management programs should be owned by a protected, network-specific account, such as **uucp** or **daemon**, rather than **root**.

This chapter provides information on:

- Controlling administrative domain
- Understanding network services
- Protecting passwords when using RFA
- Using **inetd.sec** to restrict outside access
- Denying access with **/etc/ftpusers**
- Mounting files in an NFS environment
- Safeguarding link-level access

Controlling Administrative Domain

An administrative domain is a group of systems connected by network services that allow users to access one another without password verification. An administrative domain assumes system users have already been verified by their host machine. In other words, network services assume security is established at the system level.

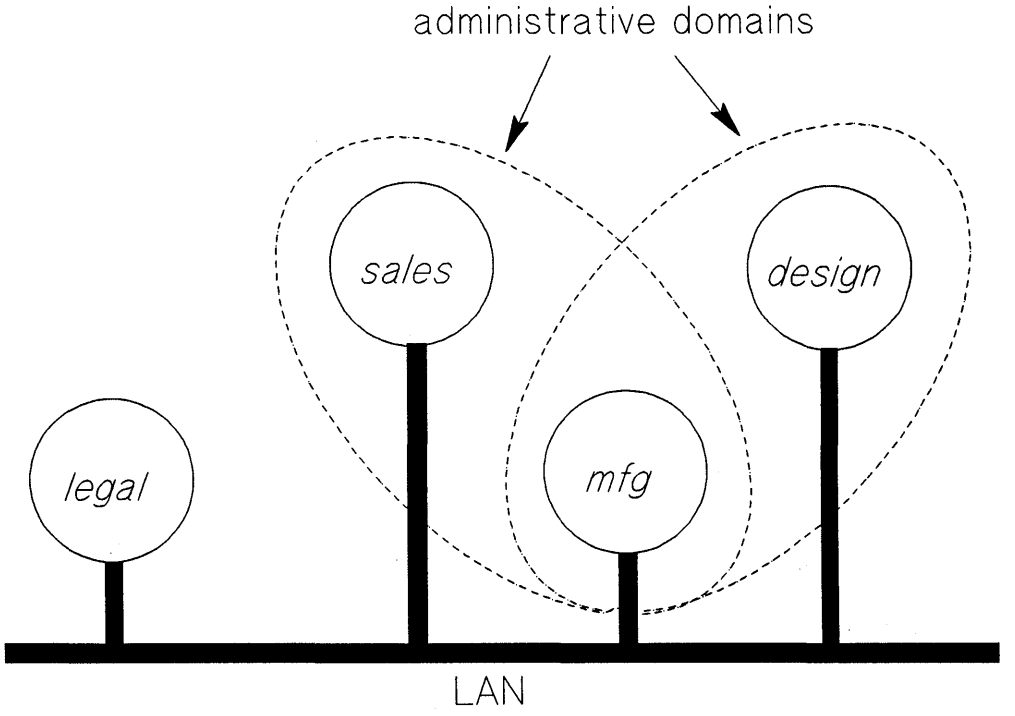


Figure 10-1. Examples of Administrative Domain

- A user on system **sales** need not enter a password to read an NFS-mounted file on system **mfg**, because **sales** verified the password when the user logged in. Systems **mfg** and **sales** are in the same administrative domain.
- A user on system **legal** executes **rlogin** to establish a remote session on system **sales**. System **legal** is not listed in **/etc/hosts.equiv** on **sales**, so

sales requires the user to provide a password. Systems **legal** and **sales** are not in the same administrative domain.

- Systems **design** and **sales** are not in the same administrative domain. However, because **design** and **mfg** are in the same domain and **mfg** and **sales** are in the same domain, by association With **mfg**, **design** and **sales** are indeed in the same administrative domain.

See Chapter 5 for techniques to identify and control administrative domain.

Understanding Network Services

HP-UX provides various networking services, each providing a means of authentication, either through password verification or authorization set up in a file on the remote system.

Network service	Access verification
netunam(1)	Password verification
dscopy(1)	Password verification
rcp(1)	Entry in **.rhosts** or **hosts.equiv** file
remsh(1)	Entry in **.rhosts** or **hosts.equiv** file
rlogin(1)	Password verification or entry in **.rhosts** or **hosts.equiv** file
telnet(1)	Password verification
ftp(1)	Password verification
mount(1M)	Entry in **/etc/exports**

For information on using the services, refer to the manual specific to the services. We have identified here some of the major security concerns related to these network services.

Protecting Passwords when using RFA

Potentially, any superuser can make a network special file anywhere in the file system, and use it to gain access to any other system on the network.

To prevent this, remote file access (RFA) requires you to respecify your password to the *netunam*(1) command. However, this command also allows you to specify your password on the command line. This poses a serious security risk. If you use **netunam** in a script, do not enter the password in the command line, as you will be doing so in an unencrypted state. It will also be seen by anyone who runs **ps** and thus gets broadcast!

Using inetd.sec to Restrict Outside Access

Except for RFA and NFT (network file transfer), access control to individual network services can be set in **/usr/adm/inetd.sec**, an optional security file for the Internet daemon. You can explicitly allow or deny use of most networking services by listing them on a per-machine or per-subnet basis.

The syntax of entries in **/usr/adm/inetd.sec** is:

```
<service name> <allow/deny> <host/net addresses, host/net names>
```

Service name is the official name (not alias) of a valid service in the file **/etc/services**. The service name for RPC-based services (NFS) is the official name (not alias) of a valid service in the file **/etc/rpc**. The wildcard character * and the range character - are permitted in addresses.

Refer to the manual entry *inetd.sec*(4) for complete details on the syntax and use of this file.

Denying Access with /etc/ftpusers

Ftpd(1M), the file transfer protocol server, is run by the Internet daemon (see *inetd*(1M)) when a service request is received at the port indicated in the `/etc/services`.

`Ftpd` rejects remote logins to local user accounts named in `/etc/ftpusers`. Each restricted account name must appear alone on a line in the file. The line cannot contain any spaces or tabs. User accounts with restricted login shells in `/etc/passwd` should be listed in `/etc/ftpusers`, because `ftpd` accesses local accounts without using their login shells. `Uucp` accounts also should be listed in `/etc/ftpusers`. If `/etc/ftpusers` does not exist, `ftpd` skips the security check.

Files Mounted in an NFS Environment

A Network File System (NFS) is used to

- Save file space
- Maintain consistent file usage
- Provide a lean cooperative user environment

NFS streamlines file-sharing between server and client systems by controlling access via the `/etc/exports` file. Entries in `/etc/exports` provide permission to mount a file system existing on the server onto any client machine. Once a file system is put into `/etc/exports`, the information is available to anyone who can do an NFS mount.

Thus, the NFS client user can access a server file system without having logged into the server system. (RFA and diskless clusters also provide access to files hooked up to a remote CPU, but do not bypass password authentication.

Server Vulnerability

Server security is maintained by setting restrictive permissions on the file `/etc/exports`. Root privileges are not maintained across NFS. Thus, having root privileges on a client system does not provide you with special access to the server.

The server performs the same permission checking remotely for the client as it does locally for its own users. The server side controls access to server files by the client by comparing the user ID and group ID of the client, which it receives via the network, with the user ID and group ID of the server file. Checking occurs within the kernel.

A user with privilege on an NFS client can exploit that privilege to obtain unlimited access to an NFS server. Never export any file system to a node on which privilege is granted more leniently than from your own node's policy!

Client Vulnerability

In earlier releases of NFS for workstations, the `/dev` inode had to reside on your (client) disk. NFS now allows for the `/dev` inode containing the major and minor numbers of a client-mounted device to exist on the server side. This opens the possibility for someone to create a Trojan Horse that overrides permissions set on the client's mounted device, by accessing the device via the file and inode number found on the server side.

Although lacking permission to make a device file on the client side, a cracker wanting to sabotage the client can create an undermining device file, such as `/dev/kmem`, using root permissions on server side. The new `/dev` file is created with the same major and minor number as that of the target device on client side, but with the following permissions:

```
crw-rw-rw-
```

The cracker can then go to client, log in as an ordinary user, and using NFS, can open up the newly created server-side device file and use it for devious means—to wipe out kernel memory on the server, read contents of everyone's processes, or other mischief.

How to Safeguard NFS-mounted Files

- If possible, make sure that the same person administers both client and server systems. Alternatively, require that administrators of both client and server systems coordinate their efforts.
- Maintain uniformity of user ID and group ID for server and client systems.
- Stay vigilent of `/dev` files in file systems exported from server.
- Restrict write access to the `/etc/passwd` and `/.secure/etc/passwd` client files.
- For strictest control, audit every host that is accessible through the network.

Link-Level Access

Link-level Access is a very powerful facility that permits a programmer to access the link driver on the host directly. In the wrong hands, this capability can enable an ordinary user to fabricate any network packet, including network control packets.

Everything transmitted on the system is enclosed in a packet, with a header and body. The header contains all administrative information, including control, routing, timing, and billing data. Sometimes this is the critical part of the transmission, rather than file content itself. Since HP-UX protocols each have their own administrative layers, information becomes more and more encapsulated the more protocols through which a packet travels.

To protect link-level access, make sure that the files `/dev/ether*`, `/dev/ieee*`, and `/dev/lan*` are owned and writable only by `root`.

Index

Special characters

/, 8-8

A

accept, 7-2
access control lists (ACLs), 4-1, 4-3, 6-5, 6-6, **8-22–38**, 9-1, 9-8, 9-10
  access check algorithm, 8-25
  ACL patterns, 8-33, 8-34
  functionality, 8-35
  long form, 8-26, **8-32**, 8-32
  network environment, 8-38
  notation, 8-26
  null entries, 8-30
  operator form, 8-26, **8-27**
  optional entries, 8-24
  patterns, 8-33
  short form, 8-26, **8-30**, 8-31
  syntax, 8-33
  uniqueness, 8-26
  wildcards, 8-33
access to software, 6-7
accountability, 6-6, 8-20
acltostr, 8-36
adm, 8-9
admin, 3-7
administrative domain, 5-7, 10-2
administrative events, 7-2
aid, 6-6
ar, 8-37
at, 2-4, 8-7, 8-18, 9-1
  at.allow, 8-18
  at.deny, 8-18
attacks, 9-3
audevent, 3-1, 7-2, 7-6
audisp, 3-1, 7-2, 7-6
auditable events, 7-2
audit data retention, 7-16
AUDIT fileset, 2-4
Audit File Switch (AFS) size, 2-8, 7-10
audit flag, 8-5
audit ID (**aid**), 2-4, 6-6, 7-1, 8-4, 8-5
auditing, 6-9, **7-1–18**, 9-11
  archiving audit files, 7-16
  auditable actions, 7-1, 7-2
  audit file, 2-8, 3-2
  audit log file, 3-11, 7-10
  audit log parameters, 2-8, 3-2
  audit monitor, 2-8, 3-2, 7-10
  auxiliary log file, 2-8, 3-2, 7-10
  commands, 3-1
  diskless environment, 7-18
  disk space, 3-13, 7-16
  events, 2-7, 3-7, 3-11, 7-2, 7-17
  file maintenance, 7-16
  guidelines, 7-17
  log file capacity, 2-8, 3-2
  log file pathname, 2-8, 3-2
  log file switch size, 2-8, 3-2
  parameters, 2-7, 3-2
  performance, 7-17
  primary log file, 2-8, 3-2, 7-10
  real-time environment, 7-17
  records, 3-11

setting parameters, 2-7, 3-1
streamling log entries, 7-5
system calls, 2-7, 3-9, 3-11, 7-2
tasks, **3-1–13**
terminals, 3-11
time intervals, 3-11
user-defined events, 7-5
users, 2-7, 3-5, 3-11, 7-17
viewing audit logs, 3-11
warning messages, 7-10
auditing records, 7-10
auditing tasks, **3-1–13**
audit log
defining, 2-8, 3-2
parameters, 3-2
size and location, 2-8, 3-2
viewing, 3-11
audit log data, reducing amount, 7-5
audit log parameters, 3-2
audit monitor, 3-2
audit record, 3-11, 7-1, 7-9
audit schedule, 8-13
audomon, 3-1, 3-2
audswitch, 7-2
audsys, 3-1, 7-2, 7-6
audusr, 3-1, 7-2, 7-6
authentication, 6-3, 6-9, 7-2, 7-9, 8-1
authorization, 6-4, 6-9
auxiliary log file, 2-8, 3-2, 7-10
awk, 5-8

B

backup, 2-3, 6-5, **9-8–9**
automating, 9-9
scheduling, 9-8
backup media, 9-8
cartridge tape, 9-9
reel tape, 9-9
batch, 2-4
bin, 8-9, 8-10
binaries, 8-10
bind, 7-2
block special files, 5-11
byte count, 5-12

C

cat, 5-7
centralized security responsibilities, 6-4
chacl, 4-1, 4-3, 4-4, 8-26, 8-29, 8-31, 8-34, 8-35
character special files, 5-11
chdir, 7-2
chfn, 7-2, 7-6, 8-4
chgrp, 8-19, 8-20
chmod, 4-1, 4-4, 4-5, 7-2, 8-12, 8-22, 8-24, 8-26, 8-36, 9-11
chown, 7-2, 8-4, 8-18, 8-19, 8-36, 9-10
chownacl, 8-36
chroot, 7-2
chsh, 7-2, 7-6
close, 7-2
cluster, 7-2
compact, 8-36
compress, 8-36
confidentiality, need for, 8-20
configuration files, 6-6
configuration parameters, 6-10
connect, 7-2
consistency, 5-8
console logs, 6-3
contaminated files, 5-12
continuation inodes, 8-37
controlling administrative domain, 10-2
controlling file access, 4-1
converting to a secure (trusted) system, **2-4–6**
corporate culture, 6-5
cpacl, 8-36
cpio, 8-37
cpset, 8-36
creat, 7-2
cron, 8-7, 8-18, 9-1

cron.allow, 8-18
cron.deny, 8-18
crontab, 2-4, 9-7
.cshrc, 8-4

D

daemon, 8-9
data loss, 9-10
data recovery, 9-10
Department of Defense
Trusted Computer System Evaluation Criteria, 1-1, 6-5
detecting irregularities, 5-1
/dev, 8-8, 9-11
device special files, 8-11
ownership, 8-11
permissions, 8-11
/dev/tty, 9-11
dialback, 8-13
dial-in accounts, 8-13
diff, 5-8
directories, 5-11
dynamic, 8-10
static, 8-10
directory access, 4-1, 4-3
disabling an account, 8-17
discretionary access control (DAC), **6-5**, 6-9, 7-2, 8-22
disk partitions, 9-12
protecting, 8-12
disks, 9-12
disk space, 2-9
disk special files, 8-11
diskusg, 8-37
disseminating security information, 6-4, 8-14
dump, 8-37
dump analysis, 6-10

E

ed, 8-36
education, 6-4
effective user ID (euid), 8-7, 9-2
enable auditing, **2-7**, 2-10
encrypted password field, 8-4, 8-5
encryption, password, 8-3
environment variables, 9-1
/etc, 5-10, 8-8
/etc/auditrc, 2-5
/etc/exports, 5-7
/etc/filesets, 2-4
/etc/ftpusers, protecting accounts with, 10-5
/etc/group, 5-9, 8-4, 8-5, 8-14, 8-18, 8-19, 8-20
/etc/hosts.equiv, 10-2
/etc/inittab, 8-8
/etc/logingroup, 8-20
/etc/mount, 8-17, 9-12
/etc/netgroup, 5-7
/etc/newconfig/auditrc, 2-5
/etc/passwd, 5-8, 8-1, 8-3, 8-4, 8-5, 8-8, 8-14, 8-17, 8-18, 8-20, 8-24, 9-11
/etc/rc, 8-8
/etc/umount, 8-17
/etc/utmp, 8-10
/etc/wtmp, 8-10
event failure, 7-9
events, auditable, 7-2
event success, 7-9
execv, 7-2
execve, 7-2
exit, 7-2
external access, 8-13

F

failed access, 7-17
failure, event, 7-9
fbackup, 8-37, 9-8

fchmod, 7-2
fchown, 7-2
fcpacl, 8-36
fgetacl, 8-35
file access, controlling, 4-1, 8-22, 8-38
file permissions, 4-1, 4-2, 4-3, 4-5, **8-7–13**, 8-22, 9-1, 9-10
  listing, 4-2
  modes, 9-3
files
  byte count, 5-12
files and directories, system, **8-8**
fileset, **AUDIT**, 2-4
File Space Switch (FSS) size, 2-8, 7-10
file systems, 5-7
  exporting, 5-7
  mounting, **9-12–13**
  unmounting, 9-12
find, 5-1, 5-12, 8-18, 8-19, 8-20, 8-34, 8-36
fork, 7-2
frecover, 8-37, 9-8, 9-10, 9-11
fsck, 8-37, 9-12
fsetacl, 7-2, 8-35
fsetaclentry, 8-36
ftio, 8-37
ftruncate, 7-2

G

games, 6-8
getaccess, 8-35
getacl, 8-35
getitimer, 9-1
getpwent, 8-6
getspwent, 8-6
  get secure password entries, 8-6
getty, 6-5
grep, 5-10, 8-17
group, 8-22
group ID (**gid**), 8-4, 8-5, 8-19, 8-20, 8-22, 8-24
groups, 6-9
  adding a group, 8-20
  criteria, 8-20
  modifying a group, 8-21
  removing a group, 8-21

H

hidden files, 5-1
hosts.equiv, 5-10
hpterm, 9-7
/hp-ux, 8-8
HP-UX operating system
  protections, **9-5**
  risks, **9-1**
HP-UX Reference, 8-17, 8-22, 8-36, 9-1, 9-10, 9-11
human factors, 6-5

I

identification, 6-3, 7-2, 7-9
IFS variables, 9-1
inetd.sec, 6-6, 10-4
  using to restrict outside access, 10-4
init, 7-2, 7-6
initgroups, 8-20
initialization, system, 9-1
inode, 8-12
insf, 8-11
install, 2-4
install due to security breach, 5-13
install HP-UX, 2-1, 2-3
integrity, system, 6-1
inter-process communication (IPC) objects, 7-4, 8-22
ipcclose, 7-4
ipcconnect, 7-2
ipccreat, 7-2
ipccreate, 7-2
ipc datagram, 7-2
ipc datagram transactions, 7-4
ipcdest, 7-2

ipcdgram, 7-4
ipclookup, 7-2
ipcopen, 7-2
ipcrecvcn, 7-2
ipcshutdown, 7-2
isolating objects, 6-5

K

kernel, 6-5
kill, 7-2
kmem, 8-11

L

least privilege, 6-6, 8-7, 8-10, 8-14
liability, 6-1
link, 7-2
listing file permissions, 4-2
load average, 5-12
log entries, streamlining, 7-5
log files, 6-6
.login, 8-7
/login, 6-3
login, 3-7, 3-11, 6-5, 7-2, 7-6, 7-17, 8-4, 8-11, 8-20
login name, 8-5
login restrictions, 6-7
logout, 7-2
loss of data, 9-10
/lost+found, 9-10
lp, 8-9
lpsched, 7-2, 7-6
ls, 5-7, 5-12, 8-36
lsacl, 4-1, 4-2, 4-3, 4-5, 8-26, 8-35

M

mail account, 8-18
mailx, 8-36
maintaining a secure system, 2-11
makecdf, 8-36
mem, 8-11
memory, 8-11
mkdir, 7-2
mkfs, 8-37
mknod, 7-2, 8-36
/mnt, 9-12
modaccess, 7-2
moddac, 3-7, 7-2
mode, 8-22, 8-26, 8-30, 8-33
modem, system access by, 8-13
mount point, 9-12
msgctl, 7-2
msgget, 7-2

N

National Computer Security Center, 6-9
ncheck, 5-12, 8-37, 9-12
need to know, 8-10
netunam, 10-4
network, 9-15
network access, 8-13
network administrator, 8-13
network control files, permissions on, 5-10
network environment, ACLs in a, 8-38
Network File Systems (NFS), 8-38
networking, trusted, **10-1–7**
networks, 8-38
network services, 10-2
network special files, 5-7
newgrp, 7-2, 7-6, 8-20, 9-7
/nfs, 5-8
NFS, 8-38, 10-2
NFS environment, 10-5
  security concerns, 10-5
nodes, 5-7
notation, access control list (ACL), 8-26
null access control lists (ACLs), 8-30

O

objects, 6-6, 7-1, 8-22
  close, 7-2

creation, 7-2
deletion, 7-2
introduction or deletion, 7-5
isolating, 6-5
open, 7-2
offsite storage, 9-8
open, 7-2, 7-5
open system, 6-7
operating system, 6-5
protection, 5-1, 9-1
risks, 9-1
operator, 6-9, 6-10
OR, 8-25
overflow, 7-17

P

pack, 8-36
parameters, auditing, 2-7, 3-2
passwd, 7-2, 7-6
password, 5-12
aging, 8-1, 8-4, 8-19
criteria, 8-3
database, 8-1
encrypted field, 8-4, 8-5
encryption, 8-3
integrity, 8-2
RFA, 10-4
security, **8-1–6**
unencrypted, 10-4
unexpected queries, 6-8
password entries, get, 8-6
password file, 6-6, 8-17
fields, 8-4
password system, 6-9
PATH, 6-7, 8-7, 9-1
penetration, 6-7
penetrator, 9-3
permission bits, 8-12, 8-22, 8-25
criteria for, 8-10
permissions, file, 4-2, 4-3, 8-22
modes, 9-3
overly liberal, 5-1
read, 5-12
write, 5-12
pipe, 7-2
power failure, 9-10
primary log file, 2-8, 3-2, 7-10
printf, 5-8
printouts, 6-3
privgrp, 7-2
privileged programs, 6-5, 8-9, 9-2, 9-12
privileges, 8-7
process, 7-2
.profile, 8-4, 8-7, 8-14
programmer, 6-10
programming guidelines, 9-5
protecting accounts with **/etc/ftpusers**, 10-5
protecting system files, 5-1
protecting your files, 4-1
protecting your operating system, 5-1
protocols, 5-10
ps, 8-17
pseudo-accounts, 8-9
ptrace, 7-2
public directories, 6-6
public files, 6-9
putpwent, 8-6
put password entries, 8-6
putspwent, 8-6
put secure password entries, 8-6
pwck, 7-2, 7-6

R

RCS, 8-37
read permissions, 5-12
reboot, 5-12, 7-2
record, audit, 7-1
recovery, **9-10–11**
recovery of data, 9-10
recovery system, 9-10
relative pathnames, 9-9

Remote File Access (RFA), 8-38
removable media events, 7-2
rename, 7-2
restore, 6-5, 8-37
restricting access, 10-4
RFA, 8-38, 10-4
rfa_netunam, 7-2
.rhosts, 6-6
risks, 9-1
rlogin, 10-2
rm, 8-18
rmdir, 7-2
root, 5-1, 5-10, **8-7–10**, 8-12, 9-2
root directory, 6-6
root privilege, 8-7
root privilege, protecting, 5-8

S

sam, 6-5, 7-2, 7-6
SAM
  invoking, 1-4
sam, System Administration Manager, 1-4, 3-1, 7-2, 7-6
SCCS, 8-37
/.secure/etc/auditlog, 2-8, 3-2
/.secure/etc/passwd, 2-4, 6-3, 8-1, 8-3, 8-4, 8-5, 8-8, 8-20
secure password entries, get, 8-6
secure password file, 2-4
secure programs, 6-10
secure system
  maintaining, 2-1, 2-11
  setting up, 2-1
security breach, 5-1, 5-12, 8-9, 8-10, 8-13, 9-3
  external access, 9-15
  load average, 5-12
  network, 9-15
  recognizing, 5-12
  telephone access, 9-15
  what to do, 5-12
security guidelines, 6-4
security personnel, 6-9–10
security policy, 6-1, 6-5
  hardware, 6-2
  limiting physical access, 6-2
  perimeter controls, 6-3
  physical security, 6-2
  procedural security, 6-3
  sabotage, 6-2
  storing backup tapes, 6-3
  system security, 6-3
  user authentication, 6-3
  user authorization, 6-4
  user identification, 6-3
security tools, 5-1
security violation, 8-3
selectable events, 7-17
semctl, 7-2
semget, 7-2
sensitive files, 6-6
setacl, 7-2, 8-24, 8-35
setaclentry, 8-36
setaudid, 7-2
setaudproc, 7-2
setdomainname, 7-2
setevent, 7-2
setgid, 7-2
setgid programs, 6-9, 8-10, 8-20, 9-2, 9-3, 9-10, 9-13
setgroups, 7-2
sethostid, 7-2
setresgid, 7-2
setresuid, 7-2
settimeofday, 7-2
setting up a secure system, 2-1
setuid, 7-2
setuid programs, 6-9, 8-9, 8-10, 8-20, 9-1–5, 9-10, 9-13
  writable, 5-1
shar, 8-37
shmat, 7-2

shmctl, 7-2
shmdt, 7-2
shmget, 7-2
shredding, 6-3
shutdown, 5-12, 7-2, **9-14–15**
single-user state, 5-12, 8-17, 8-18, 9-14
smount, 7-2
socket, 7-2
special files
  block, 5-11
  character, 5-11
spoofing, 6-7
stat, 8-37
stime, 7-2
st_mode, 8-37
streamlining audit log entries, **7-5**
strtoacl, 8-36
strtoaclpatt, 8-36
subjects, 6-5
subsystem files, 8-9
subsystems, protecting key, 8-9
success, event, 7-9
superuser, 5-10, 6-9, 8-7
suspend/resume auditing, 7-5
su (substitute user), 6-9, 8-7, 9-3, 9-7
swap, 8-11
swap files, 8-11
swapon, 7-2
sync, 9-14
System Administration Manager (SAM), **1-4**, 3-1, 8-7, 8-14, 8-20, 8-21, 9-8
System Administration Tasks manual, 1-1
system administrator, 6-9, 8-7
System Administrator Manager (SAM), 8-21
system aliases, 8-18
system crash, 6-1
system files
  protecting, 5-1
system files and directories, **8-8**
system maintenance, 6-10
system penetration, 8-13
system programmer, 6-9
system security, 6-3, 8-7, 8-12
  enforcing, 6-5
system security, enforcing, 8-9
system security officer, 6-9, 8-1
system tape, 5-12

T

talk, 8-11
tampering, 9-3, 9-9
tar, 8-37
tasks, auditing, **3-1–13**
telephone access, 8-13
terminal port, 8-11
time bombs, 6-7
/tmp, 9-11
trap, 8-14
trap door, 6-7
Trojan Horses, 5-12, 6-1, **6-7–8**, 8-10
truncate, 7-2
Trusted Computing Base (TCB), 6-5, 8-9, 9-1
trusted networking, **10-1–7**
trusted system, 1-1
  converting to, **2-4–6**
turning auditing on/off, 3-4

U

udp datagram, 7-2
uevent, 7-2
uevent1, 7-4
uevent2, 7-4
ulimit, 8-4, 8-14
umask, 6-6, 7-2, 8-4, 8-7
umount, 7-2
uniqueness, ACL, 8-26
unlink, 7-2
unpack, 8-36
update, 2-4

U.S. Computer Security Act of 1987, 6-1
user access, 8-1
user accounts, 6-9, 8-7
  adding an account, 8-14
  archiving a user's files, 8-18
  deactivating an account, 8-17
  guest accounts, 8-14
  mailbox, 8-18
  managing an account, **8-14–21**
  moving a user account, 8-19
  other systems, 8-18
  pseudo-accounts, 8-9
  reactivating accounts, 8-17
  removing an account, 8-18
  temporary accounts, 8-14
user ID (`uid`), 8-4, 8-5, 8-9, 8-19, 8-20, 8-22, 8-24
`/usr`, 5-12, 8-8
`/usr/adm/inetd.sec`, 10-4
  syntax, 10-4
`/usr/bin`, 8-7, 8-8
`/usr/lib`, 8-8
`/usr/mail`, 8-19
`/usr/spool`, 8-8
`/usr/spool/cron/crontabs`, 8-18, 8-19
`/usr/tmp`, 8-14
`uucp`, 8-9

V

`vfork`, 7-2
`vfsmount`, 7-2
viewing audit logs, 3-11
`vipw`, 8-4, 8-14
viruses, 6-1, 6-8

W

warning messages, 7-10
`who`, 8-17
worms, 6-8
`write`, 8-11, 8-17
write permissions, 5-12

X

`xterm`, 9-7

Reorder No. or
Manual Part No.
B2355-90045

**Manufacturing
Part No.
B2355-90045**

B2355-90045