HP-UX System Administrator's Guide: Security Management

HP-UX 11i Version 3

HP Part Number: B3921-90059
Published: September 2011
Edition: 7

Acknowledgments This product includes software developed by the Apache Software Foundation. This documentation is based on information from the Apache Software Foundation (`http://www.apache.org`).

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (`http://www.openssl.org`).

Table of Contents

About this Document....................15

I Protecting Systems....................21

1 Installing the HP-UX Operating Environment Securely....................23
- 1.1 Installation Security Considerations....................23
- 1.2 Preventing Security Breaches During the Boot Process....................23
- 1.3 Enable Login Security for root....................24
- 1.4 Using Boot Authentication to Prevent Unauthorized Access....................25
- 1.5 Setting Install-Time Security Options....................25
- 1.6 Installing Security Patches....................26
- 1.7 Postinstallation Security Tips for Backup and Recovery....................26

2 Administering User and System Security....................29
- 2.1 Managing User Access....................29
  - 2.1.1 Monitoring User Accounts....................29
  - 2.1.2 Monitoring Guest Accounts....................30
  - 2.1.3 Creating Application User Accounts....................30
  - 2.1.4 Managing Group Accounts....................31
- 2.2 Authenticating Users During Login....................31
  - 2.2.1 Explanation of the Login Process....................32
  - 2.2.2 Checking the login Tracking Files (btmp and wtmp)....................33
    - 2.2.2.1 Last Command Examples....................33
  - 2.2.3 Checking Who Is Logged In....................34
- 2.3 Authenticating Users with PAM....................34
  - 2.3.1 Overview....................34
  - 2.3.2 PAM Libraries....................36
  - 2.3.3 Systemwide Configuration Using /etc/pam.conf....................37
  - 2.3.4 Sample /etc/pam.conf File....................38
  - 2.3.5 The /etc/pam_user.conf User Configuration File....................39
  - 2.3.6 Examples: How PAM Works for Login....................39
- 2.4 Managing Passwords....................41
  - 2.4.1 System Administrator Responsibilities....................41
  - 2.4.2 User Responsibilities....................41
  - 2.4.3 Criteria of a Good Password....................42
  - 2.4.4 Changing the /etc/passwd Password File....................42
    - 2.4.4.1 Examples of passwd Commands....................42
    - 2.4.4.2 The /etc/passwd File Format....................43
  - 2.4.5 The /etc/shadow Shadow Password File....................43

2.4.6 Eliminating Pseudo-Accounts and Protecting Key Subsystems in /etc/passwd....45
2.4.7 Secure Login with HP-UX Secure Shell....46
2.4.8 Securing Passwords Stored in NIS....46
2.4.9 Securing Passwords Stored in LDAP Directory Server....46
2.5 Defining System Security Attributes....46
2.5.1 Configuring Systemwide Attributes....48
2.5.2 Configuring Per-User Attributes....48
2.5.2.1 Examples of Defining User-Specific Attributes with userdbset....49
2.5.2.2 INACTIVITY_MAXDAYS and the Shadow Password File....49
2.5.3 Troubleshooting the User Database....49
2.6 Handling setuid and setgid Programs....50
2.6.1 Why setuid and setgid Programs Can Be Risky....51
2.6.2 How IDs Are Set....51
2.6.3 Guidelines for Limiting Setuid Power....51
2.7 Preventing Stack Buffer Overflow Attacks....52
2.8 Protecting Unattended Terminals and Workstations....53
2.8.1 Controlling Access Using /etc/inittab and Run Levels....53
2.8.2 Protecting Terminal Device Files....54
2.8.3 Configuring the Screen Lock....54
2.8.3.1 Configuring the TMOUT Variable....54
2.8.3.2 Configuring the CDE Lock Manager....55
2.9 Protecting Against System Access by Remote Devices....55
2.9.1 Controlling Access Using /etc/dialups and /etc/d_passwd....56
2.10 Securing Login Banners....57
2.11 Protecting the root Account....58
2.11.1 Monitoring root Account Access....58
2.11.2 Using the Restricted SMH Builder for Limited Superuser Access....58
2.11.3 Reviewing Superuser Access....59

3 HP-UX Standard Mode Security Extensions....61
3.1 Overview....61
3.2 Security Attributes and the User Database....62
3.2.1 System Security Attributes....62
3.2.2 Configuring Systemwide Attributes....62
3.2.3 User Database Components....63
3.2.3.1 Configuration Files....63
3.2.3.2 Commands....63
3.2.3.3 Attributes....63
3.2.3.4 Manpages....64
3.2.4 Configuring Attributes in the User Database....65
3.2.5 Troubleshooting the User Database....65

4 Remote Access Security Administration....67
4.1 Overview of Internet Services and Remote Access Services....67
4.1.1 Securing ftp....68
4.1.2 Securing Anonymous ftp....69
4.1.3 Denying Access Using /etc/ftpd/ftpusers....69
4.1.4 Other Security Solutions for Spoofing....70
4.2 The inetd Daemon....70
4.2.1 Securing inetd....71
4.2.1.1 Denying or Allowing Access Using /var/adm/inetd.sec....72
4.3 Protection Against Spoofing with TCP Wrappers....72
4.3.1 Additional Features of TCP Wrappers....73
4.3.2 TCP Wrappers Do Not Work with RPC Services....73
4.4 Secure Internet Services....73
4.5 Controlling an Administrative Domain....74
4.5.1 Verifying Permission Settings on Network Control Files....75
4.6 Securing Remote Sessions Using HP-UX Secure Shell (SSH)....76
4.6.1 Key Security Features of HP-UX Secure Shell....76
4.6.2 Software Components of HP-UX Secure Shell....77
4.6.3 Running HP-UX Secure Shell....78
4.6.3.1 Running the ssh Client....78
4.6.3.2 Running the sftp Client....79
4.6.3.3 Running the scp Client....79
4.6.4 HP-UX Secure Shell Privilege Separation....79
4.6.5 HP-UX Secure Shell Authentication....80
4.6.5.1 GSS-API....80
4.6.5.2 Public Key Authentication....81
4.6.5.3 Host-Based and Public Key Authentication....81
4.6.5.4 Password Authentication....81
4.6.6 Communication Protocols....82
4.6.7 HP-UX Secure Shell and the HP-UX System....82
4.6.8 Associated Technologies....83
4.6.9 Strong Random Number Generator Requirement....83
4.6.10 TCP Wrappers Support....83
4.6.11 chroot Directory Jail....84

II Protecting Data....85

5 File System Security....87
5.1 Controlling File Access....87
5.1.1 Setting File Access Permissions....88
5.1.2 Setting File Ownership....89
5.1.3 Protecting Directories....89
5.1.4 Protecting Files Related to User Accounts....90

5.1.5 Locating and Correcting File Corruption Using fsck....90
5.2 Setting Access Control Lists....91
5.3 Using HFS ACLs....91
5.3.1 HFS ACLs and HP-UX Commands and Calls....93
5.4 Using JFS ACLs....95
5.4.1 Definition of a JFS ACL....95
5.4.2 How the System Generates a JFS ACL....95
5.4.3 Minimal JFS ACL....96
5.4.4 Additional JFS ACL user and group Entries....96
5.4.5 JFS ACL group and class Entries....96
5.4.6 Using the setacl and getacl Commands....97
5.4.7 Effect of chmod on class Entries....97
5.4.8 Example of Changing a Minimal JFS ACL....98
5.4.9 Default JFS ACLs....99
5.4.10 Changing JFS ACL with the setacl Command....100
5.4.10.1 Using the Modify and Delete Options....100
5.4.10.2 Using the -f Option....100
5.4.10.3 Effective Permissions and setacl -n....101
5.5 Comparison of JFS and HFS ACLs....102
5.5.1 JFS and HFS Command and Function Mapping....102
5.6 ACLs and NFS....103
5.7 Security Considerations for /dev Device Special Files....103
5.8 Protecting Disk Partitions and Logical Volumes....104
5.9 Security Guidelines for Mounting and Unmounting File Systems....104
5.10 Controlling File Security on a Network....106
5.10.1 Check Permission Settings on Network Control Files....106
5.10.2 Files Mounted in an NFS Environment....106
5.10.2.1 Server Vulnerability....107
5.10.2.2 Client Vulnerability....107
5.10.2.3 How to Safeguard NFS-Mounted Files....107

6 Compartments....109
6.1 Overview....109
6.1.1 Compartment Architecture....109
6.1.2 Default Compartment Configuration....111
6.2 Planning the Compartment Structure....111
6.3 Compartment Components....112
6.3.1 Compartment Configuration Files....112
6.3.2 Compartment Commands....112
6.3.3 Compartment Manpages....113
6.4 Compartment Rules and Syntax....114
6.4.1 Compartment Definition....114
6.4.2 File System Rules....116

6.4.3 IPC Rules....117
6.4.4 Network Rules....119
6.4.5 Miscellaneous Rules....122
6.4.6 Example Rules File....123
6.5 Configuring Compartments....124
6.5.1 Activating Compartments....124
6.5.2 Defining a Compartment Configuration....124
6.5.2.1 Changing Compartment Rules....125
6.5.2.2 Changing Compartment Names....125
6.5.3 Running an Application in a Compartment....125
6.5.4 Login Directly to a Compartment....126
6.6 Troubleshooting Compartments....126
6.7 Using Discover Mode to Generate Initial Compartment Configuration....127
6.8 Compartments in HP Serviceguard Clusters....128

7 Fine-Grained Privileges....131
7.1 Overview....131
7.2 Fine-Grained Privileges Components....131
7.2.1 Commands....131
7.2.2 Manpages....132
7.3 Available Privileges....132
7.3.1 Compatibility Information for Divided Privileges....135
7.4 Configuring Applications with Fine-Grained Privileges....136
7.4.1 Privilege Model....138
7.4.2 Compound Privileges....138
7.5 Security Implications of Fine-Grained Privileges....139
7.5.1 Privilege Escalation....139
7.6 Fine-Grained Privileges in HP Serviceguard Clusters....139
7.7 Troubleshooting Fine-Grained Privileges....140

III Protecting Identity....141

8 HP-UX Role-Based Access Control....143
8.1 Overview....143
8.2 Access Control Basics....144
8.2.1 Simplifying Access Control with Roles....145
8.3 HP-UX RBAC Components....146
8.3.1 HP-UX RBAC Access Control Policy Switch....147
8.3.2 HP-UX RBAC Configuration Files....147
8.3.3 HP-UX RBAC Commands....148
8.3.4 HP-UX RBAC Manpages....148
8.3.5 HP-UX RBAC Architecture....149
8.3.6 HP-UX RBAC Example Usage and Operation....150

8.4 Planning the HP-UX RBAC Deployment....151
8.4.1 Planning the Roles....152
8.4.2 Planning Authorizations for the Roles....152
8.4.3 Planning Command Mappings....153
8.4.4 HP-UX RBAC Limitations and Restrictions....153
8.5 Configuring HP-UX RBAC....154
8.5.1 Configuring Roles....155
8.5.1.1 Creating Roles....155
8.5.1.2 Assigning Roles to Users....156
8.5.1.3 Assigning Roles to Groups....157
8.5.2 Configuring Authorizations....157
8.5.3 Configuring Additional Command Authorizations and Privileges....158
8.5.4 Configuring HP-UX RBAC with Fine-Grained Privileges....160
8.5.5 Configuring HP-UX RBAC with Compartments....162
8.6 Using HP-UX RBAC....163
8.6.1 Using the privrun Command to Run Applications with Privileges....163
8.6.1.1 HP-UX RBAC in Serviceguard Clusters....165
8.6.2 Using the privedit Command to Edit Files Under Access Control....165
8.6.3 Customizing privrun and privedit Using the ACPS....166
8.6.4 Generating Keystroke and Command Logs....167
8.6.4.1 Keystroke Logging....167
8.6.4.2 Alternate Logging....168
8.7 Troubleshooting HP-UX RBAC....168
8.7.1 The rbacdbchk Database Syntax Tool....168
8.7.2 privrun -v Information....169

9 Audit Administration....171
9.1 Auditing Components....172
9.1.1 Commands....172
9.1.2 Audit Configuration Files....172
9.1.3 Audit Manpages....173
9.2 Auditing Your System....173
9.2.1 Planning the Auditing Implementation....173
9.2.2 Enabling Auditing....174
9.2.3 Disabling Auditing....175
9.2.4 Monitoring Audit Files....175
9.2.5 Performance Considerations....176
9.2.6 Guidelines for Administering the Auditing System....176
9.3 Auditing Users....176
9.4 Auditing Events....177
9.5 Audit Trails....179
9.5.1 Configuring Audit Trails....180
9.5.2 Monitoring and Managing Audit Trails....181

9.6 Using the Audit Filtering Tools....182
9.7 Using filter.conf....183
9.8 Using the Audit Reporting Tools....183
9.8.1 Examples of Using the auditdp Command....185
9.9 Viewing Audit Logs....186
9.9.1 Examples of Using the audisp Command....187
9.10 Self-Auditing....187
9.11 HP-UX RBAC Auditing....188
9.11.1 Auditing Based on HP-UX RBAC Criteria and the /etc/rbac/aud_filter File....188
9.11.2 Procedure for Auditing HP-UX RBAC Criteria....189

A Trusted Systems....191
A.1 Setting Up a Trusted System....191
A.2 Auditing a Trusted System....192
A.3 Managing Trusted Passwords and System Access....192
A.3.1 Password Files....193
A.3.1.1 The /etc/passwd File....193
A.3.1.2 The /tcb/files/auth/ Database....194
A.3.2 Password Selection and Generation....195
A.3.3 Password Aging....195
A.3.4 Password History and Password Reuse....196
A.3.5 Time-Based Access Control....196
A.3.6 Device-Based Access Control....196
A.3.7 Manipulating the Trusted System Databases....197
A.4 Guidelines for Trusted Backup and Recovery....197

B Other Security Products....199
B.1 Protecting Systems....199
B.1.1 HP-UX Bastille....199
B.1.2 HP-UX HIDS....199
B.1.3 HP-UX IPFilter....200
B.1.4 Security Patches....200
B.2 Protecting Data....200
B.2.1 HP-UX Containers (SRP)....200
B.2.2 HP-UX Encrypted Volume and File System (EVFS)....201
B.2.3 HP-UX IPSec....201
B.2.4 HP-UX OpenSSL....201
B.2.5 HP-UX Secure Shell....202
B.2.6 HP-UX Trusted Computing Services....202
B.2.7 HP-UX Whitelisting....203
B.3 Protecting Identity....203
B.3.1 HP-UX AAA Server (RADIUS)....203

B.3.2 HP-UX Directory Server....204
B.3.3 HP-UX LDAP-UX Integration....204

Glossary....205

Index....213

List of Figures

2-1 HP-UX Authentication Modules Under PAM..35
5-1 File and Directory Permission Fields...88
6-1 Compartment Architecture...110
8-1 HP-UX RBAC Architecture..150
8-2 Example Operation After Invoking privrun...151

List of Tables

3-1 User Database Configuration Files....63
3-2 User Database Commands....63
3-3 User Attributes....64
3-4 User Database Manpages....64
4-1 Internet Services Components and Access Verification, Authorization, and Authentication....67
4-2 Software Components of HP-UX Secure Shell....77
5-1 Differences Between File and Directory Privileges....88
5-2 HFS ACL Commands....94
5-3 HFS ACL System Calls....94
5-4 Commands and Calls Affecting ACL Entries....94
5-5 HFS and JFS ACL Equivalents....102
6-1 Compartment Configuration Files....112
6-2 Compartment Commands....113
6-3 Compartment Manpages....113
7-1 Fine-Grained Privileges Commands....132
7-2 Fine-Grained Privileges Manpages....132
7-3 Available Privileges....132
8-1 Example of Authorizations Per User....144
8-2 Example of Authorizations Per Role....145
8-3 HP-UX RBAC Configuration Files....147
8-4 HP-UX RBAC Commands....148
8-5 HP-UX RBAC Manpages....148
8-6 Example Planning Results....155
9-1 Audit Commands....172
9-2 Audit Configuration Files....172
9-3 Audit Manpages....173
9-4 audevent Command Options....178

List of Examples

2-1 Pseudo- and Special System Accounts....45
5-1 Creating an HFS ACL....93
5-2 Multiple HFS ACL Matches....93

About this Document

Publication History

The document publication date and part number indicate its current edition. The publication date will change when a new edition is released.

To ensure that you receive the new editions, you should subscribe to the appropriate product support service. Contact your HP sales representative for details.

You can find the various versions of this document at:

http://www.hp.com/go/hpux-core-docs

Click **HP-UX 11i v3**.

September 2011 — Part Number B3921–90059

- Updated the Compartment chapter (see Chapter 6).
- Updated the Fine-Grained Privileges chapter (see Chapter 7.
- Reorganized Appendix B in three parts: *Protecting Systems*, *Protecting Data*, and *Protecting Identity* and added the HP-UX OpenSSL and HP-UX Whitelisting security products (see Appendix B).

September 2010 — Part Number B3921-90020

- Removed the Bastille chapter since the Bastille product now has its own user guide. Added the Bastille product in Appendix B (page 199).
- Added the HP-UX Directory Server product in Appendix B (page 199).
- Added the HP-UX LDAP product in Appendix B (page 199).
- Updated all links to `docs.hp.com` to the Business Support Center. The HP-UX documentation is now located at the Business Support Center. For the HP-UX security collection, see http://www.hp.com/go/hpux-security-docs.

September 2009 — Part Number 5992–6416

- Added the HP-UX PAM RADIUS module to the PAM Libraries section (see Section 2.3.2).
- Added a new section in the Bastille chapter, *Selecting Install-Time Security*.

  This section used to be documented in the *HP-UX 11i v3 Installation and Update Guide*.
- Updated the Compartment chapter (see Chapter 6).

	• Updated the HP-UX Role-Based Access Control chapter (see Chapter 8). • Updated the Audit Administration chapter (see Chapter 9). • Added security products to Appendix B (see Appendix B).
March 2008	Part Number 5992–3387 • Divided the document into three parts: *Protecting Systems*, *Protecting Data*, and *Protecting Identity*. • Added a chapter to document HP-UX Standard Mode Security Extensions (see Chapter 3). • Replaced Security Patch Check with Software Assistant. • Added a figure to show the HP-UX Bastille user interface. • Added the HP-UX Bastille configuration log file `assessment-log.config`. • Made various edits.
October 2007	Part Number 5992-2395 • Added a chapter to describe HP-UX Bastille.
August 2007	Part Number 5992-1933 • Removed Process Resource Manager (PRM) from the product list that does not support shadow passwords (see Section 2.4.5). • Corrected `search` to `nsearch` in *`permission_list`* (see Section 6.4.2).
February 2007	Part Number 5991-6482 First Edition

NOTE: The volumes in the *HP-UX System Administrator's Guide* can be updated independently. Therefore, the latest versions of the volumes in the set can vary with time and with respect to each other. The latest versions of each volume are available at:

http://www.hp.com/go/hpux-core-docs

Click **HP-UX 11i v3**.

Intended Audience

The *HP-UX System Administrator's Guide* is written for administrators of HP-UX systems of all skill levels needing to administer HP-UX systems beginning with Release HP-UX 11i version 3.

While many topics in this set apply to previous releases, much has changed in HP-UX 11i version 3; therefore, for information about prior releases, see *Managing Systems and Workgroups, a Guide for System Administrators*.

About This Document Set

The *HP-UX System Administrator's Guide* documents the core set of tasks (and associated concepts) necessary to administer systems running HP-UX 11i Version 3. It is comprised of the following volumes:

Overview	Provides a high-level view of HP-UX 11i, its components, and how they relate to each other.
Configuration Management	Describes many of the tasks that you must perform to configure and customize system settings and the behavior of subsystems.
Logical Volume Management	Documents how to configure physical volumes, volume groups, and logical volumes using the HP Logical Volume Manager (LVM).
Security Management	Documents the data and system security features of HP-UX 11i.
Routine Management Tasks	Documents many of the ongoing tasks you must perform to keep your system running smoothly.

HP-UX System Administrator's Guide: Security Management is divided into three parts: *Protecting Systems*, *Protecting Data*, and *Protecting Identity*. These parts include the following topics:

Chapter 1	Describes security considerations related to the boot and installation process.
Chapter 2	Describes how to administer user and system security after the operating system is installed.
Chapter 3	Describes the features and components of HP-UX Standard Mode Security Extentions.
Chapter 4	Describes how to secure remote access to your system.
Chapter 5	Describes how to control and protect file systems.
Chapter 6	Describes compartments and how to isolate components of a system from one another.
Chapter 7	Describes fine-grained privileges and how to divide the powers of superusers into a set of privileges.
Chapter 8	Describes the features and components of HP-UX Role-Based Access Control.
Chapter 9	Describes the administration of the audit system.
Appendix A	Describes trusted systems.
Appendix B	Describes other security products.

HP-UX 11i Release Names and Release Identifiers

With HP-UX 11i, HP delivers a highly available, secure, and manageable operating system. HP-UX 11i supports enterprise, mission-critical, and technical computing environments and is available on both HP 9000 systems and HP Integrity servers.

Each HP-UX 11i release has an associated release name and release identifier. The `uname` command with the `-r` option returns the release identifier. See the following table for a list of releases available for HP-UX 11i:

Release Identifier	Release Name	Supported Processor Architecture
B.11.11	HP-UX 11i version 1	HP 9000
B.11.23	HP-UX 11i version 2	Intel™ Itanium™
B.11.23	HP-UX 11i version 2, September 2004	HP 9000 Itanium
B.11.31	HP-UX 11i version 3	HP 9000 Itanium

For information on supported systems and processor architecture for various versions of HP-UX 11i, see the HP-UX 11i system release notes specific to the version of HP-UX you are running (for example, the *HP-UX 11i Version 3 Release Notes*).

Finding HP-UX Information

The following table outlines where to find general system administration information for HP-UX. However, it does not include information for specific products.

If you need to	Refer To	Located at
Find out: • What has changed in HP-UX releases • The contents of the Operating Environments • Firmware requirements and supported systems for a specific release	The HP-UX 11i Release Notes specific to your version of HP-UX. For example, you may want to see the *HP-UX 11i Version 3 Release Notes*.	• HP Instant Information media • http://www.hp.com/go/hpux-core-docs Click **HP-UX 11i v3**. • `/usr/share/doc/` directory The `/usr/share/doc` directory contains only the original release note for your version of HP-UX. For revised release notes, see your latest HP Instant Information media or the Business Support Center: http://www.hp.com/go/hpux-core-docs Click **HP-UX 11i v3**.
Install or update HP-UX	• *Read Before Installing or Updating to HP-UX* • *HP-UX 11i Installation and Update Guide* **NOTE:** See the documents for your specific version of HP-UX.	• Media Kit (supplied with the Operating Environment) • HP Instant Information media • http://www.hp.com/go/hpux-core-docs Click **HP-UX 11i v3**.
Administer an HP-UX system	Releases beginning with HP-UX 11i Version 3: • *HP-UX System Administrator's Guide* (a multivolume set) Other sources of system administration information: • *nPartition Admnistrator's Guide* • *Planning Superdome Configurations* (white paper)	• HP Instant Information CD-ROM • http://www.hp.com/go/hpux-core-docs Click **HP-UX 11i v3**. • Planning Superdome Configurations (white paper)

Related Information

Additional information about Security and HP-UX can be found at www.hp.com/go/hpux-security-docs.

In particular, the following documents are available:

- *HP-UX AAA Server Administrator's Guide*
- *HP-UX Host Intrusion Detection System Administrator's Guide*
- *HP-UX IPFilter Administrator's Guide*
- *HP-UX IPSec Administrator's Guide*
- *HP-UX Secure Shell Release Notes*

Conventions

This document uses the following typographical conventions.

reboot(1M)	An HP-UX manpage. *reboot* is the name and *1M* is the section in the *HP-UX Reference*. On the Web and on the Instant Information media, it may be a hot link to the manpage itself. From the HP-UX command line, you can enter "`man reboot`" or "`man 1M reboot`" to view the manpage. See *man*(1) for more information.
Book Title	The title of a book. On the web and on the Instant Information media, it may be a hot link to the book itself.
KeyCap	The name of a keyboard key. **Return** and **Enter** both refer to the same key.
Emphasis	Text that is emphasized.
Emphasis	Text that is strongly emphasized.
Term	The introduction of an important word or phrase.
`ComputerOut`	Text displayed by the computer.
`UserInput`	Commands and other text that you type.
`Command`	A command name or qualified command phrase.
`Variable`	The name of a variable that you may replace in a command or function or information in a display that represents several possible values.
[ ]	The contents are optional in formats and command descriptions.
{ }	The contents are required in formats and command descriptions. If the contents are a list separated by |, you must choose one of the items
. . .	The preceding element may be repeated an arbitrary number of times.
|	Separates items in a list of choices.

Part I Protecting Systems

One critical factor in enterprise security is system minimization and hardening. HP-UX 11i offers a set of security features designed to address known and unknown vulnerabilities by running only the services that are needed, thus minimizing a potential point of attack.

This section discusses the following HP-UX tools that protect a system against an attack, and detect and react to threats:

- Installing the HP-UX operating environment securely (Chapter 1)
- Administering user and system security (Chapter 2)
- Standard Mode Security Extensions (Chapter 3)
- Remote access security administration (Chapter 4)

1 Installing the HP-UX Operating Environment Securely

This chapter describes security considerations related to the boot and installation processes, including the following topics:

- Installation security considerations (Section 1.1)
- Preventing security breaches during the boot process (Section 1.2)
- Enable login security for root (Section 1.3)
- Using boot authentication to prevent unauthorized access (Section 1.4)
- Setting Install-Time Security options (Section 1.5)
- Installing security patches (Section 1.6)
- Postinstallation security tips for backup and recovery (Section 1.7)

1.1 Installation Security Considerations

Before you install or update to a new operating system or new software, make a practice of addressing security considerations. Make the following security measures part of your preparation for installation:

- Review the contents of your media kit. Read the Release Notes and other related information at the Business Support Center:

  http://www.hp.com/go/hpux-core-docs

  Click **HP-UX 11i v3**.

- Decide which software you need and which you do not need. Do not install unnecessary software. Consult other chapters of this document for help deciding on security software products.
- Disconnect or disengage your system from the network, especially from a public network, until your security modifications are complete. Consider what, if any, security level you would like to deploy with. See Section 1.5 for more information.
- Make sure the system console is physically protected and your LAN console is either disconnected, or used only through a network where clear-text-protocols like `telnet` are allowed/protected. This is an important security consideration. Restricting access to the system console helps prevent unauthorized persons from changing the security settings of your system.
- Install the latest patches, especially security patches. See Section 1.6 for more information.
- Maintain a backup and recovery system. See Section 1.7 for more information.

1.2 Preventing Security Breaches During the Boot Process

Security breaches can occur during the boot sequence. The boot process can be interrupted, allowing an unauthorized person to access the system. If certain system files are altered incorrectly or maliciously before the reboot, the system can have problems during and after the reboot. Therefore, perform these preventative tasks:

- Make sure the system and system console are physically secure and that only authorized users have access.
- Enable the boot authentication feature to allow only specified users to boot the system to single user mode. See Section 1.4.
- Make sure system files are write protected; some might need to be read protected.

Following is a summary of the boot sequence that occurs when you turn on or reset the computer. See *HP-UX System Administrator's Guide: Routine Management Tasks* for more information on the boot sequence.

1. During booting, there is about a 10-second wait that allows you to override the automatic boot sequence. At this point, an intruder can interrupt the boot sequence and enter the system.

   You can gain root access when you interrupt the boot sequence by pressing any key. The ISL prompts you for a command. Entering the following command causes the system to be in single-user mode:

   ```
   ISL> hpux -is
   ```

   If you are not using boot authentication, a user can then log in as root with no password.

   Boot authentication allows only specified users to log in as root.

2. If the boot sequence is not interrupted, the initialization process continues.
3. HP-UX goes through its initialization process and begins normal operation, ready for login. At this point another security breach can occur if an intruder has already gained root access.

If an intruder interrupts the boot process, they have gained root access to the system and theoretically own the system. This ownership allows them to make changes to the system through a great number of mechanisms.

1.3 Enable Login Security for root

Many network protocols such as `rlogind` and `telnetd` do not encrypt network communication, making it easy for an intruder to sniff the administrative passwords from the network. Try to minimize the usage of these nonsecure protocols.

To prevent an administrative login through such a protocol, you can use the `/etc/securetty` file to limit logging in to the root account only through the system console. For instance, to restrict root logins to only the console, create the`/etc/security` file with a single line consisting of console. For more information, see *login*(1).

1.4 Using Boot Authentication to Prevent Unauthorized Access

The boot authentication feature protects single-user mode boot with password authentication. It makes it possible to configure a system so that only authorized users are allowed to boot the machine into single-user mode. The boot authentication feature must be enabled before you reboot the system.

Boot authentication is configured by two attributes in the `/etc/default/security` file:

- `BOOT_AUTH` enables or disables boot authentication. Specify `BOOT_AUTH=1` to enable boot authentication. By default, authentication is disabled (`BOOT_AUTH=0`).
- `BOOT_USERS` defines who can log in as root when the boot authentication feature is enabled. The names listed in `BOOT_USERS` are separated by commas. For example:

  ```
  BOOT_USERS=root,mary,jack,amy,jane
  ```

  `BOOT_USERS=root` is the default value.

The `/etc/default/security` configuration file is explained in Chapter 2 and in *security*(4).

1.5 Setting Install-Time Security Options

The Install-Time Security (ITS) options allow you to configure an HP-UX Bastille security lockdown engine, which can include an HP-UX IPFilter firewall. After system installation is complete, it will have one of the preconfigured levels of security.

During installation, you can choose from four preconfigured levels of security:

Sec00Tools	Install the security infrastructure but without enabling optional security features. This is the default.
Sec10Host	Install a host-based lockdown system, without HP-UX IPFilter firewall configuration. With this level of security, most network services are disabled. These services can be reinstated by running the *bastille*(1M) command.
Sec20MngDMZ	Install a managed lockdown system that blocks most incoming traffic with an HP-UX IPFilter firewall.
Sec30DMZ	Install a DMZ Full lockdown system, which is a host-based and IPFilter network lockdown. HP-UX IPFilter blocks almost all incoming connections.

For information on ITS and HP-UX Bastille, see the *HP-UX Bastille User Guide*:

www.hp.com/go/hpux-security-docs

Click **HP-UX Bastille Software**.

For information on HP-UX IPFilter, see the *HP-UX IPFilter Administrator's Guide*:

www.hp.com/go/hpux-security-docs

Click **HP-UX IPFilter Software**.

1.6 Installing Security Patches

Immediately after installation, apply the required and recommended patches using HP-UX Software Assistant (SWA).

SWA is a command-line-based tool that consolidates and simplifies patch management and security bulletin management on HP-UX systems. The SWA tool replaces Security Patch Check (SPC), and is the HP-recommended utility to use to maintain currency with HP-published security bulletins for HP-UX software.

NOTE: Use of the Software Assistant software tool can help improve system security, but it does not guarantee system security.

For more information on SWA, see the *HP-UX Software Assistant System Administration Guide*:

www.hp.com/go/hpux-security-docs

Click **HP-UX Software Assistant (SWA) Software**.

1.7 Postinstallation Security Tips for Backup and Recovery

After the system is running, you must still maintain its security. Be diligent in maintaining system backup and recovery files. Following are some guidelines:

- Use only the `fbackup` and `frecover` commands to back up and recover files selectively. Only `fbackup` and `frecover` retain access control lists (ACLs). Use the `-A` option of these commands when backing up and recovering files for use on systems that do not implement ACLs. See *fbackup*(1M) and *frecover*(1M).
- If you plan to recover the files to another system, be sure that the user's user name and group name on both systems are consistent.
- Remember that the 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 a minimum of 6 months, and then recycle the media.
- Perform daily incremental and full weekly backups.

  Synchronize the backup schedule with the information flow in your organization. For example, if a major database is updated every Friday, you might want to schedule the weekly backup on Friday evenings.
- If you must back up all files on schedule, request that all users log off before performing the backup. The `fbackup` command warns you if a file is changing while the backup is being performed.

- Examine the log file of latest backups to identify problems occurring during backup. Set restrictive permissions on the backup log file.
- Be aware that the `frecover` command allows you to overwrite a file. However, the file retains the permissions and ACLs set when the file was backed up.
- Test the recovery process beforehand to make sure you can fully recover data in the event of an emergency.
- When recovering files from another machine, you might have to execute the `chown` 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 the `frecover` command. If the owner and group names have different meanings on different systems, recovery results might be unexpected and not what you wanted.
- Although a power failure should not cause file loss, if someone reports a lost file after a power failure, look for it in the `/lost+found` directory before restoring it from a backup tape.
- To verify contents of the tape being recovered, use the `-I` option of the `frecover` command to preview the index of files on the tape. Existing permissions of a file system are kept intact by the backup. The `frecover` command prevents you from reading the file if the permissions on the file forbid it.
- Never recover in place any critical files, such as `/etc/passwd` or those in `/tcb/files`. Instead, restore the file to a temporary directory (do not use `/tmp`), and give this directory permissions `drwx------`, preventing anyone else from using it. Compare the restored files with those to be replaced. Make any necessary changes.
- Be sure to turn auditing on. Auditing is not enabled automatically when you have recovered the system.

2 Administering User and System Security

This chapter addresses basic user security after the operating system is installed. It focuses on logins, passwords, and other user interactions with the system. The following topics are discussed:

- Managing user access (Section 2.1)
- Authenticating users during login (Section 2.2)
- Authenticating users with PAM (Section 2.3)
- Managing passwords (Section 2.4)
- Defining system security attributes (Section 2.5)
- Handling setuid and setgid programs (Section 2.6)
- Preventing stack buffer overflow attacks (Section 2.7)
- Protecting unattended terminals and workstations (Section 2.8)
- Protecting against system access by remote devices (Section 2.9)
- Securing login banners (Section 2.10)
- Protecting the root account (Section 2.11)

2.1 Managing User Access

Authorized users gain access to the system by supplying a valid user name (login name) and password. Each user is defined by an entry in the `/etc/passwd` file. Use the HP System Management Homepage (HP SMH) to add, remove, deactivate, reactivate, or modify a user account.

For more information about passwords, refer to *passwd*(4), *passwd*(1), and see Section 2.4 in this document.

2.1.1 Monitoring User Accounts

Following are guidelines for monitoring user accounts:

- Regularly examine the output from the `last`, `lastb`, and `who` commands for unusual logins.
- Verify that all users with accounts have a legitimate business need to access the system.
- Be alert for multiple users sharing the same user account. Do not allow two users to share the same user account.
- Verify that no user accounts share the same user ID (UID).
- Ensure that all accounts have secure passwords that change regularly.
- Verify that all user home directories have the appropriate permissions. Most home directories have read access but no write access to other users. For better protection, set the read, write, and execute permissions for the directory owner only.

- Ensure that all users understand the security policies. Place a company security policies file in each home directory.
- Examine the `/etc/passwd` file or other appropriate user database for unused accounts, and especially for users who have left the company.
- Examine root accounts to see who has root access.
- Consider implementing HP-UX Role-based Access Control to minimize the risks associated with multiple users having access to the root account. For more information, see Chapter 8.
- Examine guest accounts to see how often they are used.

2.1.2 Monitoring Guest Accounts

For the highest level of security, do not allow guest or open accounts. If you do have guest accounts, then do the following:

- Change the guest password frequently. You can specify the password.
- Use a restricted shell (`rsh`) to limit system access. For information about the `rsh` command, refer to *sh*(1) and *sh-posix*(1).
- Guest accounts are often forgotten. Use one of the following methods to disable the guest account when not in use:
  — Use per-user security attributes to automatically disable the account after a certain number of inactive days. For more information, refer to *security*(4) and see Section 2.5.2.2.
  — Use the following command to lock the guest account:
  ```
  # passwd -l guest
  ```
  — Use the following command to delete the guest account:
  ```
  # userdel guest
  ```
- Schedule an `at` job to automatically lock temporary accounts:
  ```
  # at now +14 days passwd -l tempacct
  ```
- Regularly scan the `/var/adm/wtmp` and `/var/adm/sulog` files to check for unused accounts.

Refer to *sh*(1) and *su*(1) for more information.

2.1.3 Creating Application User Accounts

If users only use HP-UX to launch an application, they do not require access to a shell. These users should only be using the application, such as a database management system, and not need access to any HP-UX functionality.

To restrict access to HP-UX, modify the `/etc/passwd` file so that only a specific command is executed after the user logs in. The `/etc/passwd` file contains essential information required during login:

- User name
- Encrypted password
- User ID
- Group ID
- Comment field
- Home directory
- Login program

Typically, the login program is a shell, such as `/bin/sh`, but it does not have to be a shell. You can create a captive account—an account that logs a user directly into an application—by identifying the application as the login shell.

Following is an example of restricting a user to run only the `date` command. The `/etc/passwd` entry is:

```
username:rc70x.4,sx2:20:1:run only date command:/home/date:/usr/bin/date
```

At the login prompt, a user enters *`username`* and the appropriate password. The `date` command is executed and then the user is immediately logged out.

```
login:username

Password:xxxxxx

Tue Nov 14 18:38:38 PDT 2006
```

2.1.4 Managing Group Accounts

When a group has to share or have access to project-related files, follow these steps to ensure security:

1. Verify that each member has an entry in `/etc/passwd`.
2. Create an entry for the group in the `/etc/group` file.
3. Create a shared directory for the group.

   ```
   drwxrwx-- root project /home/projects
   ```

4. Set the umask in each group member's `~/.profile`. In the following example, users in the group can read, write, and execute files, but no one else can:

   ```
   umask u=rwx,g=rwx, o=
   ```

2.2 Authenticating Users During Login

To gain access to a system and its resources, users are required to log in. By controlling access to the system, you can try to prevent unauthorized users from accessing the system. However, even if unauthorized users do gain access, you can still prevent them from running programs that consume resources and from accessing system data. This section explains what happens during the `login` process from the time you type your user name to the time you get a shell prompt.

2.2.1 Explanation of the Login Process

The following steps describe the login process. This information shows how important it is to create unique user names and to maintain a password security policy. For more information, refer to *login*(1).

1. After the system is installed, the desktop Login Manager displays a login screen. The Common Desktop Environment (CDE) displays a CDE login screen if it is installed.
2. The `init` program spawns a `getty` process, which prompts you for a user name. You enter your user name. The `getty` program passes the user name to the `login` program.
3. The `login` program searches`/etc/passwd` for the user name.
   - If the user name exists, `login` goes to step 4 .
   - If the user name does not exist, then `login` does the following checks:
     — Prompts for a password (`Password:`).
     — If an invalid password is entered, the system displays the `Invalid login` error message.
     — Updates the `/var/adm/btmp` file if it exists. The `/var/adm/btmp` file keeps track of invalid login attempts. See Section 2.2.2 for more information.
     — Exits after three consecutive invalid login attempts.
4. The `login` process verifies the `/etc/passwd` file.
   - If the password field is set, `login` prompts for a password and goes to step 5.
   - If the password field is not set, the user does not need a password and `login` goes to step 6 .
5. The `login` process compares the password to the encrypted password in `/etc/passwd`.
   - If the password matches, `login` goes to step 6.
   - If the password does not match, `login` displays `Invalid login`. The `login` process allows three consecutive login attempts. After the user's third invalid login attempt, `login` exits.
6. The `login` process updates the `/var/adm/wtmp` file, which keeps track of valid logins. See Section 2.2.2 for more information.

   After a successful login, the user and group IDs, group access list, and working directory are initialized.
7. The `login` process then runs the command in the command field of the `/etc/passwd` file. Typically, the command field is the path name of a shell, such

as /bin/ksh, /bin/csh, or /bin/sh. If the command field is empty, the default is /bin/sh.

The command field does not have to be a shell. See Section 2.1.3 for an example of running another command.

8. After the shell initialization is complete, the system displays a prompt and waits for user input.

You can have the login process perform further user authentication using the Pluggable Authentication Modules (PAM). For more information, see *pam.conf*(4) and Section 2.3.

2.2.2 Checking the login Tracking Files (btmp and wtmp)

The following files keep a log of logins:

- The /var/adm/btmp file keeps track of failed logins.
- The /var/adm/wtmp file keeps track of successful logins.

Use the lastb command to read the /var/adm/btmp file to see if unauthorized users have attempted to log in.

Use the last command to read the/var/adm/wtmp file.

The last and lastb commands display the most recent user information, in descending order.

The wtmp and btmp files tend to grow without bound, so check them regularly. Periodically remove information that is no longer useful to prevent the file from becoming too large. The wtmp and btmp files are not created by the programs that maintain them. If these files are removed, login record keeping is turned off.

A common mistake users make during login is to enter the password, or part of the password at the login prompt. This failed login is recorded in the btmps file and exposes the password or partial password. For this reason, the file protection on the btmps should be set so that it is only readable by administrators.

```
# chmod 400 /var/adm/btmps
```

If the security policy requires that past sessions of one user cannot be viewed by another user, then the file protection of the /var/adm/wtmp file may also need to be changed.

See *last*(1), *utmp*(4), and *wtmp*(4) for more information.

The utmp database is a user accounting database managed and synchronized according to /var/adm/utmp by the utmpd command. Application programs can access the utmps database. See *utmpd*(1M) and *utmps*(4).

2.2.2.1 Last Command Examples

This section contains examples of using the last command. The following command lists all of the root sessions and all sessions on the console terminal:

```
# last root console | more
root pts/1 Mon Mar 12 16:22 - 18:04 (01:41)
```

```
abcdeux console Mon Mar 12 10:13 - 10:19 (00:06)
root pts/2 Fri Mar 9 13:51 - 15:12 (01:21)
abcdeux console Thu Mar 8 12:21 - 12:22 (00:00)
root pts/ta Wed Mar 7 15:38 - 18:13 (02:34)
```

The following command lists when reboots have occurred:

```
# last reboot
reboot system boot Sun Mar 28 18:06 still logged in
reboot system boot Sun Mar 28 17:48 - 18:06 (00:17)
reboot system boot Sun Mar 28 17:40 - 17:48 (00:08)
reboot system boot Thu Feb 19 18:25 - 17:40 (37+23:15)
reboot system boot Mon Feb 16 13:56 - 18:25 (3+04:28)
```

2.2.3 Checking Who Is Logged In

The `who` command examines the `/etc/utmp` file to obtain current user login information. In addition, the `who` command can list logins, logoffs, reboots, changes to the system clock, and processes spawned by the `init` process.

Use the `who -u` command to monitor who is currently logged in. For example:

```
# who -u
aperson console Aug 5 11:28 old 5796 system.home.company.com
aperson pts/0 Aug 17 18:11 0:03 24944 system
aperson pts/1 Aug 5 11:28 1:14 5840 system
```

See *who*(1) for more information.

2.3 Authenticating Users with PAM

The Pluggable Authentication Modules (PAM) are an industry-standard framework providing authentication, account management, session management, and password services. This section gives an overview of PAM and describes the PAM configuration files: `/etc/pam.conf` and `/etc/pam_user.conf`.

For more information, see *pam*(3), *pam_**(5), *pam.conf*(4), *pam_user.conf*(4), and *security*(4).

2.3.1 Overview

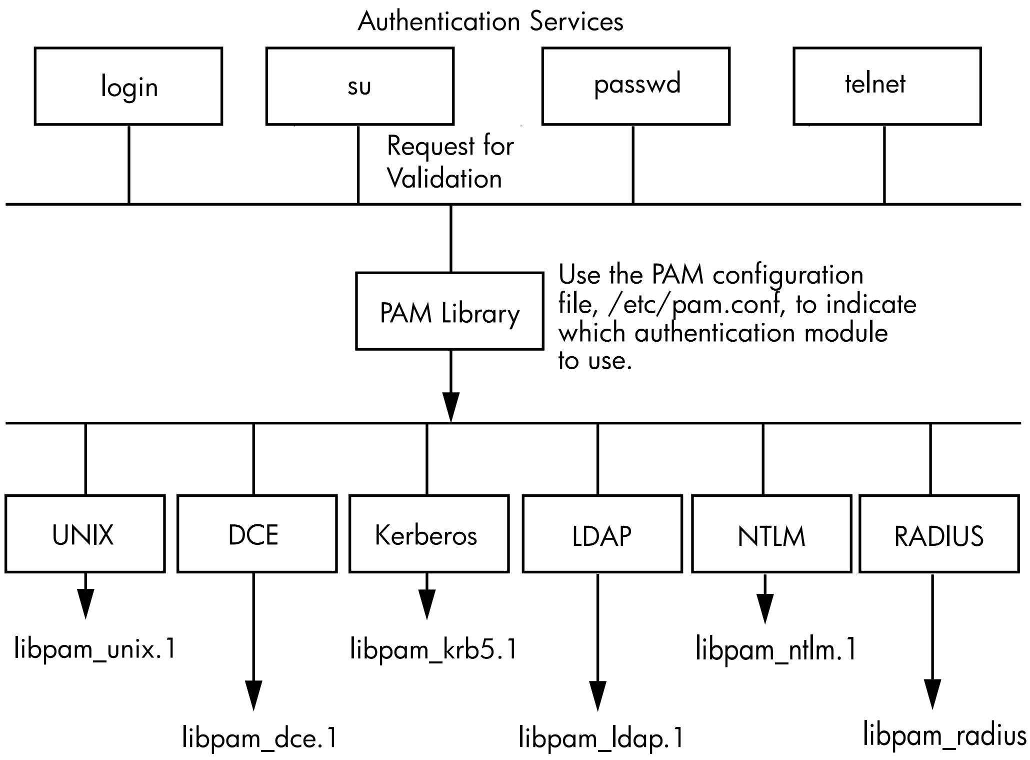
PAM provides the flexibility to choose any authentication service available on the system. The PAM framework also enables you to plug in new authentication service modules and make them available without modifying the applications.

Whenever a user logs in either locally or remotely (for example, using `login` or `rlogin`), the user must be checked or authenticated as a valid user of the system. As authentication methods improve and change over time, the login services would also have to change. To avoid constant changing of the login services just to revise the authentication code, PAM was developed so that different authentication methods can be used without modifying the login code.

As a result, login authentication, account checking, and password modification use the PAM interface.

Programs requiring user authentication pass their requests to PAM, which determines the correct verification method and returns the appropriate response. The programs do not need to know what authentication method is being used. See Figure 2-1 for an overview.

Figure 2-1 HP-UX Authentication Modules Under PAM



The authentication methods are specified on both a systemwide and individual user basis using the following PAM system files:

`/etc/pam.conf`	Systemwide control file. Defines which service modules are to be paired with services. These are regarded as system defaults.
`/etc/pam_user.conf`	Individual user control file. Defines which options are to be used by service modules on specific users. This is an optional file.

See *pam*(3), *pam.conf*(4), *pam_updbe*(5), *pam_user.conf*(4) for more information.

2.3.2 PAM Libraries

PAM service modules are implemented by shared libraries. PAM enables multiple authentication technologies to co-exist in HP-UX. The `/etc/pam.conf` configuration file determines which authentication module to use. The PAM libraries are as follows:

- `PAM_DCE`

  The PAM_DCE modules enable integration of DCE into the system entry services (such as `login`, `telnet`, `rlogin`, `ftp`). The PAM_DCE modules provide functionality for the authentication, account management, and password management modules. These modules are supported through the PAM_DCE library, `/usr/lib/security/pam_dce.sl`. See *pam_dce*(5) for more information.

- `PAM_HPSEC`

  The PAM_HPSEC modules manage extensions specific to HP-UX for authentication, account management, password management, and session management. The use of `/usr/lib/security/$ISA/libpam_hpsec.so.1` is mandatory for services such as `login`, `dtlogin`, `ftp`, `su`, `remsh`, `rexec`, and `ssh`. These services must place `libpam_hpsec.so.1` on the top of the stack above one or more nonoptional modules. The `pam_hpsec` module also enforces several attributes defined in `/etc/default/security`. See *pam_hpsec*(5) and *security*(4) for more information.

- `PAM_KRB5`

  Kerberos is a network authentication protocol that enables secure communication over networks without transmitting passwords in clear text. A password is authenticated by the Key Distribution Center (KDC), which then issues a Ticket Granting Ticket (TGT). The PAM Kerberos shared library is `/usr/lib/security/libpam_krb5.1`. See *pam_krb5*(5) for more information.

- `PAM_LDAP`

  The Lightweight Directory Access Protocol (LDAP) is a standard for centralizing user, group, and network management information through directory services. Authentication takes place on an LDAP directory server.

  For more information, see the HP-UX LDAP-UX Integration Software documentation:

  www.hp.com/go/hpux-security-docs

  Click **HP-UX LDAP-UX Integration Software**.

- `PAM_NTLM`

  The PAM NT LAN Manager enables HP-UX users to be authenticated against Windows servers during system login. PAM NTLM uses NT servers to authenticate users logging in to an HP-UX system.

  For more information, see the *HP CIFS Client Administrator's Guide*:

  http://www.hp.com/go/hpux-networking-docs

Click **HP-UX 11i v3 Networking Software**.

- PAM_RADIUS

  The HP-UX PAM RADIUS module provides authentication and session management for PAM enabled applications (typically system entry services such as login and ftp) through RADIUS server using the pam.conf configuration file. The HP-UX PAM RADIUS module consists of the following two modules:

  — Authentication module

  — Session management module

  It also provides null function for account management. All modules are supported through the same dynamically loadable library, /usr/lib/security/libpam_radius.1.

  The HP-UX PAM RADIUS module supports two-factor authentication by requesting the user's password and One Time Password (OTP).

  For more information, see *pam_radius*(5).

- PAM_UNIX

  The PAM_UNIX modules provide functionality for all four PAM modules: authentication, account management, session management, and password management. The modules are supported through the PAM UNIX library, /usr/lib/security/libpam_unix.1. See *pam_unix*(5) for more information.

- PAM_UPDBE

  The user policy definition service module for PAM, /usr/lib/security/libpam_updbe.1, reads options defined in the user configuration file, /etc/pam_user.conf, and stores the information in the PAM handle for subsequent service modules to use. See *pam_updbe*(5) for more information.

2.3.3 Systemwide Configuration Using /etc/pam.conf

The PAM configuration file /etc/pam.conf defines the security mechanisms that are used to authenticate users. Its default values provide the customary operation of the system under both standard HP-UX and trusted systems. It also provides support for controls on individual users and for the DCE integrated login functionality.

NOTE: For DCE, use the auth.adm utility to create the desired configuration file. This file is functionally equivalent to the former HP integrated login auth.conf file. See *auth.adm*(1m) for more information.

The libpam and libpam_unix PAM libraries and the /etc/pam.conf configuration file must be on the system in order for users to be able to log in or change passwords.

HP-UX authentication is dependent upon the file /etc/pam.conf. This file must be owned by root with the following file permissions:

```
-r--r--r-- 1 root sys  1050 Nov  8 10:16 /etc/pam.conf
```

If this file is corrupt or missing from the system, root can log in to the console in single-user mode to fix the problem.

The protected service names are listed in the system control file, `/etc/pam.conf`, under four test categories (*`module-type`*): authentication, account, session, and password.

See *pam*(3), *pam.conf*(4), and *pam_user.conf*(4) for more information.

2.3.4 Sample /etc/pam.conf File

Following is a partial listing of a sample `/etc/pam.conf` file. Lines beginning with pound (#) are comment lines. The sections in `/etc/pam.conf` are authentication management, account management, session management, and password management.

```
#
# PAM configuration
#
# Notes:
#
# If the path to a library is not absolute, it is assumed to be
# relative to the directory /usr/lib/security/$ISA/
#
# For PA applications, /usr/lib/security/$ISA/libpam_unix.so.1 is a
# symbolic link that points to the corresponding PA (32 or 64-bit) PAM
# backend library.
#
# The $ISA (i.e. Instruction Set Architecture) token will be replaced
# by the PAM engine with an appropriate directory string.
# See pam.conf(4).
#
# Also note that the use of pam_hpsec(5) is mandatory for some of
# the services. See pam_hpsec(5).
#
# Authentication management
#
login    auth required  libpam_hpsec.so.1
login    auth required  libpam_hpsec.so.1
su       auth required  libpam.hpsec.so.1  bypass_setaud
su       auth required  libpam_unix.so.1
dtlogin  auth required  libpam_hpsec.so.1
dtlogin  auth required  libpam_unix.so.1
dtaction auth required  libpam_hpsec.so.1
dtaction auth required  libpam_unix.so.1
ftp      auth required  libpam_hpsec.so.1
ftp      auth required  libpam_unix.so.1
rcomds   auth required  libpam_hpsec.so.1
rcomds   auth required  libpam_unix.so.1
sshd     auth required  libpam_hpsec.so.1
sshd     auth required  libpam_unix.so.1
OTHER    auth required  libpam_unix.so.1
#
# Account management
#
login    account required       libpam_hpsec.so.1
login    account required       libpam_unix.so.1
```

```
su       account required       libpam_hpsec.so.1
su       account required       libpam_unix.so.1
```

2.3.5 The /etc/pam_user.conf User Configuration File

The PAM configuration file, `/etc/pam_user.conf`, configures PAM on a per-user basis. This file is optional. It is needed only if PAM applications need to behave differently for different users.

You assign different options to individual users by listing them in `/etc/pam_user.conf`. For a *`login-name`* listed here, the *`options`* listed here replace any *`options`* specified for the *`module-type`* and *`module-path`* in `/etc/pam.conf`.

The entries in `/etc/pam_user.conf` use the following syntax:

`login-name module-type module-path options`

where:

`login-name`	User's login name.
`module-type`	The *`module-type`* specified in `/etc/pam.conf`.
`module-path`	The *`module-path`* associated with *`module-type`* in `/etc/pam.conf`.
`options`	Zero or more options recognized by the module.

The default contents of `/etc/pam_user.conf` are comments:

```
#
# This file defines PAM configuration for a user. The configuration
# here overrides pam.conf.
#
# The format for each entry is:
# user_name  module_type  module_path options
#
# For example:
#
# user_a        auth      /usr/lib/security/libpam_unix.1     debug
# user_a        auth      /usr/lib/security/libpam_dce.1      try_first_pass
# user_a        password  /usr/lib/security/libpam_unix.1     debug
#
# user_b        auth      /usr/lib/security/libpam_unix.1     debug use_psd
# user_b        password  /usr/lib/security/libpam_unix.1     debug use_psd
#
# See the pam_user.conf(4) manual page for more information
#
```

2.3.6 Examples: How PAM Works for Login

The following examples describe the `auth` process for `login`, depending upon how the `/etc/pam.conf` file is configured:

- If `/etc/pam.conf` contains a single standard `login auth`, such as the following, then `login` proceeds normally:

```
login     auth  required  /usr/lib/security/libpam_unix.1
```

- If there are two or more systemwide `login auth` entries, such as the following, they are taken in order:

  ```
  login    auth  required  /usr/lib/security/libpam_unix.1
  login    auth  required  /usr/lib/security/libpam_dce.1
  ```

  In this case, the standard HP-UX `login` process is executed. Then the DCE authentication process occurs. If both are satisfied, then the login is successful. Both processes are performed, even if the user fails one of them.

- If you require different authentication methods for different users, place the special entry `libpam_udpbe` ahead of the authentication modules in `/etc/pam.conf` (the lines are numbered for easy reference):

  ```
  #/etc/pam.conf
  #1
  login     auth  required  /usr/lib/security/libpam_udpbe.1
  #2
  login     auth  required  /usr/lib/security/libpam_unix.1
  #3
  login     auth  required  /usr/lib/security/libpam_dce.1
  ```

  Then place entries for each affected user in `/etc/pam_user.conf`:

  ```
  #/etc/pam_user.conf
  #4
  allan  auth  /usr/lib/security/libpam_unix.1  debug
  #5
  allan  auth  /usr/lib/security/libpam_dce.1   try_first_pass
  #6
  isabel auth  /usr/lib/security/libpam_unix.1  debug  use_psd
  ```

  When `allan` logs in, line 1 in `/etc/pam.conf` causes PAM to read`/etc/pam_user.conf`. Because the module paths on lines 4 and 5 of `/etc/pam_user.conf` match the module paths on lines 2 and 3 of `/etc/pam.conf`, PAM temporarily replaces the null *`options`* fields of lines 2 and 3 of `/etc/pam.conf` with `debug` and `try_first_pass`, respectively. Then the modules specified by lines 2 and 3 are executed with the revised options.

  When `isabel` logs in, line 1 in `/etc/pam.conf` causes PAM to read `/etc/pam_user.conf` and temporarily replace the *`options`* field of line 2 of `/etc/pam.conf` with `debug use_psd`. Line 3 is unchanged. Then the modules specified by lines 2 and 3 are executed with the revised options.

  When `george` logs in, line 1 in `/etc/pam.conf` causes PAM to read `/etc/pam_user.conf`. Because entries for `george` do not exist, lines 2 and 3 of `/etc/pam_user.conf` are not changed. The modules specified by lines 2 and 3 are executed with no changes.

2.4 Managing Passwords

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

2.4.1 System Administrator Responsibilities

The system administrator and every user on the system must share responsibility for password security. System administrators perform the following security tasks:

- Ensure that all users have passwords.
- Maintain proper permissions on all system files, including the standard password and group files, `/etc/passwd` and `/etc/group`.
- Delete or nullify user IDs and passwords of users no longer eligible to access the system.
- Verify that all application passwords are encrypted.
- Verify that permissions on `/var/adm/btmp` and `/var/adm/wtmp` are set appropriately.
- Implement one-time passwords for single guest access.
- Inform users of their responsibilities regarding password security.
- Use password aging to force users to change their passwords regularly.
- Prevent reuse of recent passwords.
- Configure systemwide security attributes in the `/etc/default/security` file. See Section 2.5 and refer to *security*(4) for more information.
- Convert the system to use shadow passwords. See Section 2.4.5 and refer to *shadow*(4) and *pwconv*(1M) for more information.

2.4.2 User Responsibilities

Every user must observe the following rules:

- Remember the password and keep it secret at all times.
- Change the initial password immediately and continue to change it.
- Report any changes in status and any suspected security violations.
- Make sure no one is watching a password being entered.

2.4.3 Criteria of a Good Password

Observe the following guidelines when choosing a password and communicate these guidelines to users:

- Choose a password with at least 6 characters and no more than 80 characters. Special characters can include control characters and symbols, such as asterisks and slashes. In standard mode, only the first 8 characters are used.
- Do not choose a word found in a dictionary in any language, 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 (for example, if your login is `ann`; a bad password choice is `annann`).
- Consider using misspelled words or combined syllables from two unrelated words to make suitable passwords. Another popular method is to use the first characters of a favorite title or phrase for a password.
- Consider using a password generator that combines syllables to make pronounceable gibberish.
- Do not share passwords with other users. Management must forbid sharing of passwords.
- Always have a password. Do not have your password field cleared in the `/etc/passwd` file.

2.4.4 Changing the /etc/passwd Password File

A standard system maintains one password file: `/etc/passwd`.

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

Follow these guidelines if you need to change the password file:

- Do not permit any empty or null password fields; this is a security breach. An empty password field enables any user to set the password for that account.
- Do not edit the password file directly. Use HP SMH or the `useradd`, `userdel`, or `usermod` commands to modify password file entries. If you must edit the password file directly, use the `vipw` command and check it with the `pwck` command. See *vipw*(1M) and *pwck*(1M) for more information.

2.4.4.1 Examples of passwd Commands

Following are some useful `passwd` command examples:

- Reset a user's password:

  ```
  # passwd user1
  ```

- Force a password change at next login:

```
# passwd -f user1
```

- Lock or disable an account:

  ```
  # passwd -l user2
  ```

- Enable password aging:

  ```
  # passwd -n 7 -x 28 user1
  ```

- View password aging status for a specific user:

  ```
  # passwd -s user
  ```

- View password aging status for all users:

  ```
  # passwd -sa
  ```

2.4.4.2 The /etc/passwd File Format

The `/etc/passwd` file is used to authenticate a user at login time. The file contains an entry for every account on the HP-UX system. Each entry consists of seven fields, separated by colons. A typical `/etc/passwd` entry looks like this:

```
robin:Z.yxGaSvxAXGg:102:99:Robin Hood,Rm 3,x9876,408-555-1234:/home/robin:/usr/bin/sh
```

The fields contain the following information (listed in order), separated by colons:

1. `robin`—User (login) name, consisting of up to 8 characters.
2. `Z.yxGaSvxAXGg`—Encrypted password field.
3. `102`—User ID, an integer ranging from 0 to `MAXINT-1` (equal to 2,147,483,646 or 2^{31} -2).
4. `99`—Group ID, from `/etc/group`, an integer ranging from 0 to `MAXINT-1`.
5. `Robin Hood,Rm 3,x9876,408-555-1234`—Comment field, used to identify such information as the user's full name, location, and phone numbers. For historic reasons, this is also called the `gecos` field.
6. `/home/robin`—Home directory, the user's initial login directory.
7. `/usr/bin/sh`—Login shell path name, executed when the user logs in.

The user can change the password by invoking `passwd`, the comment field (fifth field) with `chfn`, and the login program path name (seventh field) with `chsh`. The system administrator sets the remaining fields. The user ID must be unique. See *chfn*(1), *chsh*(1), *passwd*(1), and *passwd*(4) for more information.

2.4.5 The /etc/shadow Shadow Password File

Increasing computational power available to malicious password decrytpers has made the nonhidden passwords in the `/etc/passwd` file vulnerable to decryption.

A shadow password enhances system security by hiding encrypted passwords in a shadow password file. You can move encrypted passwords previously stored in the publicly readable `/etc/passwd` file to the `/etc/shadow` file, which is accessible only by a user with the appropriate privileges.

Use the following commands to enable, verify, and disable shadow passwords:

- The `pwconv` command creates a shadow password file and copies the encrypted passwords from the `/etc/passwd` file to the `/etc/shadow` file.
- The `pwck` command checks the `/etc/passwd` and `/etc/shadow` files for inconsistencies.
- The `pwunconv` command copies the encryped passwords and aging information from the `/etc/shadow` file to the `/etc/passwd` file and then deletes the `/etc/shadow` file.

For more information, see *pwconv*(1M), *pwck*(1M), *pwunconv*(1M) and *shadow*(4).

Note the following points about the shadow password feature.

- When the shadow password feature is enabled, applications can be affected if they directly access the password field of the `/etc/passwd` file to obtain password and aging information. That field will now contain an `x`, indicating that the information is in `/etc/shadow`.

  Applications that use the PAM interfaces to authenticate, are not effected.

  To access the `/etc/shadow` file programmatically, use the `getspent` calls. These calls are similar to the `getpwent` calls for `/etc/passwd`. For more information, see *getspent*(3C) and *getpwent*(3C).
- In the `/etc/nsswitch.conf` file, shadow passwords are supported with `files`, `NIS`, and `LDAP` name services, but they may not be supported with other name server switch backends. To configure the system to use only `files`, `NIS`, and/or `LDAP`, ensure that the `passwd` line in `/etc/nsswitch.conf` contains only `files`, `NIS`, and/or `LDAP`. If `/etc/nsswitch.conf` does not exist, or if the `passwd` line is not present, then the default is `files` only. For more information, see *nsswitch.conf*(4).
- The shadow password is based on the de facto standard provided with other UNIX systems.

The following attributes, defined in `/etc/default/security`, apply to shadow passwords. See Section 2.5 and consult the *security*(4) manpage for more information.

- `INACTIVITY_MAXDAYS`—Number of days before expiring an account for inactivity.
- `PASSWORD_MINDAYS`—Minimum number of days before a password can be changed.
- `PASSWORD_MAXDAYS`—Maximum number of days that passwords are valid.
- `PASSWORD_WARNDAYS`—Number of days before warning users of password expiration.

Shadow passwords are supported with Serviceguard.

NOTE: Shadow passwords are not supported with LDAP-UX. Instead, LDAP-UX provides the ability to hide user passwords in the directory server itself. LDAP-UX also enforces centralized security policies, similar to `/etc/shadow`, based on the security policy of the directory server.

Shadow passwords are not supported by the applications that expect passwords to reside in `/etc/passwd`.

For more information, see the following manpages:

passwd(1), *pwck*(1M), *pwconv*(1M), *pwunconv*(1M), *getspent*(3C), *putspent*(3C), *nsswitch.conf*(4), *passwd*(4), *security*(4), *shadow*(4)

2.4.6 Eliminating Pseudo-Accounts and Protecting Key Subsystems in /etc/passwd

By tradition, the `/etc/passwd` file contains numerous "pseudo-accounts," which are entries not associated with individual users and which do not have true interactive login shells.

Some of these entries, such as `date`, `who`, `sync`, and `tty`, evolved strictly for user convenience, providing commands that could be executed without logging in. To tighten security, they have been eliminated in the distributed `/etc/passwd` so that these programs can be run only by a user who is logged in.

Other such entries remain in `/etc/passwd` because they are owners of files. Programs with owners such as `adm`, `bin`, `daemon`, `hpdb`, `lp`, and `uucp` encompass entire subsystems, and represent a special case. Because 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`. The customary pseudo- and special accounts are shown in Example 2-1.

Example 2-1 Pseudo- and Special System Accounts

```
root::0:3::/:/sbin/sh
daemon:*:1:5::/:/sbin/sh
bin:*:2:2::/usr/bin:/sbin/sh
sys:*:3:3::/:
adm:*:4:4::/var/adm:/sbin/sh
uucp:*:5:3::/var/spool/uucppublic:/usr/lbin/uucp/uucico
lp:*:9:7::/var/spool/lp:/sbin/sh
nuucp:*:11:11::/var/spool/uucppublic:/usr/lbin/uucp/uucico
hpdb:*:27:1:ALLBASE:/:/sbin/sh
nobody:*:-2:-2::/:
```

The key to the privileged status of these subsystems is their ability to grant access to programs under their jurisdiction without granting root access (`uid 0`). Instead, the `setuid` bit for the executable file is set and the effective user of the process corresponds to the owner of the executable file. For example, the `cancel` command is part of the `lp` subsystem and runs as effective user `lp`.

When the `setuid` is set, the security mediation of that subsystem enforces the security of all programs encompassed by the subsystem, not the entire system. Hence, the subsystem vulnerability to a breach of security is also limited to only those subsystem files. Breaches cannot affect the programs under different subsystems. For example, programs under `lp` do not affect those under `daemon`.

2.4.7 Secure Login with HP-UX Secure Shell

The HP-UX Secure Shell provides secure remote login, file transfer, and remote command execution. All client-server communication is encrypted. Passwords going across the network are never sent in clear text. For more information, see *ssh*(1) and Section 4.6.

2.4.8 Securing Passwords Stored in NIS

The Network Information Service (NIS) is part of the Network File System (NFS). NIS enables configuration administration of several hosts from a central location, a master server. Instead of having host configurations stored separately on each host, the information is consolidated onto a central location. The `/etc/password` file is among the several configuration files stored on the NIS server.

The `/etc/shadow` shadow password file is not supported on NIS.

See the *NFS Services Administrator's Guide* for information about NIS.

2.4.9 Securing Passwords Stored in LDAP Directory Server

LDAP-UX Client Services interoperates with PAM to authenticate passwords stored on an LDAP directory server. The `PAM_LDAP` library provides the authentication service.

2.5 Defining System Security Attributes

Security attributes provide additional control of system configurations, adding security enhancements to passwords, logins, and auditing.

There are more than 20 attributes. These attributes are described in *security*(4) . The categories of attributes are summarized as follows:

Login attributes	These attributes control login activities, such as login times, number of logins allowed, and the number of login failures allowed before locking and account.
Password attributes	These attributes control password activities, such as password length, number of characters and their types, history depth, number of days to change a password, and password expiration.

Boot attributes	These attributes control boot authentication, defining which users are authorized to boot the system into single-user mode. See boot authentication information in Chapter 1.
Switch user (`su`) attributes	These attributes define the `PATH` environment value, root group name for the `su` command, and whether or not `su` should propagate certain environment variables. See *su*(1) for more information.
Audit attribute	This attribute controls whether or not users are to be audited. The audit attribute is checked during the login process. See *audit*(5) for more information about HP-UX auditing.
`umask` attribute	This attribute controls `umask()` of all sessions initiated by `pam_unix` or `pam_hpsec`. See *pam_unix*(5) and *pam_hpsec*(5) for more information. The `umask` attribute is checked during the login process.

The system uses these files to process the attributes:

- `/etc/default/security`
- `/var/adm/userdb`
- `/etc/security.dsc`
- `/etc/passwd`
- `/etc/shadow`

Each attribute has a per-user value in only one of these locations: `/etc/password`, `/etc/shadow`, or the user database in `/var/adm/userdb`. Each attribute and its per-user location are explained in the *security*(4) manpage.

The system checks what attributes apply in the following ways:

- The system examines the per-user attribute values in the `/var/adm/userdb` user database, the `/etc/passwd` file, or the `/etc/shadow` file.
- If there is no per-user value, then the system examines the configurable systemwide default attributes in `/etc/default/security`.
- If there are no configurable systemwide default attributes, then the system uses the default attributes in `/etc/security.dsc`.

The security attributes description file, `/etc/security.dsc`, lists the attributes you can define `/etc/default/security` and in the user database in `/var/adm/userdb`. Some attributes are configurable and some are internal. Do not modify the `/etc/security.dsc` file in any way.

2.5.1 Configuring Systemwide Attributes

The following steps explain how to define security attributes on a systemwide basis.

1. Review the *security*(4) manpage, which explains the configurable systemwide default values for attributes. These attributes are configured in the `/etc/default/security` file, which is also explained in the *security*(4) manpage.

   If an attribute is not defined in the `/etc/default/security` file, then the default value defined in the `/etc/security.dsc` file will be used by the system. See the *userdb*(4) manpage for an explanation of the `/etc/security.dsc` file.

2. To change a configurable systemwide default, edit the security defaults file, `/etc/default/security`, with a text editor such as `vi`. The file is world readable and root writable.

   Each line in the `/etc/default/security` file is either a comment or attribute configuration information. Comment lines begin with a pound (#) sign. Noncomment lines are in the form of `attribute=value` pairs, for example, `PASSWORD_MAXDAYS=30`.

2.5.2 Configuring Per-User Attributes

Use the following commands to configure specific attributes for individual users. When you configure per-user attributes, they override the systemwide defaults.

`userdbset`	Changes the attribute for the specified user to override the systemwide default defined in the `/etc/default/security` file. For an example, see Section 2.5.2.1, and see *userdbset*(1M) for more information.
`userdbget`	Displays the user-defined values for a specific user or all users. See *userdbget*(1M) for more information.
`userdbck`	Verifies or fixes the user-defined values. See *userdbck*(1M) for more information.

For example, you can change `PASSWORD_MAXDAYS` from 60 to 30 days only for user `amy`. The password for `amy` is valid for 30 days instead of 60 days. For all other users, the systemwide value of 60 days applies.

Use the following procedure to change an attribute value for a user:

1. Review the *security*(4) manpage, which explains the systemwide attributes and values, and how to set a per-user value. Not all attributes have a per-user value.
2. Review the manpages for the `userdbset`, `userdbget`, and `userdbck` commands.
3. Decide which users to modify and which attributes will apply to them. For example, you might want to have users in an accounting department change their passwords every 30 days and a classroom of students change their passwords every quarter.

4. Use the `userdbset` command to change an attribute for a user.

   The per-user information is stored in a user database in the `/var/adm/userdb` directory. The user database is described in the *userdb*(4) manpage.

   You cannot use the `userdbset` command to configure all attributes. Some per-user values are defined in the `/etc/passwd` and `/etc/shadow` files. For more information, see *security*(4).

5. Use the `userdbget` command to get user information.

2.5.2.1 Examples of Defining User-Specific Attributes with userdbset

In the following example, the `userdbset` command deletes all user-defined attributes for user `joe`. When `joe` logs in, the systemwide defaults in `/etc/default/security` will then apply to `joe`.

```
# /usr/sbin/userdbset -d -u joe
```

Next, `userdbset` sets the minimum password length to `7` and sets `UMASK` to `0022` (octal 022). These changes apply only to `joe`.

```
# /usr/sbin/userdbset -u joe MIN_PASSWORD_LENGTH=7 UMASK=0022
```

In the next example, `userdbset` displays all attributes for user `amy`:

```
# /usr/sbin/userdbget -u amy
amy AUDIT_FLAG=1
amy DISPLAY_LAST_LOGIN=0
```

In the display, the audit flag is enabled and the last login feature is disabled for `amy`.

2.5.2.2 INACTIVITY_MAXDAYS and the Shadow Password File

The `INACTIVITY_MAXDAYS` attribute defined in the `/etc/default/security` file controls whether to expire inactive accounts on a systemwide basis. To override the systemwide default and configure `INACTIVITY_MAXDAYS` on a per-user basis, use the `useradd -f` command or the `usermod -f` command. Use the `userdel` command to delete the per-user configuration. See *useradd*(1M), *usermod*(1M), and *userdel*(1M) manpages for more information.

You cannot use the `userdbset` command to configure the `INACTIVITY_MAXDAYS` on a per-user basis. The `INACTIVITY_MAXDAYS` attribute is related to the inactivity field of the shadow password file. The `useradd` and `usermod` commands modify the inactivity field of the shadow password file for the specified user. See the description of `INACTIVITY_MAXDAYS` in the *security*(4) manpage for more information.

2.5.3 Troubleshooting the User Database

Use the following procedures to troubleshoot the user database.

Problem 1: A user's security attributes seems to be misconfigured. If you suspect that user information is misconfigured in the user database, run the following command:

```
# userdbget -u username
```

The attributes configured for the user `username` are displayed. If an attribute is misconfigured, reconfigure the attribute.

Problem 2: The user database is not functioning properly. If you need to check the user database, enter the following command:

```
# userdbck
```

The `userdbck` command identifies and repairs problems in the user database.

2.6 Handling setuid and setgid Programs

Because they pose a potential security risk to the system, note which programs are `setuid` (set user ID) and `setgid` (set group ID) programs. 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.
- By substituting bogus data for the data created by a program.

Follow these guidelines to secure `setuid` and `setgid` programs:

- Watch for any changes to `setuid` and `setgid` programs.
- Investigate further any programs that appear to be unnecessary `setuid` programs.
- Change the permission of a program that is unnecessarily a `setuid` program to a `setgid` program. See *chmod*(1) and *chmod*(2) for more information.

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

  You can expect to find `setuid` and `setgid` system files, but they should have the same permissions as provided by the factory media, unless you have customized them.

- Do not allow users to normally have `setuid` programs, especially when they use `setuid` to users other than themselves.
- Examine the code of all programs imported from external sources for destructive programs known as Trojan Horses. Never restore or install a `setuid` program for which you have no source to examine.
- To allow users access to certain superuser programs, HP recommends that you use Restricted SMH. Restricted SMH allows non-superusers to access particular areas of SMH. See *smh*(1M) for details.

2.6.1 Why setuid and setgid Programs Can Be Risky

Whenever any program is executed, it creates a process with four ID 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 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 the `setgid` bit is turned on, the privileges of the process are unchanged.
- As a particularly risky case, 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 that violates system security. To a lesser extent, this problem exists in other `setuid` and `setgid` cases as well.

For security reasons, the `setuid` and `setgid` bits on scripts are normally ignored by the HP-UX kernel. This rule can be relaxed by changing the tunable `secure_sid_scripts`, but it is strongly recommended that this tunable be not changed from the default. For more information on this tunable, see *secure_sid_scripts*(5).

2.6.2 How IDs Are Set

IDs are set in these different ways:

- The `ruid` and `rgid` are inherited from the `login` process, which sets your `uid` and `gid`. The `uid` and `gid` values are specified in `/etc/passwd`.
- 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.
- Set the `setuid` and `setgid` bits by using the `chmod` system call or `chmod` command. See *chmod*(1) and *chmod*(2) for more information.

2.6.3 Guidelines for Limiting Setuid Power

Use caution if you add `setuid`-to-`root` programs to an existing system. Adding a `setuid`-to-`root` program changes the system configuration and might compromise security.

Enforce restrictive use of privileged programs through the following administrative and programming recommendations:

- Use `setuid` and `setgid` only when absolutely necessary.
- Make sure that no `setuid` program is writable by others.
- 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 the file systems for new or modified `setuid` and `setgid` programs. You can use the `ncheck -s` command.
- Know exactly what the `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 noncritical 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.
- Make sure that a `setuid` program does not create files writable by anyone other than its intended user.
- Reset the `euid` before an `exec*` system call. Be aware that `exec*` can 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. See *exec*(2), *popen*(3S), and *system*(3S) for more information.
- When writing `setuid` programs, use `setresuid` around the pieces of code that require privileges, to reduce the window of vulnerability. See *setresuid*(2) for more information.
- Close all unnecessary file descriptors before calling `exec*`.
- Ensure that all variables (`PATH`, `IFS`) and the `umask` value 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. See *lockf*(2) and *fcntl*(2) for more information.
- Be especially careful to avoid buffer overruns, for example, by using `sprintf`, `strcpy`, and `strcat` without proper parameter length validation. See *printf*(3S) and *string*(3C) for more information.

2.7 Preventing Stack Buffer Overflow Attacks

The passing of large amounts of data to a program is called a **stack buffer overflow attack**. Usually, the data contains commands that the program is tricked into executing. These attacks are used to gain unauthorized access to the system, to destroy or alter data, or to cause denial of service to legitimate users.

To monitor for stack buffer overflow attacks, watch for the following changes:

- A `setuid` program executing other programs.
- A program unexpectedly gaining a user ID of zero (0). The user ID of zero is for superuser or root only.

To prevent stack buffer overflow attacks:

- Enable the `executable_stack` kernel tunable parameter.
- Use the `chatr +es` command.

The `executable_stack` kernel tunable parameter enables you to prevent a program from executing code from its stack. This guards against an intruder passing illegal data to a program, thereby causing the program to execute arbitrary code from its program stack.

The `executable_stack` kernel tunable parameter globally enables or disables stack buffer overflow protection. A setting of `0` (zero) causes stacks to be nonexecutable and is preferred for security reasons. By default, for backward compatibility, `executable_stack` is set to `1`, which allows stack execution and therefore no protection. Use HP SMH or the `kmtune` command to change the value of `executable_stack`.

An additional way to manage stack buffer overflow protection is to use the `+es` option of the `chatr` command. For example, if `executable_stack` is set to zero but a program does need to execute its stack, use the following `chatr` command to allow stack execution for that program:

```
# chatr -es enable program
```

For more information, see *chatr*(1), *kmtune*(1M), and *executable_stack*(5).

2.8 Protecting Unattended Terminals and Workstations

Unattended workstations and terminals are extremely vulnerable to unauthorized users. Like a front door left unlocked, they are open to anyone. This section explains the following ways to reduce that risk:

- Control access using `/etc/inittab` and run levels. Edit `/etc/inittab` to identify which devices should run at different run levels.
- Protect terminal device files by denying world access to user terminal sessions.
- Configure the screen lock.

2.8.1 Controlling Access Using /etc/inittab and Run Levels

A **run level** is a system state in which a specific set of processes is permitted to run. The processes and default run levels are defined in `/etc/inittab`. Run levels are 0 through 6, s, or S. If a process is not at the same run level as the system, it is terminated. If a process is at the same run level, it is started or it continues to execute.

Following is an example to enable terminals and modems to be run at selected run levels. Both `ttp1` and `ttp2` are at run levels 2 and 3.

```
ttp1:23:respawn:/usr/sbin/getty -h tty0p1 9600
ttp2:23:respawn:/usr/sbin/uugetty -h ttypd0p2 9600
```

Following is an example of changing run levels after normal work hours to disable terminals and modems using a `cron` job. During the day, the run level is 3 and the `ttp1` and `ttp2` terminals can be used because they are at run levels 2 and 3. At 8:00 a.m. from Monday through Friday, the system run level is set to 3:

```
# crontab -e
0 8 * * 1-5 /sbin/init 3
0 17 * * * /sbin/init 4
```

At 5:00 p.m. every day (the `17` in the previous example means 1700 hours or 5:00 p.m.), the system run level is changed to 4. The `ttp1` and `ttp2` terminals cannot operate after 5:00p.m. because they are at run levels 2 and 3.

2.8.2 Protecting Terminal Device Files

If an intruder gains access to an open terminal, they can redirect a command to another terminal window. In the following example, a remove (`rm`) command is redirected to `/dev/tty0p0`:

```
# echo "\r rm -r / \r\033d" > /dev/tty0p0
```

To prevent messages from writing to a terminal, you can use the `mesg -n` (or `mesg n`) command. This command revokes write permissions to users who do not have the appropriate privileges. See *mesg*(1) and *write*(1) for more information.

```
# vi ~/.shrc
mesg n
```

Another way to protect the workstation or terminal is to use the `xhost` command. See *xhost*(1) for more information. The `xhost` command defines the names of hosts and users who are allowed to make connections to the workstation.

```
# xhost +Another.system
```

To allow all systems and users to access the workstation, thereby turning access control off, use the following command:

```
# xhost +
```

2.8.3 Configuring the Screen Lock

This section discusses how to configure the screen lock using the `TMOUT` variable and the CDE lock manager.

2.8.3.1 Configuring the TMOUT Variable

You can configure the `TMOUT` variable to automatically lock inactive terminals.

If you use other systems often and if you copy the `.profile` file from one system to another, then adding the `TMOUT` variable to the `.profile` is more convenient. If you typically stay on one system, then either method of locking the terminal can be used.

To configure the `TMOUT` variable, edit the `.profile` file as shown in the following:

```
# vi ~/.profile
```

`export TMOUT=600 #` (lock after 600 seconds of inactivity)

You can change the 600 to another desired value.

2.8.3.2 Configuring the CDE Lock Manager

You can configure the CDE lock manager to lock your screen after a certain amount of inactive time. To configure the CDE lock manager to lock the screen after 10 minutes of inactive time, enter the following commands:

```
# cp /usr/dt/config/C/sys.resources /etc/dt/config/C/sys.resources
# vi /etc/dt/config/C/sys.resources
dtsession*lockTimeout: 10
```

You can also use the Style Manager task panel to adjust the CDE lock manager. To do this, click on the `screen` icon.

2.9 Protecting Against System Access by Remote Devices

To protect against system penetration by remote access, observe the following precautions:

- Require the use of a hardware dial-back system for all interactive modems.
- Require an additional password from modem users by adding an entry for the modem device in `/etc/dialups` and, optionally, `/etc/d_passwd`. See Section 2.9.1.
- Have users renew their dial-in accounts frequently.
- Cancel system access promptly when a user is no longer an employee.
- Establish a regular audit schedule to review remote usage.
- Connect the modems and dial-back equipment to a single HP-UX system, and allow network services to reach the destination system from that point.
- Make exceptions to dial-back for UUCP access. Additional restrictions are possible through proper UUCP configuration. See *uucp*(1) for more information.

  Another potential exception is file transfer via `kermit`. See *kermit*(1) for more information.
- If a security breach with unknown factors occurs, shut down both network and telephone access and inform the network administrator.
- To maximize security when configuring a dial-back modem system, dedicate the dial-out mechanism to the dial-out function only. Do not configure it to accept dial-in. Use another modem on another telephone line for your dial-in service.

- Keep telephone numbers for modems unlisted and on a different system from other business phones. Do not publicize the dial-in phone numbers.
- Physically secure the modems.
- Use caller ID to identify all incoming calls to the modems.
- Do not allow call forwarding or other extra phone services on the modem lines. Do not use cell phone modems.
- For remote and local access, consider installing an HP-UX AAA server product. Using the industry-standard Remote Authentication Dial-In User Service (RADIUS) protocol, the HP-UX AAA Server provides authentication, authorization, and accounting of user network access at the entry point to a network. See the *HP-UX AAA Server Administrator's Guide* for more information.

2.9.1 Controlling Access Using /etc/dialups and /etc/d_passwd

For additional security in identifying remote users, add entries into the `/etc/dialups` and `/etc/d_passwd` files. These files are used to control the dialup security feature of login. See *dialups*(4) and *login*(1) for more information.

If the `/etc/dialups` file exists, the login process compares the terminal to those listed in `/etc/dialups`. If the terminal exists in `/etc/dialups`, a password is requested by `login`. That password is compared to those in `/etc/d_passwd`.

In addition, the `/etc/passwd` file is used to verify the password.

Following is an example of configuring the `/etc/dialups` file:

`#` **`vi /etc/dialups`** (list the terminals that are allowed)

```
/dev/ttyd0p1
```

```
/dev/ttyd0p2
```

`#` **`vi /etc/d_passwd`**

`/usr/bin/sh`:xxxencrypted-passwordxxxxxxxxx:comments

`/usr/bin/ksh:`xxxencrypted-passwordxxxxxxxx:comments

`/sbin/sh:`xxxencrypted-passwordxxxxxxxxx:comments

The user sees:

```
Login:
Password:
```

```
Dialup password:
```

To change passwords in `/etc/d_passwd`, use the `passwd` command as follows:

`#` **`passwd -F /etc/d_passwd`** ***`shell_path`***

The *`shell_path`* is the shell path listed in `/etc/d_passwd`.

2.10 Securing Login Banners

Login banners are often used to display such system information as the system name, release version, and purpose of the system. This information can help an unauthorized user to learn more about the system. Following are some guidelines for creating more secure login banners:

- Consult the legal department to determine an appropriate message.
- Add a warning to the banner message prohibiting unauthorized use.
- Be consistent in what is displayed in all banners regardless of the login method.

You can modify a banner in the following ways:

- Modify the `login` banner defined in `/etc/copyright` and `/etc/motd`.
- Modify the `telnet` banner defined in`/etc/issue`. The `telnetd -b` banner file command defines a custom banner. To use `/etc/issue` as the login banner, add the following lines to the `/etc/inetd.conf` file:

  ```
  telnet stream tcp nowait root /usr/lbin/telnetd \
  telnetd -b /etc/issue
  ```

  When `inetd` starts `telnetd`, the banner in `/etc/issue` is used. See *inetd*(1M), *telnetd*(IM), and *inetd.conf*(4) for more information.
- Modify the `ftp` banner defined in `/etc/ftpd/ftpaccess`, which is the `ftpd` configuration file. Other displayed messages are defined in `/etc/ftpd/ftpaccess`: greeting, banner, host name, and message. See *ftpdaccess*(4) and *ftpd*(1M) for more information.

Following is an unsecured `telnet` example showing a login banner:

```
# telnet computerAmy
```

The `telnet` login banner shows the release version and machine type. If an unauthorized user tries to use `telnet` to access `computerAmy`, this might be too much information.

Following is a `telnet` example showing a more secure login banner:

```
$ telnet computerMom

Trying...

Connected to computerMom.city.company.com.

Escape character is '^]'.

Local flow control on

Telnet TERMINAL-SPEED option ON

*************************************************************
This is a private system operated for Hewlett-Packard company business. Authorization from HP
management is required to use this system. Use by unauthorized persons is prohibited.
************************************************************

login: Connection closed by foreign host.
```

2.11 Protecting the root Account

Following are suggestions for protecting the root account:

- Do not share the root password.
- Do not use / as the root home directory.
- Examine output from `last -R` and `lastb -R` for unusual or failed root logins and to see who has logged in as root.
- Examine `/var/adm/sulog` for attempts to use the `su root` command.
- Look for unauthorized accounts with a UID of zero (0); use the `logins -d` command.

The following sections discuss how to protect the root account in more detail.

2.11.1 Monitoring root Account Access

If you have two or more system administrators that need root access, following are some suggestions for how to track them:

- Allow only direct root logins on the system console. Create the `/etc/securetty` file with the single entry, `console`, as follows:

  ```
  #echo console > /etc/securetty
  ```

  This restriction applies to all login names that have a UID of zero (0). See *login*(1) for more details.

- Require administrators to use the `su root` command from their personal account to access root. For example:

  ```
  login:me
  $ su root
  password:xxxx
  ```

- Monitor `/var/adm/sulog` to see who has accessed root using `su`.
- Configure a separate root account for each system administrator.

  ```
  # vipw
  root:xxx:0:3::/home/root:/sbin/sh
  root1:xxx:0:3::/home/root1:/sbin/sh
  root2:xxx:0:3::/home/root2:/sbin/sh
  ```

- Monitor each system administrator's history file as follows:

  ```
  #more ~root1/.sh_history
  #more ~root2/.sh_history
  ```

- Monitor successful and failed `su` attempts in `/var/adm/syslog`.

2.11.2 Using the Restricted SMH Builder for Limited Superuser Access

If you need to give limited superuser access to a nonsuperuser, you can activate the Restricted SMH Builder. Using the Restricted SMH Builder, you can enable or disable selected SMH areas for the user. To activate the Restricted SMH Builder, enter:

```
# smh -r
```

When users with restricted access execute SMH, they will have superuser status in the defined areas and will only see those SMH areas in the menu. All other areas of SMH will be hidden from the user. When users without access permissions execute SMH, they will receive an error message stating they must be superuser.

You can also add more applications to SMH and set them up for restricted access.

2.11.3 Reviewing Superuser Access

The `/var/adm/sulog` file logs all attempts of the `su root` command including failures. Successful attempts are flagged with a plus (+) and failures are flagged with a minus (-). Only root can view the `/var/adm/sulog` file. For example:

```
# su root

Password:

# ll /var/adm/sulog

-rw------- 1 root root 690 Aug 17 19:37 /var/adm/sulog
```

In the following example, `userone` has successfully used the `su` command to access root. A second user, `usertwo`, has not been successful. In addition, `usertwo` has not been successful in using `su` to access `gooduser1` either.

```
# more /var/adm/sulog

SU 08/17 19:10 + 0 userone-root

SU 08/17 19:36 - 0 usertwo-root

SU 08/17 19:36 - 0 usertwo-root

SU 08/17 19:36 + 0 userone-root

SU 08/17 19:37 - 0 usertwo-gooduser1
```

3 HP-UX Standard Mode Security Extensions

This chapter describes the HP-UX Standard Mode Security Extensions (HP-UX SMSE). The following topics are discussed:

- Overview (Section 3.1)
- Security attributes and the user database (Section 3.2)

3.1 Overview

HP-UX Standard Mode Security Extensions (HP-UX SMSE) is a group of features that enhances both user and operating system security. HP-UX SMSE includes enhancements or changes to the HP-UX auditing system, passwords, and logins for systems in standard mode. Previously, these features were supported only on systems converted to trusted mode. With HP-UX SMSE, you can use these features on a standard mode system.

NOTE: HP *does not recommend* that you use HP-UX SMSE on systems running in trusted mode. HP-UX SMSE makes available in standard mode many account and password policies currently available only by converting an HP-UX system to trusted mode. Policies configured with HP-UX SMSE are not enforced on systems running in trusted mode.

To determine whether a system has been converted to trusted mode, check for the following file:

```
/tcb/files/auth/system/default
```

If this file exists, the system is running in trusted mode. To convert the system back to standard mode, use the `sam(1M)` command.

Refer to *security*(4) for more information on configurations supported with each of the HP-UX SMSE security features.

HP-UX SMSE offers a new feature, **user database**. Previously, all HP-UX security attributes and password policy restrictions were set on a systemwide basis. The introduction of the user database enables you to set security attributes on a per-user basis that overrides systemwide defaults.

The following trusted mode features are available in standard mode with HP-UX SMSE:

- Audit all users and events on a system
- Display the last successful and unsuccessful user logins
- Lock a user account if there are too many authentication failures
- Display password history
- Expire inactive accounts
- Prevent users from logging in with a null password
- Restrict user logins to specific time periods

- Usage of the `userdbset` command can be restricted based on a user's authorizations. See *userdbset*(1M) for more information.
- The `userstat` command displays the account status of local users. It checks the status of local user accounts and reports abnormal conditions, such as account locks. See *userstat*(1M) for more information.

3.2 Security Attributes and the User Database

Previously, in standard mode, all HP-UX security attributes and password policy restrictions were set on a systemwide basis. The introduction of the user database enables you to set security attributes on a per-user basis, which override systemwide defaults.

3.2.1 System Security Attributes

A security attribute defines how to control security configurations, such as passwords, logins, and auditing. The security attributes description file, `/etc/security.dsc`, lists the attributes that can be defined either in `/etc/default/security`, in the user database in `/var/adm/userdb`, or in both files. Some attributes are configurable and some are internal.

CAUTION: Do not modify the `/etc/security.dsc` file in any way.

When a user logs in, the system checks for applicable security attributes in the following order:

1. The system examines per-user attributes in the following locations:
   - `/var/adm/userdb`
   - `/etc/passwd`
   - `/etc/shadow`

   NOTE: For each per-use attribute, a value is stored in one of the three files above. Refer to *security*(4) to see which attributes are stored in each file.
2. If there is no per-user value, then the system examines the configured systemwide attributes in `/etc/default/security`.
3. If there are no configured systemwide attributes, then the system uses the default attributes in `/etc/security.dsc`.

3.2.2 Configuring Systemwide Attributes

To configure systemwide attributes, follow these steps:

1. Plan your configuration using available resources. Refer to *security*(4) for information about configuring systemwide attributes.

2. To change a systemwide default, edit the `/etc/default/security` file with a text editor such as `vi`. Comments begin with a pound sign (#). Attributes are written in `attribute=value` format.

   For example, to set the systemwide minimum number of uppercase characters in a password to two (2), enter the following values into `/etc/default/security`:

   ```
   PASSWORD_MIN_UPPER_CASE_CHARS=2
   ```

NOTE: Changes to systemwide security attributes do not take effect immediately. Password attributes take effect the next time users change their passwords. Login attributes take effect the next time users log in.

3.2.3 User Database Components

The user database feature of HP-UX SMSE includes files, commands, manpages, and per-user attributes you can apply to specific users on your HP-UX system. All these elements of the user database are described in the following sections.

3.2.3.1 Configuration Files

Table 3-1 briefly describes the files you use with the user database.

Table 3-1 User Database Configuration Files

File	Description
`/var/adm/userdb`	Stores most per-user information.

3.2.3.2 Commands

Table 3-2 briefly describes the commands you can use to modify and administer entries in the user database.

Table 3-2 User Database Commands

Command	Description
`userdbset`	Changes attribute values configured in the user database.
`userdbget`	Displays attribute values configured in the user database.
`userdbck`	Verifies the integrity of the information in the user database.
`userstat`	Reports the status of local user accounts.

3.2.3.3 Attributes

The following security attributes are available for individual users:

Table 3-3 User Attributes

Attribute	Description
`ALLOW_NULL_PASSWORD`	Allows or denies login with a null password.
`AUDIT_FLAG`	Audits or stops auditing the user.
`AUTH_MAXTRIES`	Defines the number of login failures allowed before a user is locked out of the system.
`DISPLAY_LAST_LOGIN`	Displays information about the user's last login.
`LOGIN_TIMES`	Restricts login time periods.
`MIN_PASSWORD_LENGTH`	Defines the minimum password length.
`NUMBER_OF_LOGINS_ALLOWED`	Defines the number of simultaneous logins allowed per user.
`PASSWORD_HISTORY_DEPTH`	Defines the password history depth.
`PASSWORD_MIN_LOWER_CASE_CHARS`	Defines the minimum number of lowercase characters required in a password.
`PASSWORD_MIN_UPPER_CASE_CHARS`	Defines the minimum number of uppercase characters required in a password.
`PASSWORD_MIN_DIGIT_CHARS`	Defines the minimum number of digit characters required in a password.
`PASSWORD_MIN_SPECIAL_CHARS`	Defines the minimum number of special characters required in a password.
`UMASK`	Defines the umask for file creation.

NOTE: The previous list contains only security attributes that can be configured in the user database. For a complete list of HP-UX system security attributes, refer to *security*(4).

3.2.3.4 Manpages

Table 3-4 briefly describes the manpages you use with the user database.

Table 3-4 User Database Manpages

Manpage	Description
userdb(4)	Provides an overview of the use of the user database.
userdbset(1M)	Describes `userdbset` functionality and syntax.
userdbget(1M)	Describes `userdbget` functionality and syntax.
userdbck(1M)	Describes `userdbck` functionality and syntax.
userstat(1M)	Describes the `userstat` functionality and syntax.

3.2.4 Configuring Attributes in the User Database

In previous HP-UX systems, security attributes and password policy restrictions were set a systemwide basis. With HP-UX SMSE, you can configure some security attributes on a per-user basis. Attributes configured per-user override systemwide configured attributes.

To modify a user's attribute values, follow these steps:

1. Decide which users to modify and which attributes will apply to them.

   For example, you want user joe to be able to log in to the system only from 8am to 5pm on Mondays.

2. Change the attributes using the `userdbset` command as follows:

   ```
   # userdbset -u user-name attribute-name=attribute-value
   ```

   For example, to specify that user joe can log in to the system only from 8am to 5pm, enter:

   ```
   # userdbset -u joe LOGIN_TIMES=Mo0800-1700
   ```

3.2.5 Troubleshooting the User Database

Use the following procedures to troubleshoot the user database.

Problem 1: A user's security attributes seems to be misconfigured. If you suspect that user information is misconfigured in the user database, run the following command:

```
# userdbget -u username
```

The attributes configured for the user username are displayed. If an attribute is misconfigured, reconfigure the attribute. Refer to "Configuring Attributes in the User Database" for instructions.

Problem 2: The user database is not functioning properly. If you need to check the user database, run the following command:

```
# userdbck
```

The `userdbck` command identifies and repairs problems in the user database.

4 Remote Access Security Administration

HP-UX provides several remote access services, such as file transfer, remote login, remote command execution, management of IP addresses and network clients, routing protocols, mail exchange, network services, and a security mechanism spawned by `inetd`, the Internet super daemon.

This chapter discusses the following topics:

- Overview of internet services and remote access services (Section 4.1)
- The inetd Daemon (Section 4.2)
- Protection against spoofing with TCP wrappers (Section 4.3)
- Secure internet services (Section 4.4)
- Controlling an administrative domain (Section 4.5)
- Securing remote sessions using HP-UX Secure Shell (SSH) (Section 4.6)

4.1 Overview of Internet Services and Remote Access Services

This section provides brief descriptions of the authentication or authorization mechanism used by various Internet Services, and the security risks.

For more information, see the *HP-UX Internet Services Administrator's Guide* and *Using HP-UX Internet Services*:

http://www.hp.com/go/hpux-networking-docs

Click **HP-UX 11i v3 Networking Software**.

The HP-UX Internet Services provides authentication, either through password verification or authorization that is set up in a configuration file. See Table 4-1 for a list of Internet Services components and their access verification or authorization mechanism.

Table 4-1 Internet Services Components and Access Verification, Authorization, and Authentication

Internet Services Component	Access Verification, Authorization, or Authentication Mechanism
`ftp` (file transfer)	Password verification. Also can use Kerberos authentication mechanism defined in `/etc/inetsvcs.conf`. See *ftp*(1).
`rcp` (remote copy)	Entry in `$HOME/.rhosts` or `/etc/hosts.equiv` file. Also can use Kerberos authentication mechanism defined in `/etc/inetsvcs.conf`. See *rcp*(1).
`rdist` (remote file distribution)	Entry in `$HOME/.rhosts` or `/etc/hosts.equiv` file. See *rdist*(1).
`remsh`, `rexec` (execute from remote shell)	Entry in `$HOME/.rhosts` or`/etc/hosts.equiv` file. Also can use Kerberos authentication mechanism defined in `/etc/inetsvcs.conf`. See *remsh*(1).

Table 4-1 Internet Services Components and Access Verification, Authorization, and Authentication *(continued)*

Internet Services Component	Access Verification, Authorization, or Authentication Mechanism
`rlogin` (remote login)	Password verification or entry in `$HOME/.rhosts` or `/etc/hosts.equiv` file. Also can use Kerberos authentication mechanism defined in `/etc/inetsvcs.conf`. See *rlogin*(1).
`telnet` (remote login using TELNET protocol)	Password verification. If the TAC User ID option is enabled by the `telnetd` daemon, `telnet` uses `$HOME/.rhosts` or `/etc/hosts.equiv` file. See *telnet*(1) and *telnetd*(1M).

NOTE: Information (including passwords) is passed between two systems in clear text and is not encrypted. Use Internet Services only between hosts that are well-known and defined to each other and within a private internal network behind a firewall. When communicating over an untrusted network, secure the communications using IPSec or Kerberos

Remote Access Services connect remote systems in a network. By default, the remote access services function in a nonsecure environment. To function in a secure environment, enable the Kerberos V5 network authentication. In a nonsecure environment, you must have a login name and password to access a remote system, and the login name is not checked for authentication and authorization. In a secure environment, you need not have a login name and password. When you attempt to connect to a remote system, the Kerberos protocol checks if the user is allowed to access the remote system.

4.1.1 Securing ftp

An unauthorized user might try to gain access to a system by using the `ftp` command. Following are some suggestions to prevent this problem:

- Enable `ftp` logging in `/etc/inetd.conf` by using the `ftpd -l` command.
- Review the `ftp` logs in `/var/adm/syslog/syslog.log` and `/var/adm/syslog/xferlog` for unusual remote access attempts.

  See *syslogd*(1M) and *xferlog*(5).
- Deny `ftp` access to `guest`, `root`, and other accounts by listing them in `/etc/ftpd/ftpusers`.

  See *ftpusers*(4).
- Regularly search and remove users' `~/.netrc` files. The `.netrc` file contains login, password, and account information used by the `ftp` autologin process, by the `rexec()` library routine, and by the `rexec` command.

  See *netrc*(4).

4.1.2 Securing Anonymous ftp

If a `$HOME/.rhosts` file is put into `/home/ftp`, then an unauthorized user could use `rlogin` to log in as the user, `ftp`. The `.rhosts` file specifies hosts and users that are allowed access to a local account using `rcp`, `remsh`, or `rlogin` without a password. For more information, see *hosts.equiv*(4).

Following are some suggestions to making anonymous `ftp` more secure:

- Make sure that neither `/home/ftp` nor any of its children is writable:

  ```
  $chmod -R a -w /home/ftp
  ```

- Make sure that the `ftp` entry in `/etc/passwd` is correctly configured:

  ```
  ftp:*:500:100:Anonymous FTP user:/var/ftp:/usr/bin/false
  ```

- Make sure that all passwords in `~ftp/etc/passwd` are asterisks (`*`):

  ```
  $more ~ftp/etc/passwd
  root:*:0:3::/:/usr/bin/false daemon:*:1:5::/:/usr/bin/false
  ```

- If you must have a writable `pub` directory, use 1733 permissions:

  ```
  $chmod 1733 /home/ftp/pub
  ```

- Use disk quotas or a `cron` job to control the size of `/home/ftp/pub`:

  ```
  0 1 * * * find /home/ftp/pub/* -atime +1 exec rm -rf {} \;
  ```

- Check anonymous `ftp` activity in `/var/adm/syslog/syslog.log`:

  ```
  $tail /var/adm/syslog/syslog.log
  ```

4.1.3 Denying Access Using /etc/ftpd/ftpusers

The `inetd` daemon runs the file transfer protocol server, `ftpd`, when a service request is received at the port indicated in `/etc/services`. The `ftpd` server rejects remote logins to local user accounts which are listed in `/etc/ftpd/ftpusers`. These user accounts are known as restricted accounts. See *ftpd*(1M), *privatepw*(1), and *services*(4).

In the `/etc/ftpd/ftpusers` file, each restricted account name must appear by itself on a line. Also add user accounts with restricted login shells that are defined in `/etc/passwd`, because `ftpd` accesses local accounts without using their login shells.

If `/etc/ftpd/ftpusers` does not exist, `ftpd` does not perform a security check. For more information, see *ftpusers*(4).

On HP-UX 11i, the `ftpd` daemon is based on WU-FTPD. WU-FTPD is the HP implementation of the `ftpd` daemon developed at Washington University. WU-FTPD includes increased access control, enhanced logging capabilities, virtual hosts support, and RFC1413 (Identification Protocol) support.:

For more information, see the *HP-UX Remote Access Services Administrator's Guide*:

http://www.hp.com/go/hpux-networking-docs

Click **HP-UX 11i v3 Networking Software**.

4.1.4 Other Security Solutions for Spoofing

Spoofing is a method of pretending to be a valid user or host to gain unauthorized access to a system. Because IP addresses and hostnames can be spoofed, using the `/var/adm/inetd.sec` security file for `inetd` (the internet daemon) is not a guaranteed security solution. See Section 4.2 for information about `inetd`.

The following security features or products are alternative security solutions:

- IPFilter is a TCP/IP packet filter suitable for use as a system firewall to protect application servers.

  For information on HP-UX IPFilter, see the *HP-UX IPFilter Administrator's Guide*:

  www.hp.com/go/hpux-security-docs

  Click **HP-UX IPFilter Software**.

- TCP Wrappers provides a TCP wrapper daemon, `tcpd`, that is invoked by `inetd` to provide additional security. For more information, see Section 4.3 and the *HP-UX Internet Services Administrator's Guide*:

  http://www.hp.com/go/hpux-networking-docs

  Click **HP-UX 11i v3 Networking Software**.

- Secure Internet Services allows use of Kerberos authentication and authorization for `ftp`, `rcp`, `remsh`, `rlogin`, and `telnet`. Instead of user passwords, encrypted Kerberos authentication records transmit over the network. For more information, see the following:
  - Section 4.4
  - *Installing and Administering Internet Services*:

    http://www.hp.com/go/hpux-networking-docs

    Click **HP-UX 11i v3 Networking Software**.
  - *Configuration Guide for Kerberos Client Products on HP-UX*:

    www.hp.com/go/hpux-security-docs

    Click **HP-UX Kerberos Data Security Software**.

- IPSec, an IP security protocol suite, provides security for IP networks such as data integrity, authentication, data privacy, application-transparent security, and encryption.

  For information on HP-UX IPSec, see the *HP-UX IPSec Administrator's Guide*:

  www.hp.com/go/hpux-security-docs

  Click **HP-UX IPSec Software**.

4.2 The inetd Daemon

The Internet daemon, `/usr/sbin/inetd`, is the master server for many Internet Services.

The `inetd` daemon is usually started automatically by the `/sbin/init.d/inetd` script as part of the boot process.

The `inetd` daemon monitors for connection requests for the services listed in the `/etc/inetd.conf` configuration file, and spawns the appropriate server on receiving a request. In other words, users connect to remote systems by using an Internet Service, such as `telnet`. The `inetd` daemon determines if a `telnet` connection from the host is allowed before completing the connection. The host information for allowing or denying access is in the `/var/adm/inetd.sec` file.

The `inetd` daemon works as follows:

1. Starts at run level 2 during system boot. (if the following command is in the system startup script: `/sbin/init.d/inetd start`)
2. Checks `/etc/inetd.conf` to determine which services to provide. For more information, see *ftp*(1) and *inetd.conf*(4).
3. Checks `/etc/services` to determine which ports to monitor for the services listed in `/etc/inetd.conf`. The `/etc/services` file maps service names to port numbers. For more information, see *services*(4).
4. Receives an Internet Service connection request from a client. For example, someone runs `telnet`.
5. Consults `/var/adm/inetd.sec` to determine if the client is permitted access. For more information, see *inetd.sec*(4).
6. Logs the request in `/var/adm/syslog/syslog.log` if logging is enabled. For more information, see *syslogd*(1M).
7. If `inetd` refuses the connection for security reasons, the connection is shut down.
8. If the connection request is valid, `inetd` starts a server process to handle the valid connection request. The server process can have other security features in addition to `inetd`.

4.2.1 Securing inetd

The `/etc/inetd.conf` file is the `inetd` configuration file, which lists the services that the `inetd`daemon can start. Each service listed in `/etc/inetd.conf` must also appear in the `/etc/services` file. The `/etc/services` file maps service names to port numbers. Each port number has an associated protocol name, such as `tcp` or `udp`. Every entry for a protocol must have a matching entry in the `/etc/protocols` file.

The following suggestions can make `inetd` more secure:

- Enable `inetd` logging in `/etc/rc.config.d/netdaemons`. For more information, see *rc.config.d*(4).
- Review `/etc/inetd.conf` and `/etc/services` for changes. An unauthorized user might have gained root access and modified the `/etc/services` and `/etc/inetd.conf` files. In `/etc/inetd.conf`, look for names of services you are not using. In `/etc/services`, look for port numbers that are not registered with the

Internet Assigned Numbers Authority (IANA) at http://www.iana.org. Verify that the port numbers listed for Internet Services match port numbers registered with IANA.

- Comment out unnecessary services, such as `finger`, in `/etc/inetd.conf`. The `finger` command displays user information without needing a password.
- Comment out Remote Procedure Calls (RPC) services in `/etc/inetd.conf`.
- Comment out `inetd` "internal trivial" services in `/etc/inetd.conf` to avoid denial-of-service attacks. A malicious user might overload `inetd` with `chargen` (character generator) requests. For more information, see *inetd*(1M) and *inetd.conf*(4).

4.2.1.1 Denying or Allowing Access Using /var/adm/inetd.sec

In addition to configuring the `/etc/inetd.conf` file, you can configure an optional security file called `/var/adm/inetd.sec` to restrict access to the services started by `inetd`. The `/var/adm/inetd.sec` file lists which hosts are allowed or denied access to each service. For more information, see *inetd.conf*(4).

For example:

```
login allow 10.3-5 192.34.56.5 ahost anetwork
```

```
login deny 192.54.24.5 cory.example.edu.testlan
```

4.3 Protection Against Spoofing with TCP Wrappers

Transmission Control Protocol (TCP) Wrappers provide enhanced security for services spawned by `inetd`. TCP Wrappers are an alternative to using `/etc/inetd.sec`. TCP Wrappers provide protection against host name and host address spoofing. Spoofing is a method of pretending to be a valid user or host to gain unauthorized access to a system.

To prevent spoofing, TCP Wrappers uses Access Control Lists (ACLs). The ACLs are lists of systems in the `/etc/hosts.allow` and `/etc/hosts.deny` files. TCP Wrappers provide some protection against IP spoofing when configured to verify host name to IP address mapping and to reject packets with IP source routing.

However, TCP Wrappers do not provide cryptographic authentication or data encryption. Like `inetd`, information is passed in clear text.

TCP Wrappers are part of the HP-UX Internet Services software. For more information, see the *HP-UX Internet Services Administrator's Guide*:

http://www.hp.com/go/hpux-networking-docs

Click **HP-UX 11i v3 Networking Software**.

You can also see the following manpages:

tcpd(1M), *tcpdmatch*(1), *tcpdchk*(1), *tcpd.conf*(4), *hosts_access*(3), *hosts_access*(5), and *hosts_options*(5).

When you enable TCP Wrappers, `inetd` runs a TCP wrapper daemon, `tcpd`, instead of running the requested service directly. The TCP Wrappers work as follows:

1. Clients send connection requests to `inetd` as they normally do, for example, `telnet`.
2. Instead of invoking the server process, `inetd` calls the TCP Wrapper daemon (`tcpd`).
3. The TCP Wrapper daemon determines the validity of the client's connection request. The `tcpd` daemon logs the request and checks the access control files (`/etc/hosts.allow` and `/etc/hosts.deny`).
4. If the client is valid,`tcpd` calls the appropriate server process.
5. The server process processes the client's request, for example, the `telnet` connection completes.

4.3.1 Additional Features of TCP Wrappers

You can also define configuration parameters in the `/etc/tcpd.conf` configuration file, such as logging behavior, user name lookups, and reverse look up failure behavior. The `tcpd` daemon reads this configuration file to look for configuration parameters during run time.

You can configure the `/etc/hosts.allow` and `/etc/hosts.deny` files for other security features, such as trap setting and banner message.

The trap setting feature of TCP Wrappers enables you to trigger appropriate action on the host depending upon the number of denied connection attempts from a remote host.

The banner message feature causes a message to be sent to the client when an ACL rule is included in an access control file.

4.3.2 TCP Wrappers Do Not Work with RPC Services

TCP Wrappers do not work with remote procedure call (RPC) services over TCP. These services are registered as `rpc` or `tcp` in the `/etc/inetd.conf` file. The only important service that is affected by this limitation is `rexd`, used by the `on` command.

4.4 Secure Internet Services

Secure Internet Services (SIS) is an optionally enabled mechanism that incorporates Kerberos V5 authentication and authorization for remote access services: `ftp`, `rcp`, `remsh`, `rlogin`, and `telnet`.

Secure Internet Services is part of the HP-UX Internet Services product, which is documented in *Using HP-UX Internet Services*:

http://www.hp.com/go/hpux-networking-docs

Click **HP-UX 11i v3 Networking Software**.

You can also see the following manpages:

sis(5), *kinit*(1), *klist*(1), *kdestroy*(1M), *krbval*(1M), *k5dcelogin*(1M), *inetsvcs_sec*(1M), and *inetsvcs*(4).

When you run SIS commands, the security is enhanced because you no longer have to transmit a password in readable form over the network.

NOTE: The SIS libraries do not encrypt the session beyond what is necessary to authorize you or to authenticate the service. Therefore, these services do not provide integrity checking or encryption services on the data or on remote services. To encrypt the data, use OpenSSL. For more information, see the *OpenSSL Release Notes*:

www.hp.com/go/hpux-security-docs

Click **HP-UX OpenSSL Software**.

When two systems are operating in a Kerberos V5-based secure environment, Secure Internet Services ensures that a local and remote host are identified to each other in a secure and trusted manner and that the user is authorized to access the remote account.

For `ftp/ftpd`, `rlogin/rlogind`, and `telnet/telnetd`, the Kerberos V5 authentication mechanism sends encrypted tickets instead of a password over the network to verify and to identify the user. For `rcp/remshd` and `remsh/remshd`, the secure versions of these services ensure that the user is authorized to access the remote account.

4.5 Controlling an Administrative Domain

All network administration programs should be owned by a protected, network-specific account, such as `uucp`, `nso`, or by a daemon, instead of by root.

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 that system users have already been verified by their host system. Use the following steps to identify and control an administrative domain:

1. List the nodes to which you export file systems in `/etc/exports`. The `/etc/exports` file contains entries of a file system path name and a list of systems or groups of systems that are allowed access to the file system. The `/etc/exports` entries might contain names of groups of systems. You can find out what individual systems are included in a group by checking `/etc/netgroup`.
2. List the nodes that have equivalent password databases in `/etc/hosts.equiv`.
3. Verify that each node in the administrative domain does not extend privileges to any nodes that are not included. Repeat steps 2 and 3 for each node in the domain.
4. Control root and local security on every node in the administrative domain. A user with superuser privileges on any machine in the domain can acquire those privileges on every machine in the domain.

5. Maintain consistency of user name, `uid`, and `gid` among password files in the administrative domain.
6. Maintain consistency among any group files on all nodes in the administrative domain. For example, to check consistency with the `hq` and `mfg` systems, if the root file system of the `mfg` system is remotely mounted to `hq` as `/nfs/mfg/`, enter the following `diff` command:

   ```
   $diff /etc/group /nfs/mfg/etc/group
   ```

   If any differences are displayed, the two `/etc/group` files are inconsistent and they should not be.

4.5.1 Verifying Permission Settings on Network Control Files

The network control files in the `/etc` directory are security targets because they provide access to the network itself. Network control files should never be writable by the public.

Set the modes, owners, and groups on all system files carefully. Check these files regularly for any changes and correct any changes.

The most commonly used network control files are as follows:

- `/etc/exports`

  List of file directories that can be exported to NFS clients. For more information, see *exports*(4).

- `/etc/hosts`

  List of network hosts and their IP addresses. For more information, see *hosts*(4).

- `/etc/hosts.equiv`

  List of remote hosts that are allowed access and that are equivalent to the local host. For more information, see *hosts.equiv*(4).

- `/etc/inetd.conf`

  Internet Services configuration file. For more information, see*inetd.conf*(4).

- `/etc/netgroup`

  List of networkwide groups. For more information, see*netgroup*(4).

- `/etc/networks`

  List of network names and their network numbers. For more information, see *networks*(4).

- `/etc/protocols`

  List of protocol names and numbers. For more information, see *protocols*(4).

- `/etc/services`

  List of official service names and aliases with the port number and protocol that the services use. For more information, see *services*(4).

4.6 Securing Remote Sessions Using HP-UX Secure Shell (SSH)

HP-UX Secure Shell is based on the OpenSSH product, an open source SSH product (http://www.openssh.org). It enables a secure connection between a client and a remote host over an otherwise insecure network. Following are the key attributes of this secure connection:

- Strong authentication for both client and the remote host.
- Strong encryption and public key cryptography for communication between a client and the remote host.
- A secure channel for the client to use to execute commands on the remote host.

HP-UX Secure Shell offers a secure replacement for such commonly used functions and commands as `telnet`, `remsh`, `rlogin`, `ftp`, and `rcp`.

For HP-UX Secure Shell documentation see the *ssh*(1) manpage for the `ssh` client and to the *sshd*(8) manpage for the `sshd` server. Both manpages include references to the other HP-UX Secure Shell manpages that come with the product.

Also see the *HP-UX Secure Shell Release Notes*:

www.hp.com/go/hpux-security-docs

Click **HP-UX 11i Secure Shell Software**.

4.6.1 Key Security Features of HP-UX Secure Shell

The key security features of HP-UX Secure Shell include the following:

- Strong encryption

  All communication between the client and the remote host is encrypted using patent-free encryption algorithms, such as Blowfish, 3DES, AES, and arcfour. Authentication information, such as passwords, is never sent in clear text across the network. Encryption in conjunction with strong public key-based cryptography also provides protection against potential security attacks.

- Strong authentication

  HP-UX Secure Shell supports a strong set of authentication methods between client and server. The authentication can be two-way: the server authenticates the client, and the client authenticates the server. This protects the session against a variety of security issues. The supported authentication methods are described Section 4.6.5.

- Port forwarding

  The redirection of TCP/IP connections between a client and a remote host (and back) is referred to as **port forwarding** or **SSH tunneling**. HP-UX Secure Shell supports port forwarding. For example, `ftp` traffic between a client and a server (or email traffic between an email client and a POP/IMAP server) can be redirected using port forwarding. Instead of the client directly communicating with its server, the traffic can be redirected to an sshd server over a secure channel, and the sshd server can then forward the traffic to a designated port on the real server machine.

- Integration with underlying HP-UX security features.

  The HP-UX Secure Shell product is integrated with important HP-UX security features. For more information, see Section 4.6.7.

4.6.2 Software Components of HP-UX Secure Shell

HP-UX Secure Shell software consists of a set of client and server components. See Table 4-2.

Table 4-2 Software Components of HP-UX Secure Shell

Component	Description	Location	Equivalent non-secure component(s)
`ssh`	Secure Shell client is a secure replacement for `telnet` and `remsh`; it is most similar to `remsh` with security features	Client	`remsh`, `telnet`, `rlogin`
`slogin`	Symbolic link to `ssh`	Client	`remsh`, `telnet`, `rlogin`
`scp`	Secure copy client and secure copy server	Client and server	`rcp`
`sftp`	Secure `ftp` client	Client	`ftp`
`sshd`	Secure shell daemon	Server	`remshd`, `telnetd`, `rlogind`
`sftp-server`	Secure `ftp` daemon	Server	`ftpd`
`ssh-rand-helper`	Random number generator, which is used when `sshd` is not able to find `/dev/random` or `/dev/urandom` on the server. HP-UX is shipped with a kernel-resident random number generator, `rng`. If `rng` is deconfigured, `sshd` uses `prngd`.	Server	Not applicable
`ssh-agent`	Tool for "automatic" key-based login from client to server	Client and server	`rhosts` file mechanism
`ssh-add`	Tool for making key pairs of the client known to `ssh-agent`	Client	Not applicable
`ssh-keygen`	Tool for generating key pairs for public key authentication	Client	Not applicable

Table 4-2 Software Components of HP-UX Secure Shell *(continued)*

Component	Description	Location	Equivalent non-secure component(s)
`ssh-keyscan`	Tool for a client to gather the public keys for a set of hosts running the Secure Shell daemon (`sshd`)	Client	Not applicable
`ssh-keysign`	Tools to generate the digital signature required during host based authentication is and it is used by ssh() to access the local host keys host based authentication	Client	Not applicable

4.6.3 Running HP-UX Secure Shell

Before running any of the Secure Shell clients listed in Table 4-2, first start the Secure Shell server daemon, `sshd`. The `sshd` daemon obtains its initial configuration values from the `sshd_config` file, located in the `/opt/ssh/etc` directory on the server system. One of the most important configuration directives in `sshd_config` is the set of authentication methods supported by the `sshd` daemon. See Section 4.6.5 for more information.

4.6.3.1 Running the ssh Client

The `ssh` client application establishes a socket connection with the `sshd` server. The `sshd` server spawns a child `sshd` process. This child inherits the connection socket and authenticates the client based on the selected authentication method. A successful secure client session is established only upon successful authentication.

After a session is created, all subsequent communication occurs directly between the client and this child `sshd` process. The client can now execute remote commands on the server. Each command request from the `ssh` client causes the child `sshd` process to spawn a shell process to execute that command.

In summary, a running `ssh` client-server session consists of the following processes:

- On every client system connected to the `sshd` server, there is one `ssh` client process for each `ssh` connection currently established from that client system.
- On the server system, there is one parent `sshd` process and as many child `sshd` processes as there are concurrent `ssh` clients connected to the server. The number of child `sshd` processes running on the server doubles if privilege separation is enabled on the server. See Section 4.6.4.
- On the server system, for each command execution request from a `ssh` client, the corresponding child `sshd` process spawns a shell process, and uses a UNIX pipe to communicate the command request to this shell process. This shell process returns the command execution results to the child `sshd` process using the UNIX pipe and terminates when the command execution is complete.

4.6.3.2 Running the sftp Client

The `sftp` client application causes the `sftp` client process to spawn the `ssh` client, and then communicates with it using a UNIX pipe. The `ssh` client then establishes a socket connection with the `sshd` server.

The rest of the server interaction is similar to the `ssh` client case described in Section 4.6.3.1. The difference is that instead of spawning a shell to execute the remote command, the child `sshd` process spawns the `sftp-server` process. All subsequent communication during this `sftp` session occurs among the following processes:

- The `sftp` client and the `ssh` client, on the client system, using a UNIX pipe.
- The `ssh` client and the child `sshd` process, over the established connection socket.
- The child `sshd` process and the `sftp` server process, using a UNIX pipe.

4.6.3.3 Running the scp Client

The `scp` client case is almost identical with the `sftp` client execution. The difference is that instead of spawning the `sftp-server` process, the child `sshd` process spawns the `scp` process. All subsequent communication during the `scp` session occurs among the following processes:

- The `scp` client and the `ssh` client, on the client system using a UNIX pipe.
- The `ssh` client and the child `sshd` process, over the established connection socket.
- The child `sshd` process and the `scp` server process, using a UNIX pipe.

4.6.4 HP-UX Secure Shell Privilege Separation

HP-UX Secure Shell offers a more enhanced level of security through the **privileged separation** feature. As described in Section 4.6.3, both the parent `sshd` and the child `sshd` processes run as privileged users. When privilege separation is enabled, one extra process is spawned per user connection.

When an `ssh` client connects to an `sshd` server which is configured for privilege separation, the parent `sshd` process spawns a privileged child `sshd` process. When privilege separation is enabled, the child `sshd` process spawns an additional nonprivileged child `sshd` process. This nonprivileged child `sshd` process then inherits the connection socket. All subsequent communication between client and server occurs with this nonprivileged child `sshd` process.

Most remote command execution requests from the client are nonprivileged, and are handled by a shell spawned under this nonprivileged child `sshd` process. When the nonprivileged child `sshd` process needs a privileged function to be executed, it communicates with its privileged parent `sshd` process using a UNIX pipe.

Privilege separation helps contain potential damage from an intruder. For example, if a buffer overflow attack occurs during a shell command execution, control is within the nonprivileged process, thereby containing the potential security risk.

NOTE: Privilege separation is the default configuration for HP-UX Secure Shell. You can turn off privilege separation by setting `UsePrivilegeSeparation NO` in the `sshd_config` file. Because of the potential security risk, turn off privilege separation only after careful consideration.

4.6.5 HP-UX Secure Shell Authentication

HP-UX Secure Shell supports the following authentication methods:

- GSS-API (Kerberos-based client authentication)
- Public key authentication
- Host-based authentication
- Password authentication

When a client connects with a remote `sshd` daemon, it selects the desired authentication method (one of the methods listed previously), and either presents the appropriate credentials as part of the connection request or responds to a prompt sent back by the server. All authentication methods work in this way.

The server requires the appropriate key, pass phrase, password, or credentials from the client to establish a successful connection.

You can choose to have the `sshd` instance support only a subset of the supported authentication methods based on security requirements.

Although HP-UX Secure Shell supports the authentication methods listed previously, system administrators can limit the authentication methods offered by an sshd instance, based on the specific security requirements of their environment. For example, an HP-UX Secure Shell environment can dictate that all clients must authenticate using the public key or Kerberos methods. As a result, may disable the remaining methods. The enabling and disabling of supported authentication methods is through configuration directives specified in the `sshd_config` file.

When an `ssh` client connection request is made, the server first responds with its list of supported authentication methods. This list represents the authentication methods supported by the `sshd` server and the sequence in which these methods will be tried. The client can omit one or more of those authentication methods. The client can also change the sequence in which the methods are attempted. You achieve this with a configuration directive in the client configuration file, `/opt/ssh/etc/ssh_config`.

The authentication methods supported by HP-UX Secure Shell are summarized in the following sections.

4.6.5.1 GSS-API

With the Generic Security Service application Programming Interface (GSS-API), a Kerberos-based client authentication, the client must obtain Kerberos credentials in advance, and also have a Kerberos configuration file present in the appropriate client directory. When a client connects with an `sshd` daemon, it presents its credentials at connection time. The server matches these credentials with its copy of credentials for this specific user. Also, the server can optionally establish the legitimacy of the client's host environment.

For more information, see *gssapi*(5), *kerberos*(9) and the HP-UX Kerberos Data Security documentation:

www.hp.com/go/hpux-security-docs

Click **HP-UX Kerberos Data Security Software**.

4.6.5.2 Public Key Authentication

For public key authentication, the Secure Shell environment must have the following setup:

- Both the client and server must have a key pair. Every `ssh` client and every `sshd` server must generate a key pair for themselves using the `ssh-keygen` utility.
- The client must make its public key known to all `sshd` servers it needs to communicate with. Do this by copying every client's public key into a predetermined directory on every relevant server.
- The client must acquire the public key for every server it needs to communicate with. The client acquires the public keys using the `ssh-keyscan` utility.

After this setup is completed, `ssh` clients connecting to `sshd` servers are authenticated using public and private keys. For more information on public key cryptography, see *public key cryptography*.

HP-UX Secure Shell offers an additional feature for streamlining public key authentication. For some environments, you might want the convenience of not having to respond to password prompts all the time. You can eliminate the need to respond to password prompts by using a combination of the `ssh-agent` and `ssh-add` processes, both running on the client machine. The client registers all its key information with the `ssh-agent` process through the `ssh-add` utility. Then, public key authentication between client and server is facilitated by `ssh-agent` without the `sshd` daemon having to interact with the client.

4.6.5.3 Host-Based and Public Key Authentication

Host-based and public key authentication is a more secure extension of the public key authentication method. In addition to having key pairs for both client and server, this method enables client environments to restrict the servers that they will communicate with. Implement this restriction by creating a `.rhosts` file in the client's home directory.

4.6.5.4 Password Authentication

The password authentication method relies on the existence of a single user ID and password-based login. This login could be based on the user's login specified in `/etc/passwd`, or it could be PAM-based.

HP-UX Secure Shell is fully integrated with PAM modules available on the server system. For this purpose, the `/opt/ssh/etc/sshd_config` file carries a UsePAM configuration directive. If set to `YES`, any password authentication request from the client causes `sshd` to look at the PAM configuration file (`/etc/pam.conf`). Password authentication is then done through the configured PAM modules, in sequence, until successful. For more information on PAM authentication, see *pam.conf*(4).

Set the UsePAM directive to `NO` to ignore PAM authentication. Then any password authentication request from the client causes `sshd` to ignore PAM configuration settings on the server. Instead, `sshd` obtains user password information by directly calling the `getpwnam` library call

HP-UX Secure Shell has been tested with PAM_UNIX, PAM_LDAP and PAM_KERBEROS. It is also expected to work with other PAM modules, such as PAM_DCE and PAM_NTLM.

4.6.6 Communication Protocols

HP-UX Secure Shell users can connect with a remote `sshd` daemon using the SSH-1 or SSH-2 protocol. SSH-2 is more secure, and is strongly recommended instead of SSH-1.

4.6.7 HP-UX Secure Shell and the HP-UX System

HP-UX Secure Shell is actually not a true shell. It is a mechanism for creating a secure connection between a client and a remote host to execute remote shell sessions securely on the host. To achieve the secure connection, HP-UX Secure Shell does most of the authentication and session creation itself. Following is a partial list of features that HP-UX Secure Shell uses:

- Logging of `login` attempts

  Like `telnet` or `remsh`, HP-UX Secure Shell logs successful and unsuccessful sessions in the `/var/adm/wtmp` and `/var/adm/btmp` files, respectively. For more information, see *utmp*(4).

- PAM modules

  As described in Section 4.6.5, HP-UX Secure Shell can use PAM authentication for client sessions. When PAM authentication is selected, HP-UX Secure Shell uses the `/etc/pam.conf` file and invokes the appropriate PAM module for authentication. See *pam.conf*(4) for more information about the `/etc/pam.conf` file.

- Use of `/etc/default/security` file

  This is a systemwide configuration file that contains attributes defining the behavior of login, passwords, and other security configurations. HP-UX Secure Shell allows use of these attributes with some restrictions, which are explained in the `/opt/ssh/README.hp` file for HP-UX Secure Shell.

  More information on the `/etc/default/security` file is in *security*(4).

- Shadow passwords

  HP-UX Secure Shell is integrated with the HP-UX shadow password feature. For more information, see *shadow*(4).
- Control system log (`syslog`)

  HP-UX Secure Shell uses `syslog` to write important messages. For more information, see *syslog*(3C) and *syslogd*(1M).
- Audit logging

  HP-UX Secure Shell has implemented audit logging (in trusted mode) in its own code. For more information, see *audit*(5).

4.6.8 Associated Technologies

HP-UX Secure Shell has been tested with the following technologies:

- Kerberos 5 and GSS-API
- OpenSSL
- IPv6
- TCP Wrappers
- PAM (PAM_UNIX, PAM_Kerberos, PAM_LDAP)
- HP-UX Strong Random Number Generator

4.6.9 Strong Random Number Generator Requirement

As with all cryptographic key-based products, HP-UX Secure Shell requires a random number generator. It looks for the HP-UX Strong Random Number Generator special device files, `/dev/urandom` and `/dev/random`, and uses the first special device file it locates. If these two files do not exist on the system, HP-UX Secure Shell uses its own internal random number generator, `ssh-rand-helper`.

The HP-UX Strong Random Number Generator improves the performance and entropy (a measure of the randomness and therefore the security of generated keys) of HP-UX Secure Shell. It generates nonreproducible, true random numbers. The use of the HP-UX Strong Random Number Generator is highly recommended with HP-UX Secure Shell.

The HP-UX Strong Random Number Generator is available by default. For more information, see *random*(7).

4.6.10 TCP Wrappers Support

The HP-UX Secure Shell daemon, `sshd`, is linked with the archive library, `libwrap.a`, to support TCP Wrappers. See also Section 4.3.

4.6.11 chroot Directory Jail

`chroot` is a directory jail. It starts up an application in a specified directory and restricts users to accessing that directory and the directories below it. It prevents users from changing directories above that specified directory. It is intended to restrict file and directory access to users of that application while they are using the application.

You must enable `chroot` for an application. You must create new directories and copy the relevant set of files into those newly created directories.

You can optionally set up `ssh`, `scp`, and `sftp` with a `chroot` directory.

The HP-UX Secure Shell `README` file in `/opt/ssh/README.hp` explains the `chroot` feature, the `chroot` setup script, and the specific files that this script copies to enable `ssh`, `sftp`, and `scp` for a `chroot` environment. Refer also to *chroot*(1M).

The `chroot` setup script is in the `/opt/ssh/utils/ssh_chroot_setup.sh` file, which is part of the HP-UX Secure Shell software product (Secure Shell 4.30.004/005).

Part II Protecting Data

HP-UX 11i offers data protection in many forms: protecting data in transit, in use, and at rest. By using security features designed to protect data in its three forms, HP-UX 11i customers can minimize possible breaches not only in terms of data loss, but in customer trust as well.

This section describes the following topics:

- File system security (Chapter 5)
- Compartments (Chapter 6)
- Fine-grained privileges (Chapter 7)

5 File System Security

This chapter explains file system security. Before you read this chapter, you should have a basic understanding of files and file systems.

Because data is stored in files, it is important to understand how to protect them. This chapter discusses the following topics:

- Controlling file access (Section 5.1)
- Setting access control lists (Section 5.2)
- Using HFS ACLs (Section 5.3)
- Using JFS ACLs (Section 5.4)
- Comparison of JFS and HFS ACLs (Section 5.5)
- ACLs and NFS (Section 5.6)
- Security considerations for /dev devices special files (Section 5.7)
- Protecting disk partitions and logical volumes (Section 5.8)
- Security guidelines for mounting and unmounting file systems (Section 5.9)
- Controlling file security on a network (Section 5.10)

5.1 Controlling File Access

Working groups, file permissions, file ownership, and compartment rules determine who can access a given file. The simplest of the file access rules are standard UNIX file permissions.

You can divide users into groups so that files owned by the group can be shared within the group and can be protected from outsiders.

The traditional UNIX file permissions are displayed using the `ls` command with the `-l` flag. The permissions indicate what kind of access (that is, the ability to read, write, and execute) is granted to the owner and groups on your system. Traditional UNIX file protections allow some control over who can access your files and directories, but they do not allow you to define access for individual users and groups beyond the owning user and the owning group. The following is a brief review of UNIX file permissions.

Each file and each directory has nine permissions associated with it. Files and directories have the following three types of permissions:

- `r` (read)
- `w` (write)
- `x` (execute)

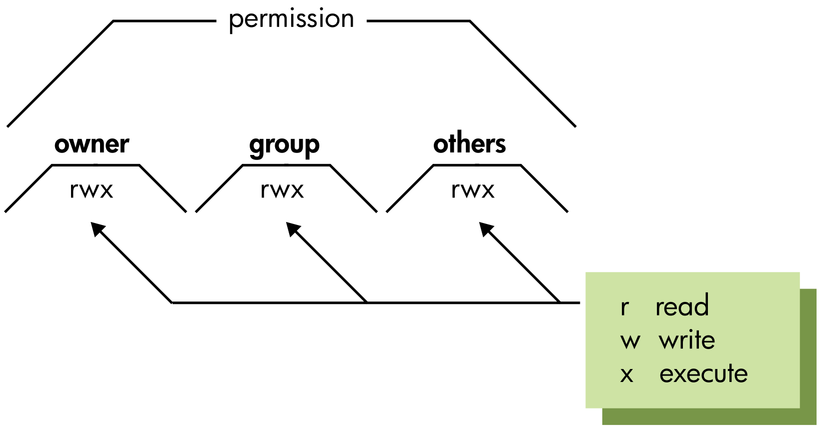
These three permissions occur for each of the following three classes of users:

- `u` (user/owner)
- `g` (group)
- `o` (all others; also known as world)

The r permission allows users to view or print the file. The w permission allows users to write (modify) the file. The x permission allows users to execute (run) the file or to search directories.

Figure 5-1 shows the traditional permissions fields.

Figure 5-1 File and Directory Permission Fields



The user/owner of a file or directory is generally the person who created it. If you are the owner of a file, you can change the file permissions with the chmod command.

The group specifies the group to which the file belongs. If you are the owner of a file, you can change the group ID of the file with the chgrp command.

The meanings of the three types of permissions differ slightly between ordinary files and directories. See Table 5-1 for more information.

Table 5-1 Differences Between File and Directory Privileges

Permission	File	Directory
r (read)	Contents can be viewed or printed.	Contents can be read, but not searched. Normally r and x are used together.
w (write)	Contents can be changed or deleted.	Entries can be added or removed.
x (execute)	File can be used as a program.	Directory can be searched.

5.1.1 Setting File Access Permissions

The chmod command changes the type of access (read, write, and execute privileges) for the file's owner, group members, or all others. Only the owner of a file or a user with the appropriate privileges can change file access. See *chmod*(1).

By default, the initial set of read and write permissions for files and directories are determined by the creator's `umask` value. To change the default file permissions, use the `umask` command. See *umask*(1).

Each bit that is set in the file mode creation mask causes the corresponding permission bit in the file mode to be cleared (disabled). Conversely, bits that are clear in the mask allow the corresponding file mode bits to be enabled in newly created files.

For example, a `umask` of octal 022 creates a mask of `u=rwx`, `g=rx`, `o=rx`, which disables group and other write permissions.

5.1.2 Setting File Ownership

The `chown` command changes file ownership. To change the owner, you must own the file or have the appropriate privileges.

The `chgrp` command changes file group ownership. To change the group, you must own the file or have the appropriate privileges.

For more information, see *chown*(1) and *chgrp*(1).

5.1.3 Protecting Directories

Normally, if a directory is writable either through standard permissions or through ACLs, anyone can remove the files in the directory, regardless of the permissions on the files themselves. To protect against unwanted file deletions in a directory:

- Remove write permissions for directories that should not have them.

  This is particularly effective for users' private directories. The following command allows others to read and search the `mydir` directory, but only the owner can delete files from it:

  ```
  # chmod 755 mydir
  ```

  See *chmod*(1) and *chmod*(2).

- Set the **sticky bit** on the directory.
- The sticky bit is a special bit in the mode of every file. Setting the sticky bit prevents users from removing other users' files from that directory. Setting the sticky bit for a directory allows only the owner of the file, the owner of the directory, or a user with the appropriate privileges to delete or to rename the files.

  This is effective for temporary or project directories (such as `/tmp` and `/var/tmp`) that must be accessible to many authorized users. The following command allows anyone to create, read, and write files in `/mfgproj`, but only the file owner, the directory owner, or a user with the appropriate privileges can delete files:

  ```
  # chmod a+rwxt /mfgproj
  ```

  Setting the sticky bit is important for directories that are used for temporary files. In the event that a temporary directory is not set to sticky, an attacker may alter the expected behavior of user programs by waiting for a temporary file to be created and then deleting and recreating a new file with modified content, but the same name. In most cases, the application is unaware of the change and may unintentionally perform malicious acts on behalf of the attacker.

5.1.4 Protecting Files Related to User Accounts

Follow these guidelines to protect files related to user accounts:

- A home directory should not be writable by anyone except for the owner. Otherwise, any user can add and remove files from the directory.
- The `.profile`, `.kshrc`, `.login`, and `.cshrc` files for each user should not be writable by anyone other than the account owner.
- 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, and prevents anyone from editing your `.rhosts` file to gain access to those systems. For more information, see *hosts.equiv*(4).
- Do not use a `.netrc` file, because it bypasses `login` authentication for remote login and even contains the user's unencrypted password. If used, `.netrc` must not be readable or writable by anyone other than its owner. For more information, see *netrc*(4).

5.1.5 Locating and Correcting File Corruption Using fsck

The following problems can indicate a corrupt file system:

- A file contains incorrect data (garbage).
- A file has been truncated or is missing data.
- Files disappear or change locations unexpectedly.
- Error messages appear on a user's terminal, the system console, or in the system log.
- You are not able to change directories or list files.
- The system fails to reboot.

If you or other users cannot readily identify problems with the file system, use the `fsck` command to check the file system. The `fsck` command is the primary tool for finding and correcting file system inconsistencies. The `fsck` command examines the file system listed in `/etc/fstab`.

The `fsck` utility is not capable of detecting file corruption. If `fsck` does not find any errors, the problem is likely not a corrupted file system. That is, the file system is usable, even if the underlying data is lost or corrupted. Look for one or more of these other file problems:

- A user, program, or application deleted, overwrote, moved, or truncated the file or files.
- The file system associated with a particular directory when the file was created might not be mounted to that directory.

- A file or files were placed in a directory that now has a file system mounted to it. The files still exist but are not accessible. Unmount the file system to access the files.
- The file protection or ownership is preventing access. Use the `chmod` or `chown` command to change file permissions.

5.2 Setting Access Control Lists

Access control lists (ACLs) offer a finer degree of file protection than traditional file access permissions. Use ACLs to allow or restrict file access to individual users unrelated to the group they belong to. Only the owner of a file, or a user with the appropriate privileges can create ACLs.

Both the Journaled File System (JFS) and High-Performance File System (HFS) support ACLs but they use different mechanisms and syntax.

JFS is the HP-UX implementation of the Veritas journaled file system (VxFS). HFS is the HP-UX version of the UNIX File System (UFS) and is compatible with earlier versions of HP-UX.

An access control list (ACL) is a set of user, group, and mode entries associated with a file. The list specifies permissions for all possible user ID and group ID combinations. Access control lists give you a more precise way to control access to files than you have with traditional UNIX file permissions. ACLs enable you to grant or restrict file access in terms of individual users and specific groups, in addition to the traditional control.

Both HFS and JFS file systems support ACLs, but they use different mechanisms and use different syntax.

NOTE: HFS is now deprecated. It will be removed from the operating system in a future release.

HP-UX supports two separate JFS products: the basic JFS product, which is included in the operating system, and the optional advanced product, OnLineJFS, which is installed separately. Both JFS products support ACLs.

For more information, see *setacl*(1), *getacl*(1), *aclv*(5), *chacl*(1), *lsacl*(1), and *acl*(5).

5.3 Using HFS ACLs

You set HFS ACL permissions with the `chacl` command and display them with the `lsacl` command. See Example 5-1.

IMPORTANT: You must use `chmod` with the `-A` option when working with files that have HFS ACL permissions assigned. Without the `-A` option, `chmod` will delete the ACL permissions from the file. The syntax is:

```
# chmod -A mode file
```

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.

When a file has ACLs, the `ll` command displays a plus sign (+) after the permission string.

If a *`user.group`* matches more than one HFS ACL entry, the more specific entry takes precedence. See Example 5-2.

Example 5-1 Creating an HFS ACL

In this example, the `chmod` command restricts write permissions for `myfile` to only the user, `allan`. The `chmod` command also deletes any previous HFS ACLs.

```
$ chmod 644 myfile
$ ll myfile
-rw-r--r--   1 allan      users         0 Sep 21 16:56 myfile
$ lsacl myfile
(allan.%,rw-)(%.users,r--)(%.%,r--) myfile
```

The `lsacl` command displays just the default (no ACL) values, corresponding to the basic owner, group, and other permissions.

The `chacl` command gives read and write access to `myfile` to another user.

```
$ chacl 'naomi.users=rw' myfile
$ ll myfile
-rw-r--r--+  1 allan      users         0 Sep 21 16:56 myfile
$ lsacl myfile
(naomi.users,rw-)(allan.%,rw-)(%.users,r--)(%.%,r--) myfile
```

Notice two things: the `ll` permissions display has a + appended, indicating that ACLs exist and that the `ll` permissions string did not change. The additional entry in the `lsacl` display specifies that user `naomi` in group `users` has read and write access to `myfile`.

Example 5-2 Multiple HFS ACL Matches

If a user's *`user.group`* combination matches more than one ACL entry, the most specific entry takes precedence. In this example, first set the file permissions.

```
$ chmod 644 myfile
```

Use the `chacl` command on `myfile` to add a write-only entry for user `naomi`:

```
$ chacl naomi.%=w myfile
$ lsacl myfile
(naomi.%,-w-)(allan.%,rw-)(%.users,r--)(%.%,r--) myfile
```

Now, user `naomi` has write access to file `myfile`, using the ACL defined for `naomi.%`, but does not have read access to the file because `naomi.%` takes precedence over the ACLs defined for `%.users` and `%.%`.

The `lsacl`command displays the HFS ACLs in decreasing order of specificity. That is, permission matches are attempted from left to right.

5.3.1 HFS ACLs and HP-UX Commands and Calls

The following commands and system calls work with ACLs on HFS file systems:

Table 5-2 HFS ACL Commands

Commands	Description
`chacl`	Changes HFS ACLs of files.
`getaccess`	Lists user's access rights to files.
`lsacl`	Lists HFS ACLs of files.

Table 5-3 HFS ACL System Calls

System Call	Description
`getaccess`	Gets a user's effective access rights to a file.
`getacl`, `fgetacl`	Gets HFS ACL information.
`setacl`, `fsetacl`	Sets HFS ACL information.
`acltostr`	Converts HFS ACL structure to string form.
`chownacl`	Changes the owner or group represented in an HFS file's ACL.
`cpacl`, `fcpacl`	Copies HFS ACL and mode bits from one file to another.
`setaclentry`, `fsetaclentry`	Adds, modifies, or deletes an HFS file's ACL entry.
`strtoacl`	Parses and converts HFS ACL structure to string form.
`strtoaclpatt`	Parses and converts HFS ACL pattern strings to arrays.

The following commands, system calls, and subroutine libraries affect ACL entries, sometimes in unexpected ways.

Table 5-4 Commands and Calls Affecting ACL Entries

Command or Call	Description
`chmod`	Deletes HFS ACLs by default. Use the `-A` option to retain HFS ACLs.
`chmod`	Deletes HFS ACL entries. Use `getacl` and `setacl` to save and restore the HFS ACL entries.
`cpset`	Does not set a file's optional ACL entries.
`find`	Identifies files whose ACL entries match or include specific ACL patterns on HFS or JFS file systems.
`ls -l`	The long form indicates the existence of ACLs by displaying a plus sign (+) after the file's permission bits.

Table 5-4 Commands and Calls Affecting ACL Entries *(continued)*

Command or Call	Description
`mailx`	Does not support optional ACL entries on `/var/mail/*` files.
`compact`, `compress`, `cp`, `ed`, `pack`, `unpack`	Copies ACL entries to the new files they create.
`frecover`, `fbackup`	Use only these commands to selectively recover and back up files. Use the `-A` option when backing up from an ACL system for recovery on a system that does not support ACLs.
`ar`, `cpio`, `ftio`, `shar`, `tar`, `dump`, `restore`	These commands do not retain ACLs when archiving and restoring. They use the `st_mode` value returned by `stat`.
`rcs`, `sccs`	These commands do not support ACLs.

HFS access control lists use additional "continuation inodes" when creating new file systems. Consider them when using the following commands:

- `fsck`: Returns the number of files with ACL entries as a value for *`icont`*. Use the `-p` option to clear unreferenced continuation inodes. See *fsck*(1M).
- `diskusg`, `ncheck`: Ignores continuation inodes. See *diskusg*(1M) and *ncheck*(1M).
- `mkfs`: Allows for continuation inodes on new disks. See *mkfs*(1M).

5.4 Using JFS ACLs

This section describes JFS ACLs and how to use them.

NOTE: To use JFS ACLs, you must have a VxFS file system using disk layout Version 4. See *vxupgrade*(1M) for information about upgrading the file system to Version 4.

5.4.1 Definition of a JFS ACL

A JFS ACL contains one-line entries naming specific users and groups and indicating what access is granted to each. The presence of a JFS ACL also changes the meaning of the `group` permission bits, which are displayed using the `ls -l` command.

A JFS ACL always has at least four entries: a `user` entry, a `group` entry, a `class` entry, and an `other` entry. When a JFS ACL contains only these four entries, the permissions it grants are exactly the same as the permissions represented by the standard UNIX system permission bits.

5.4.2 How the System Generates a JFS ACL

Whenever a file is created on a JFS file system, the system initializes a minimal JFS ACL for the file, containing a `user` entry for the owner permissions, a `group` entry for the owning group permissions, a `class` entry for the owning group permissions, and an `other` entry for the other group permissions. Additional entries can be added by the user, or as a result of default entries specified on the parent directory.

5.4.3 Minimal JFS ACL

An ACL with the four basic entries defined previously is called a minimal JFS ACL. An example minimal ACL looks like this:

```
user::rw-
group::r--
class:r--
other:---
```

- The `user` entry indicates the permissions of the owner of the file and maps directly to the owner permission bits. Because the first entry applies to the owner of the file, no user name needs to be indicated. This example ACL entry grants read and write access to the file's owner.
- The `group` and `class` entries specify the permission granted to members of the file's owning group. The example ACL entry grants read-only access to the file's owning group. The `group` and `class` entries are described more in Section 5.4.5.
- The `other` entry is a catch-all entry that specifies permissions for anyone who is not granted or denied permission by any other entry. The example `other` entry denies access to all users who are not the owner of the file nor in the file's owning group.

The permission bits displayed by `ls -l` for this file would look like this:

```
rw-r-----
```

The next section describes how additional JFS ACL entries affect file access and the interpretation of the permission bits.

5.4.4 Additional JFS ACL user and group Entries

If you want to grant or deny access to specific users and groups on the system, you can add up to 13 more `user` and `group` entries to the four minimal entries described in the previous section.

For example, the following entry in the ACL of a file grants read, write, and execute access to a user logged in as `boss`:

```
user:boss:rwx
```

In the next example, an ACL with the following entry denies access to a user in the group `spies`:

```
group:spies:---
```

5.4.5 JFS ACL group and class Entries

In a file with a minimal ACL, the owning `group` and `class` ACL entries are identical. However, in a file with additional entries, the owning `group` and `class` ACL entries are distinct. The owning `group` entry grants permissions to a specific group: the owning `group`.

The `class` entry is more general; it specifies the maximum permissions that can be granted by any of the additional `user` and `group` entries.

If a particular permission is not granted in the `class` entry, it cannot be granted by any ACL entries except for the first `user` (owner) entry and the `other` entry. Any permission can be denied to a particular user or group. The `class` entry functions as an upper bound for file permissions.

When an ACL contains more than one `group` or `user` entry, the additional `user` and `group` entries are referred to as the `group class` entries, because the effective permission granted by any of these additional entries is limited by the `class` entry.

5.4.6 Using the setacl and getacl Commands

Use the `setacl` and `getacl` commands to change and view ACLs.

Use the `setacl` command to change the ACL in one of the following ways:

- Replace a file's entire ACL, including the default ACL on a directory.
- Add, modify, or delete one or more entries, including default entries on directories.

The `getacl` command displays the entries in the ACL. File permission bits for `user` and `group` are translated into special cases of these entries:

- The bits representing owner permissions are represented by a `user` entry without a specified user ID.
- The bits representing group permissions are represented by a `group` entry without a specified group ID.

  An ACL must contain one each of these special `user` and `group` entries. The ACL can have any number of additional `user` entries and `group` entries, but these must all contain a user ID or group ID, respectively. An ACL has only one `other` entry, representing the permission bits for permissions to be granted to other users.

See *setacl*(1) and *getacl*(1) for command descriptions.

5.4.7 Effect of chmod on class Entries

When a file has a minimal ACL, the owning `group` and `class` ACL entries are identical, and `chmod` affects both of them. However, when a file contains additional, optional entries in the ACL, the following consequences occur:

- The `class` ACL entry no longer necessarily equals the owning `group` ACL entry.
- The `chmod` command affects the `class` ACL entry, not the owning `group` entry.
- You must use the `setacl` command to change the owning `group` entry.

5.4.8 Example of Changing a Minimal JFS ACL

To illustrate the function of the JFS ACL `class` entry, this section describes how `chmod` and `setacl` affect a file with a minimal JFS ACL and a file with `group class` entries.

NOTE: Further details about the use of the `getacl` and `setacl` commands are in Section 5.4.10. Refer also to *getacl*(1) and *setacl*(1).

Consider a file, `exfile`, with read-only (`444`) permissions and a minimal JFS ACL. The **`ls -l`** command shows the permissions for `exfile`:

```
$ ls -l exfile
-r--r--r-- 1 jsmith users 12 Sep 20 15:02 exfile
```

The `getacl` command lists the following output for `exfile`, which is a minimal JFS ACL:

```
$ getacl exfile
# file: exfile
# owner: jsmith
# group: users
user::r--
group::r--
class:r--
other:r--
```

Using the `chmod` command to add write permissions to `exfile` changes both the owning `group` and the `class` ACL entries. For example, look at the `getacl` command output:

```
$ chmod 666 exfile
$ getacl exfile
# file: exfile
# owner: jsmith
# group: users
user::rw-
group::rw-
class:rw-
other:rw-
```

Now add additional user and group entries, that will affect the `class` ACL entry but not the owning `group` entry. The first `setacl` command that follows grants read-only permission to user `guest`; the other ACL entries are unaffected. However, the second `setacl` command grants read-execute permissions to the group `dev`, and the upper bound on permissions (the `class` entry) is extended to include execute permission.

```
$ setacl -m u:guest:r-- exfile
$ setacl -m g:dev:r-x exfile
$ getacl exfile# file: exfile
# owner: jsmith
# group: users
user::rw-
user:guest:r--
group::rw-
```

```
group:dev:r-x
class:rwx
other:rw-
```

Next, the `chmod` command removes write and execute permission from `group`, and actually reduces the `class` permissions to read-only. The owning `group` permissions, while unchanged, are effectively reduced to read-only as well.

```
$ chmod g-wx exfile
$ getacl exfile
# file: exfile
# owner: jsmith
# group: users
user::rw-
user:guest:r--
group::rw-      # effective:r--
group:dev:r-x  # effective:r--
class:r--
other:rw-
```

The `other` permissions are unchanged. The `class` entry does not limit the access that can be granted by the first `user` (owner) entry or the `other` entry.

Next the `ls -l` command lists the permissions of `exfile`. The plus sign (+) at the end of the permissions string indicates that there is an ACL for the file.

```
$ ls -l exfile
-rw-r--rw-+ 1 jsmith users 12 Sep 20 15:02 exfile
```

5.4.9 Default JFS ACLs

You might want all the files created in a directory to have certain ACL entries. For example, you can allow another person to write to any file in a directory of yours when the two of you are working on something together.

You can put an ACL entry granting the desired access on every file in the directory, but every time you create a new file, you have to add that entry again. Using default ACL entries, you can get the system to do this for you automatically every time you create a file.

A default ACL entry appears as follows:

```
default:user:boss:rw-
```

Default ACLs can only be placed only on a directory and have no influence on what access to the directory is granted to a user. The default ACL is applied to files created in the directory.

When the newly created file is a directory, the default ACL entries have two effects:

- The corresponding nondefault ACL entries are created, so that the desired permissions are granted and denied for the directory, just as for any file created in the directory.
- The default entries themselves are copied, so that the new subdirectory has the same default ACL as the parent directory.

For example, if you want any files created in the directory `projectdir` to be readable by certain users, you can create the appropriate default entries, as follows:

```
$ setacl -m d:u:boss:r,d:u:jjones:r,d:u:dev:r projectdir
```

```
$ getacl projectdir
# file: projectdir
# owner: jsmith
# group: users
user::rw-
user:boss:rw-
user:jjones:rw-
user:jdoe:---
group::rw-
group:dev:rw-
class:rw-
other:---
default:user:boss:r---
default:user:jjones:r--
default:group:dev:r--
```

If the newly created file is a directory, the same ACL entries are generated. In addition, the default entries themselves are also placed in the ACL.

With these entries in place, any new file created in the directory `projectdir` will have an ACL like that shown previously without the default entries.

5.4.10 Changing JFS ACL with the setacl Command

This section presents more examples of using the `setacl` command.

5.4.10.1 Using the Modify and Delete Options

The following `setacl` command uses the `-m` (modify) option to give read-only access to the user `boss` for the `junk` file:

```
$ setacl -m u:boss:r-- junk
```

To grant read and write access to everyone in the group `dev`, use the group (`g:`) parameter with the `setacl -m` command:

```
$ setacl -m g:dev:rw- junk
```

The `-d` option deletes an entry. With `-d`, do not specify any permissions in the ACL entry. For example, the following command deletes the entry for the group `dev`:

```
$ setacl -d g:dev junk
```

5.4.10.2 Using the -f Option

If you are adding or changing several entries, you can use a different procedure. You can save the ACL to a file, edit the file, and then apply this new ACL to the file. For example, save the ACL to a file with this command:

```
$ getacl junk > junk.acl
```

Edit the file so that it appears as follows:

```
$ cat junk.acl
# file: junk
# owner: user1
# group: group1
user::rw-
user:user2:rw-
user:user3:rw-
user:user4:---
user:user5:r--
group::rw-
group:group2:rw-
group:group3:r--
group:group4:---
group:group5:rw-
class:rw-
other:r--
```

Apply the ACL to the file using the `setacl -f` command:

```
$ setacl -f junk.acl junk
```

5.4.10.3 Effective Permissions and setacl -n

Normally, `setacl` recalculates the `class` entry to ensure that permissions granted in the additional ACL entries are granted. If you specify the `-n` option, the `class` entry is not recalculated; the existing value is used. This means that some permissions granted by the ACL entries will not be granted in practice.

For example, this ACL is modified with the `setacl -n` command to add read and execute permissions to group `dev` as follows:

```
$ getacl exfile
# file: exfile
# owner: jsmith
# group: users
user::rw-
group::rw-
class:rw-
other:rw-
```

```
$ setacl -n -m group:dev:r-x exfile
$ getacl exfile
# file: exfile
# owner: jsmith
# group: users
user::rw-
group::rw-
group:dev:r-x     #effective r--
class:rw-
other:rw-
```

The group `dev` ACL entry is added as specified, but execute permission is not actually granted. Execute permission is denied by the `class` entry, and the `class` entry was not recalculated because `-n` was specified. If `-n` was not used, `class` would have been reset to `class:rwx`, and the `effective` comment would not be there.

5.5 Comparison of JFS and HFS ACLs

JFS ACLs adhere to the POSIX ACL standard.

JFS ACLs differ from HFS ACLs in both format (internal and external) and functionality.

Functional differences between JFS and HFS ACLs include the following:

- A JFS directory's ACL can have default entries, which are applied to files subsequently created in that directory. HFS ACLs do not have this capability.
- An HFS ACL has an owner that can be different from the owner of the file the ACL controls. JFS ACLs are owned by the owner of the corresponding file.
- An HFS ACL can have different entries for a particular user in specific groups. For example, `userx` might have read and write access as a member of group `users`, but have only read access as a member of group `other`.

5.5.1 JFS and HFS Command and Function Mapping

Table 5-5 lists the manpages for the equivalent commands and functions for JFS ACLs and HFS ACLs.

Table 5-5 HFS and JFS ACL Equivalents

HFS Name	**JFS Equivalent**
chacl(1)	*setacl*(1)
lsacl(1)	*getacl*(1)
getacl(2)	*acl*(2)
fgetacl(2)	—none—
setacl(2)	*acl*(2)
fsetacl(2)	—none—
acltostr(3C)	—none—
chownacl(3C)	—none—
cpacl(3C)	—none—
setaclentry(3C)	—none—
strtoacl(3C)	—none—
—none—	*aclsort*(3C)
acl(5)	*aclv*(5)

5.6 ACLs and NFS

The Network File System (NFS) has no facility to pass ACL information about remote files. Therefore, ACLs are not visible on remote files by NFS. The `ls -l` command will not show that ACLs exist on a remote file, but the ACL control over access permissions remains effective.

Individual manpage entries specify the behavior of the various system calls, library calls, and commands under these circumstances.

IMPORTANT: Use caution when transferring a file with optional entries over a network, or when manipulating a remote file, because NFS can delete optional entries with no notification.

5.7 Security Considerations for /dev Device Special Files

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

If you install any other device special files, see *insf*(1M) for information about correct permission settings.

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

- Keep all device special files in the `/dev` directory.
- Protect the memory files, `/dev/mem` and `/dev/kmem`, from casual access, because these files contain sensitive user information. For example, a program that watches memory for an invocation of the `login` program might copy the password from the `login` program buffers when a user types it in. The file protections should be set to:

```
crw-r-----   1 bin        sys           3 0x000001 Jun  9  2006 /dev/kmem
crw-r-----   1 bin        sys           3 0x000000 Jun  9  2006 /dev/mem
```

- Protect all disk special files:
  — Write protect all disk special files from general users to prevent inadvertent data corruption. Turn off write access for `group` and `other`.
  — Read protect disk special files to prevent disclosure. Turn off read access for `other`.

  The file protections should be set to:

```
brw-r-----   1 bin        sys          31 0x002000 Feb 18  2004 /dev/dsk/c0t2d0
crw-r-----   1 bin        sys         188 0x002000 Aug  3  2004 /dev/rdsk/c0t2d0
brw-r-----   1 root       sys          64 0x000002 Jun 11  2006 /dev/vg00/lvol2
crw-r-----   1 root       sys          64 0x000002 Jun 11  2006 /dev/vg00/rlvol2
```

- Terminal ports on HP-UX systems are writable by anyone if you allow users to communicate by using the `write` or `talk` programs. Permit only the owner to have read permission.

- Do not permit individual users to own a device special file other than for a terminal device or personal printer.
- Before putting a disk or other mountable device of unknown origin into service, check its files for device special files and `setuid` programs. See Section 5.9.

5.8 Protecting Disk Partitions and Logical Volumes

A Logical Volume Manager (LVM) is a common disk management tool. LVM divides up the disk more easily than disk partitions, and the volumes can span multiple disks. Volumes are logical devices that appear as a physical disk partition. You can use a volume as a virtual disk partition for such applications as creating a file system or a database.

Following are some security considerations regarding disk partitions and logical volumes:

- Ensure that the device special files for disk partitions and logical volumes are readable only by root and perhaps by an account used for disk backups. See Section 5.7.
- Because 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. The file is subject to change regardless of the owner, bypassing the `chmod` system call and other security checks.

  If the device special file is writable, a user can open that file and access the raw disk. The user can then directly edit the file system, read files, or change file permissions and owners.

  Make sure the file permissions forbid access to the device special file and allow only root to read.
- If a program, such as a database application, requires direct access to the partition, reserve that partition exclusively for the program. Do not mount a partition as a file system if users can access the partition directly. If you do mount a partition as a file system, users could edit the underlying file system.

  Inform program users that the file's security is enforced by its permission settings rather than by the HP-UX file system.

5.9 Security Guidelines for Mounting and Unmounting File Systems

The `mount` command enables you to attach removable file systems and disk or disk partitions to an existing file tree. The `mount` command uses a file called `/etc/fstab`, which contains a list of available file systems and their corresponding mount points. Make the `/etc/fstab` file writable only by root, but readable by others. For more information on mounting file systems, see *fstab*(4).

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 on a directory that already contains 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.

- Set permissions and access control list entries on disk path names to control access to disks.
- Use the `-r` option of the `mount` command to mount the file system as read-only. You must mount physically write-protected file systems this way.
- When mounting a new or foreign file system, assume that the medium is insecure.
  - Make sure that the `PATH` environment variable does not include "`.`" (the current directory); otherwise, you might run a Trojan horse version of `ls` or some similar command while examining the new file system.
  - Run the `fsck` command to verify that the file system is not technically corrupted. See *fsck*(1M).
  - Run the `ncheck_hfs -s` or `ncheck_vxfs -s` command to scan for `setuid` and `setgid` programs and device files, and investigate any suspicious findings. The `-s` option is intended to discover concealed violations of security policy. For more information, see *ncheck_hfs*(1M) and *ncheck_vxfs*(1M).
  - Create a directory restricted to root by setting its permissions at `700` (`drwx------`).

    ```
    # mkdir /securefile
    # chmod 700 /securefile
    ```

  - Mount the foreign file system read-only at that location:

    ```
    #  mount -r /dev/disk1 /securefile
    ```

  - Check all directories for privileged programs, and verify the identity of every program.
  - Remount the system read and write permissions and remove any unnecessary `setuid` and `setgid` permissions from files that you discovered in the previous step. These precautions are especially important if a user requests that you mount a personal file system.

  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 Section 5.10.2.

5.10 Controlling File Security on a Network

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

Although you cannot completely control security over the network, you can control the security of each node on the network to limit penetration risk without reducing the usefulness of the system or user productivity.

Ensure that all network administration programs are owned by a protected, network-specific account, such as `uucp`, `nso`, or `daemon`, rather than by root.

5.10.1 Check Permission Settings on Network Control Files

Modes, owners, and groups on all system files are set carefully. Check these files regularly for any changes. Note and correct any changes from the original values.

Pay particular attention to the network control files in the `/etc` directory. These files are of notable interest to those attempting to gain unauthorized access, because they provide access to the network itself. Network control files should never be writable by the public. These files include:

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

5.10.2 Files Mounted in an NFS Environment

A Network File System (NFS) provides the following conveniences:

- Saves file space.
- Maintains consistent file usage.
- Provides a lean cooperative user environment.

NFS streamlines filesharing 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 or specified list of machines. When 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 in to the server system. See *exports*(4) for information on controlling access to exported file systems and see Section 5.10.2.3 for security guidelines.

5.10.2.1 Server Vulnerability

Maintain server security by setting restrictive permissions on the `/etc/exports` file. 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 by the client to server files 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 privileges on an NFS client can exploit that privilege to obtain unlimited access to an NFS server.

NOTE: Never export any file system to a node on which privilege is granted more leniently than in your own node's policy.

5.10.2.2 Client Vulnerability

In earlier releases of NFS for workstations, the `/dev` inode had to reside on the client's disk. NFS now allows the `/dev` inode containing the major and minor numbers of a client-mounted device special file 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 special file, by accessing the device special file through the file and inode number found on the server side.

Although lacking permission to make a device special file on the client side, a system violator can create a device special file, such as `/dev/kmem`, using root permissions on the 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 violator can then go to the client, log in as an ordinary user, and, using NFS, open up the newly created server-side device special file and use it for devious means.

5.10.2.3 How to Safeguard NFS-Mounted Files

Following are suggestions to safeguard NFS-mounted files:

- If possible, make sure that the same person administers both client and server systems.
- Maintain uniformity of user ID and group ID for server and client systems.
- Routinely check the `/dev` files in the file systems exported from server.
- Restrict who can have write access to the `/etc/passwd` client files.
- For strictest control, audit every host that is accessible through the network.
- Consider using the `fstab nosuid` command to protect the system against `setuid` programs that can run as root and damage the system. The default mount option is `suid`, which allows mounted programs with `setuid` permission to run with the permissions of their owners, regardless of who starts them. Therefore, if a program

with `setuid` permission is owned by root, it will run with root permissions, regardless of who starts it.

6 Compartments

This chapter describes the compartments feature of HP-UX 11i v3. This chapter addresses the following topics:

- Overview (Section 6.1)
- Planning the compartment structure (Section 6.2)
- Compartment components (Section 6.3)
- Compartment rules and syntax (Section 6.4)
- Configuring a compartment (Section 6.5)
- Troubleshooting compartments (Section 6.6)
- Using discover mode to generate initial compartment configuration (Section 6.7)
- Compartments in HP Servicegard Clusters (Section 6.8)

6.1 Overview

Compartments are a method of isolating components of a system from one another. When configured properly, they can be an effective method to safeguard the HP-UX system and the data that resides on it.

Compartments allow you to isolate processes, or subjects, from each other and also from resources, or objects.

Conceptually, each process belongs to a compartment, and resources are handled in one of two ways:

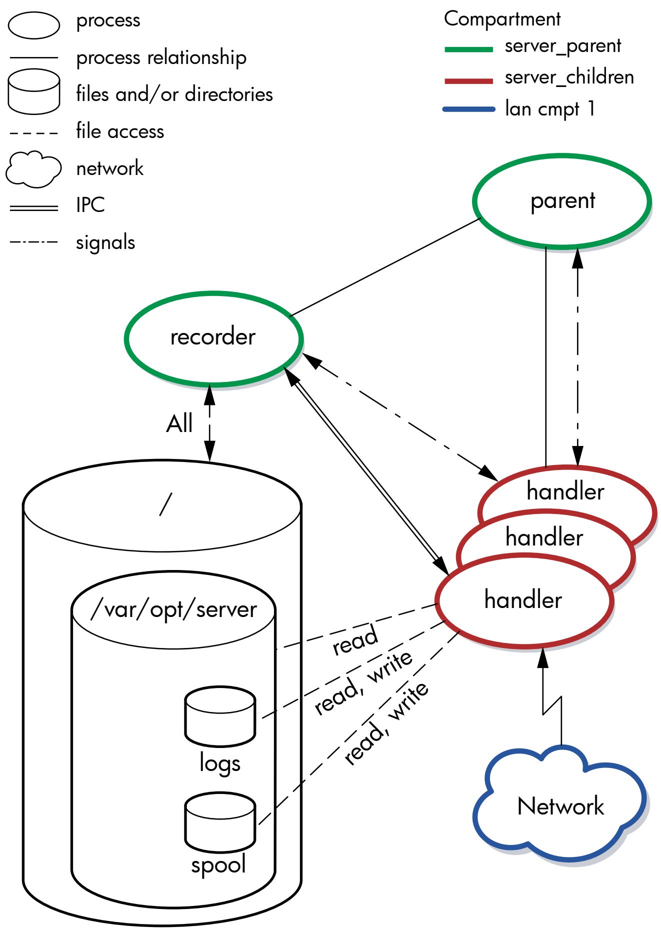
1. The resource is labeled with the compartment of the creating process. This is how transient resources, such as communication endpoints and shared memory, are assigned a compartment.
2. Resources can be associated with an access list that specifies how processes in different compartments can access them, for persistent resources such as files and directories. That is, processes can access resources or communicate with processes belonging to a different compartment only if a rule exists between those compartments. Processes that belong to the same compartment can communicate with each other and access resources in that compartment without a rule.

Compartments separate subjects from objects. This enables a virtual grouping of related subjects and objects. You can configure the system so that, if a service running in a compartment is compromised, it does not affect services running in other compartments. This restricts any damage to the affected compartment only.

6.1.1 Compartment Architecture

Compartments isolate a process and its child processes within a system. Figure 6-1 shows a parent process that spawns a number of handler processes that need to access various parts of the system. The compartments on the system are configured so that the processes can access the resources they need.

Figure 6-1 Compartment Architecture



In Figure 6-1, the parent process is configured in a compartment, compartment A. As part of its functioning, the parent process spawns a number of handler processes in a different compartment, compartment B. The handler processes inherit the compartment configuration of the parent process. The network card that connects this system to the LAN is configured in another compartment, compartment C. The file system is configured to allow full access to compartment A, but only allow partial access to compartment B. Communication between the system components in their separate compartments is configured as follows:

- All handler processes are configured to communicate with the network.
- The recorder can access the file system.
- The handler processes have read, and read/write access to parts of the file system.

- The handler processes can communicate with the parent process, and with the recorder using IPC and signals.
- The network is isolated from the recorder and the parent process.

This compartment configuration provides security for the file system and the recorder. Both are isolated by their compartments. Though the handler processes can communicate with the network, the network cannot be accessed by the recorder or the parent process.

6.1.2 Default Compartment Configuration

When you enable compartments, a default compartment named `INIT` is created. When you boot up the system, the `init` process belongs to this compartment. The `INIT` compartment is defined to have access to all other compartments and is not defined in a compartment rules file.

IMPORTANT: If you redefine the `INIT` compartment by creating explicit rules in a rules file, all special characteristics of the compartment are lost and cannot be restored without rebooting the system.

6.2 Planning the Compartment Structure

Plan the compartment structure before you begin creating compartment rules.

To plan the compartment structure, answer the following questions:

- Do you want to isolate different groups of users accessing this system? For example, is this system used by both the accounting department and the human resources department, and must these groups of users be kept separate?
- Do you want to isolate one network interface on this system, which communicates outside the firewall, from the rest of the system, which communicates only inside the firewall?
- Does the security policy include requirements or problems that can be solved by using compartments?
- Does the security policy specify or suggest a specific compartment rules configuration?

When you have answered these questions, use the answers to determine how to assign parts of the system to specific compartments.

Consider the following recommendations when planning the compartment configuration:

- Put all compartment configuration files in the `/etc/cmpt` directory.

  You can use the `#include` directive to create compartment configuration files anywhere on the system. However, HP recommends that you avoid using this option. Instead, keep the compartment configuration files together and easy to locate.

- Develop a separate compartment configuration for each component of the system.

  Unless there is a defined, specific software dependency between two components, do not mix rules for different components. One component compartment does not contain rules referring to compartments for another component. If you must remove a component, you can modify the compartment configuration more easily if the compartment configurations are kept separate.

- Create a single compartment configuration file for each software component.

  This enables you to remove the compartment configuration easily if you remove the software from the system. You can also find all rules pertaining to the software component easily.

- Some software products are shipped with compartment rules already configured. Avoid modifying these rules.

  Before you make modifications to shipped compartment configurations, be sure you understand the existing configuration. Read the documentation for the software product and examine the existing configuration carefully.

CAUTION: Do not redefine the existing `INIT` compartment. If you attempt to change or redefine the `INIT` compartment, all automatically generated definitions will be destroyed and compartments will not function properly.

6.3 Compartment Components

The compartments feature comprises a set of configuration files and commands you use to configure and administer compartments. Manpages are included to assist you in using the compartments features. These components are listed in the following sections:

6.3.1 Compartment Configuration Files

Table 6-1 briefly describes the files you use with compartment components.

Table 6-1 Compartment Configuration Files

Configuration File	Description
`/etc/cmpt`	The directory in which compartment rules files reside.
`/etc/cmpt/*.rules`	The file containing the compartment rules configured for the system.
`/etc/cmpt/cmpt.conf`	Compartment configuration file used to enable or disable the compartment login feature.
`/etc/cmpt/hardlinks/hardlinks.config`	The file containing valid mount points to be scanned to check the consistency of compartment rules for files with multiple hardlinks pointing to them.

6.3.2 Compartment Commands

Table 6-2 contains the commands you use to manage compartments.

Table 6-2 Compartment Commands

Command	Description
`cmpt_tune`	Queries, enables, and disables the compartments feature.
`setfilexsec`	Sets security attributes of binary files, including the compartment attribute.
`getfilexsec`	Displays security attributes associated with binary executable files, including the compartment attribute.
`getprocxsec`	Displays security attributes of processes, including the compartment attribute.
`getrules`	Displays the compartment rules currently active in the kernel.
`setrules`	Activates new or modified rules in the kernel. With the `-p` option, displays the modified rules for review without passing them to the kernel.
`vhardlinks`	Checks the consistency of compartment rules for files that have multiple hard links, to ensure that conflicting rules for access do not exist.

6.3.3 Compartment Manpages

Table 6-3 contains the manpages associated with compartments.

Table 6-3 Compartment Manpages

Manpage	Description
compartments(4)	Describes the HP-UX compartment files and rule syntax.
compartments(5)	Provides an overview of compartment functionality and describes the use of compartment rules.
cmpt_allow_local(5)	A kernel tunable that defines the default rule for inter-compartment local-to-local communications. This kernel tunable is available only if the HP-UX ContainmentPlus (version B.11.31.01 and later) product is installed on the system.
cmpt_restrict_tl(5)	Defines the restrictions for the inter-compartment communications through Streams Local Transport Drivers. These restrictions are available only if the HP-UX ContainmentPlus (version B.11.31.02 and later) product is installed on the system.
cmpt_tune(1M)	Describes `cmpt_tune` functionality and syntax.
setfilexsec(1M)	Describes `setfilexsec` functionality and syntax.
getfilexsec(1M)	Describes `getfilexsec` functionality and syntax.
getprocxsec(1M)	Describes `getprocxsec` functionality and syntax.

Table 6-3 Compartment Manpages *(continued)*

Manpage	Description
getrules(1M)	Describes `getrules` functionality and syntax.
setrules(1M)	Describes `setrules` functionality and syntax.
vhardlinks(1M)	Describes `vhardlinks` functionality and syntax.
compartment_login(5)	Describes the compartment login feature.
pam_hpsec(5)	Extended authentication, account, password, and session service module for HP-UX.

6.4 Compartment Rules and Syntax

A compartment consists of a name and a set of rules. This section describes the four types of compartment rules:

- File system rules
- IPC rules
- Network rules
- Miscellaneous rules

Add rules to a rules file you create in the `/etc/cmpt` directory. You can edit this file using `vi` or a similar text editor. The rules file must have a `.rules` extension.

See *compartments*(5) for additional information.

6.4.1 Compartment Definition

Define compartments by configuring a name for each compartment, and associating one or more compartment rules with the compartment name. You can specify rules in any order.

The syntax for a compartment definition is as follows:

```
[sealed] [discover] compartment new_compartment_name { rules }
```

If the HP-UX ContainmentPlus product (version B.11.31.02 or later) is installed on the system, compartment definitions use the following format:

```
[sealed] [discover] [system] [blocked] compartment new_compartment_name { rules }
```

where:

`sealed`	(Optional) A process in this compartment cannot gain privileges or change compartments by calling `execve`.
`discover`	(Optional) Discover and automatically add rules so that compartment violations are overridden. This is a development feature to determine the rules necessary, and should not be used on a production system. See Section 6.7 for more information on this keyword.

`system`	(Optional) Indicates that this compartment shares the ownership of network interfaces with default compartments, such as the `init` compartment, and other compartments that are marked as `system` compartments. The ownership of network interfaces are typically specified by network interface rules. When a compartment is marked as a `system` compartment, all of the network interfaces that are configured to belong to this compartment are also considered as belonging to the `init` compartment and all other compartments that are marked as `system` compartments. The `init` compartment will be in favor of using these network interfaces for network communications, over using the other network interfaces. When a compartment is marked as `system` compartment, it also shares the connectivities through loopback interfaces with the `init` compartment. The `system` keyword is valid only if the HP-UX ContainmentPlus product (version B.11.31.02 or later) is installed on the system.
`blocked`	(Optional) Indicates that no processes can be launched in this compartment from other compartments, either through calling the `cmpt_change`() routine or through executing a binary file that is configured with a compartment name as one of its extended security attributes (see *setfilexsec*(1M)). The `blocked` keyword is valid only if the HP-UX ContainmentPlus product (version B.11.31.02 or later) is installed on the system.
`compartment`	Designates that the rule is a compartment definition.
`new_compartment_name`	The label associated with the new compartment. This label is case sensitive. For example, `compartmenta` and `CompartmentA` are different compartments.
{ }	Enclose the rules for this compartment.

In the following example, the `server_children` compartment is denied all access to any file system objects:

```
sealed compartment server_children {

/permission none /
}
```

In the following example, the `ifaces` compartment shares the ownership of `lan0` and `lan1` with the `init` compartment. The `init` compartment will be in favor of using `lan0` and `lan1` for network communications, over using the network interfaces that are owned by other compartments.

```
system compartment ifaces {
 interface lan0
 interface lan1
}
```

NOTE: The `INIT` compartment name is not case sensitive. `INIT`, `init`, and `Init` are all treated as the same compartment by the system.

Compartment specifications are preprocessed with `cpp()` before parsing begins. This is why you use `cpp()` directives such as `#include`, `#define`, `#ifdef`, and C-style comments to organize and document rules files.

6.4.2 File System Rules

File system rules govern access by processes to files and directories on the system. File system rules are inherited from a parent directory to all subdirectories and files within the parent, unless an explicit rule overrides inheritance.

By default, if no permissions are specified, all permissions are granted for a file system object.

When multiple file system rules are defined for the same pathname, the rules will be aggregated. That is, the union of the permissions is taken.

The syntax for file system rules is as follows:

`(permission|perm)` *`permission_list file_object`*

where:

`permission` or `perm` — Sets permissions for a file or directory.

permission_list — The types of permission you can apply to a file or directory are:

- `none`: Denies all permissions to a file or directory.
- `read`: Controls the read access to the object. If the object is a file, reading and executing the file is controlled. If the object is a directory, searching and listing the directory is controlled. Additionally, due to inheritance, reading of all files under the directory is controlled. Files must have read access in order to be opened for execution.
- `nread`: Controls `search` and `read` access to the `file_object`. The rule has an effect only if `file_object` is a directory. It allows processes in

the compartment not only to lookup in the directory (see the `nsearch` parameter), but also to list contents of the directory. Similar to the `nsearch` parameter, this access control is not inherited. Therefore, even if a directory is searchable and readable, any directory or file underneath it is not searchable or readable unless it is explicitly allowed.

The `nread` keyword is valid only if the HP-UX ContainmentPlus product is installed on the system.

- `write`: Controls the write access to the object. If the object is a file, writing to the file is controlled. If the object is a directory, due to inheritance, writing for all files under the directory is controlled.
- `create`: Controls the ability to create objects. This applies to directory objects only. This is inherited by all directories under the specified directory.
- `unlink`: Controls the ability to delete objects. This applies to directory objects only. This is inherited by all directories under the specified directory.
- `nsearch`: Controls the ability to search for an element if the `file_object` is a directory. This attribute is not inherited by subdirectories.

file_object The full path name of the file or directory.

For example:

```
/* deny all permissions except read to entire system */
perm read  /

/* except for this directory   */
perm read,write,create,unlink /var/opt/server

/* just read and write log files, not create them */
perm  read,write /var/opt/server/logs
```

NOTE: To grant any permission on a file system object, the compartment must have a minimum of read permission on every directory above that object. For example, to grant read and write permissions on `/var/opt/tmp/file1`, you must grant read permissions on `/var/opt/tmp`, `/var/opt`, `/var`, and `/`.

6.4.3 IPC Rules

Interprocess communication (IPC) rules govern how processes use interprocess communication methods between compartments. IPC communication methods include direct process-to-process communication or shared access to an IPC object. When an object is associated with a process, the object exists in the same compartment as the process that created it. You define compartment rules to describe the relationship between the process accessing the object and the object being accessed. When the rule describes two processes communicating with each other, you treat the second process as an object. The default behavior for IPC objects is that all operations between different compartments are prohibited unless explicitly allowed by a rule.

There are two types of IPC rules. The syntax for the first rule type is as follows:

```
(grant|access) (pty|fifo|uxsock|ipc) compartment_name
(grant|access) [pty][, fifo][, uxsock][, ipc] compartment_name
```

If the HP-UX ContainmentPlus product (version B.11.31.02 or later) is installed on the system, a new keyword `tl` is also supported and the first form of IPC rules uses the following format:

```
(grant|access) (pty|fifo|uxsock|ipc|tl) compartment_name
(grant|access) [pty][, fifo][, uxsock][, ipc]  [, tl] compartment_name
```

where:

Access

Specifies whether the rule is object-centric or subject-centric. The options are:

- `grant`: Specifies an object-centric rule. This rule allows processes in the compartment *`compartment_name`* to access the specified IPC mechanism in the current compartment.
- `access`: Specifies a subject-centric rule. This rule allows processes in the current compartment to access the specified IPC mechanism in the compartment *`compartment_name`*.

Method

Specifies the method of communication this rule applies to. The options are:

- `pty`: Specifies that the rule applies to `pty` used in interprocess communication.
- `fifo`: Specifies that the rule applies to FIFOs.
- `uxsock`: Specifies that the rule applies to UNIX domain sockets.
- `ipc`: Specifies that the rule applies to SYSV and POSIX IPC objects, such as shared memory, semaphores, and message queues.
- `tl`: Applies to Streams Local Transport Drivers that are used to communicate between processes.

  The `tl` keyword is valid only if the HP-UX ContainmentPlus product (version B.11.31.02 or later) is installed on the system. See *compartments*(4). The `tl` keyword only has effect when the `cmpt_restrict_tl` tunable is set to `1`. See *t_open*(3), *t_connect*(3), and *cmpt_restrict_tl*(5).

compartment_name	The name of the other compartment where processes in this compartment can communicate with.

When multiple IPC rules are defined for the same compartment, the rules will be aggregated. That is, the union of the IPC mechanisms is taken.

For example:

```
/* allow the children to access UNIX domain */
/* sockets created by the parent compartment */
```

```
grant  uxsock  server_children
```

The second type of IPC rule governs process access. The syntax for this type of rule is as follows:

`(send|receive) signal` *`compartment_name`*

where:

Direction — Specifies whether processes in the current compartment have access to view and alter process behavior from another specified compartment. The options are:

- `send`: Specifies a subject-centric rule. Allows processes in the current compartment to send signals and view process data in the compartment *`compartment_name`*.
- `receive`: Specifies an object-centric rule. Allows processes in the compartment *`compartment_name`* to send signals and view process data in the current compartment.

`signal` — Specifies that this rule applies to signals and process visibility.

compartment_name — The name of the other compartment where processes in the current compartment can have access to view process information or to be viewed from.

For example:

```
/* allow the parent to send signals to children */
   send signal server_children
```

6.4.4 Network Rules

Network rules control access between a process and a network interface, as well as between two processes using loopback communications. They do not control the communications through Streams Local Transport Drivers (see *cmpt_restrict_tl*(5) and the `tl` keyword).

These rules control the direction of network traffic (incoming, outgoing, or both) between the subject compartment and the target compartment specified in the rule. For loopback communications, the subject and target compartments should be of the processes that are communicating and not that of the interface being used for communication. Each rule is specified by protocol (TCP, UDP, or any raw protocol number) and the target compartment, and can optionally filter based on local or peer port numbers (TCP and UDP only). If an explicit rule does not match a communication attempt, the default is to deny communication.

If the HP-UX ContainmentPlus product is installed on the system, the default rule for access between two processes through loopback communications (excluding those through loopback interfaces) is also configurable through the `cmpt_allow_local` tunable. See *ifconfig*(1M) for more information about loopback interfaces.

See *cmpt_allow_local*(5) for more information upon installation of the HP-UX ContainmentPlus product.

The syntax for a network rule is as follows:

```
(grant|deny) (server|client|bidir) (tcp|udp|raw [protonum] )
[port port_num] [peer[portport]] compartment_name
```

If the HP-UX ContainmentPlus product (version B.11.31.02 or later) is installed on the system, the network rules using the following formats are also supported:

```
(grant-local|deny-local) (server|client|bidir) (tcp|udp|raw [protonum] )
[port port_num] [peer[portport]] compartment_name
```

where:

Access

Grants or denies the compartment access to the network traffic in the specified compartment. The options are:

- `grant`: Allows access to the network (both access between a process and a network interface, as well as between two processes using loopback communications) described by this rule.
- `deny`: Denies access to the network (both access between a process and a network interface, as well as between two processes using loopback communications) described by this rule.
- `grant-local`: Allows access described by this rule between two processes using loopback communications. The `grant-local` keyword is valid only if the HP-UX ContainmentPlus product is installed on the system**.**
- `deny-local`: Denies access described by this rule between two processes using loopback communications. The `deny-local` keyword is valid only if the HP-UX ContainmentPlus product is installed on the system.

Direction

Specifies which direction the rule applies to. The options are:

- `server`: This rule applies to inbound requests only. For TCP, only incoming connections are controlled by this

	rule. For UDP and RAW, this rule applies to all inbound packets. • `client`: This rule applies outbound requests only. For TCP, only connection initiations are controlled by this rule. For UDP and RAW, this rule applies to all outbound packets. • `bidir`: This rule applies to both inbound and outbound requests. For TCP, connections initiated and received by the endpoint are controlled by this rule. For UDP and RAW, this rule applies to all packets passing through the endpoint.
Protocol	Specifies the networking protocol that applies to this rule. The options are: • `tcp`: This rule applies to the TCP protocol. • `udp`: This rule applies to the UDP protocol. • `raw`: This rule applies to any other protocol in the INET domain.
protonum	The protocol number specified for this rule. The `protonum` option is relevant only for `raw` specification.
`port`	(Optional) Specifies that this rule applies to a specific port.
port	Identifies the port specified in this rule.
`peer`	(Optional) The port information applies to the peer endpoint involved in the communication for this rule.
compartment_name	Specifies the name of the compartment that is the target of the rule. This is usually the interface compartment name, but can also be specified as another compartment to indicate a loopback communication.

For example:

```
/* allow all inbound TCP connections(any port)from interfaces labeled lancmpt1  */
grant server tcp lancmpt1
/* allow DNS client lookups (both TCP and UDP) through interface labeled lancmpt1 */
grant client tcp port 53 lancmpt1
grant bidir udp port 53 lancmpt1
/* allow only outbound telnet connections through interface labeled ifacelan0 */
grant client tcp peer port 23 ifacelan0
/* allow all TCP traffic except inbound telnet through interface labeled ifacelan0 */
/* the following two lines can be specified in either order */
grant bidir tcp ifacelan0
deny server tcp port 23 ifacelan0
/* allow inbound web server traffic through interface lan1cmpt */
```

```
grant server tcp port 80 lan1cmpt
```

The network rules control how a process can communicate on a given port and interface, as well as how the process can bind to a port or address. In other words, the network rules are enforced at the time a communication takes place, and when a process calls the bind routine. The multibind facility enables processes to attach to `IFADDR_ANY` on a specific port in different compartments having disjoint set of interface rules. When multiple network rules are defined for the same compartment, the rules will be aggregated. That is, the union of all the rules is taken.

For more information about network rules, see *compartments*(4).

6.4.5 Miscellaneous Rules

These are rules that do not fit neatly into any other rules category.

Network Interface Rules A network interface rule specifies the compartment that an interface belongs to. A network interface that is not in a compartment cannot be brought on line.

NOTE: For stricter security policies, configure network interfaces in separate compartments from those assigned to processes. Define rules for network access for each compartment accordingly. Equal compartments are always granted full access to one another.

The network interface rule syntax is as follows:

```
compartment compartment_name {
interface interface_or_ip[,interface_or_ip][...]
}
```

where:

`interface`	Specifies that this is an interface definition.
interface_or_ip[,interface_or_ip][...]	A comma-separated list of interface names, IP address, or range of IP addresses. IP addresses or ranges can be specified as IPv4 addresses or IPv6 addresses with an optional mask.

For example:

```
compartment iface0 {
/* Define the compartment for the network interface lan0  */
 interface  lan0
/* All addresses in the range 192.168.0.0-192.168.0.255 */
  interface  192.160.0.0/24
}
compartment other_ifaces {
/* Define the compartment for two of the other network interfaces */
interface lan1,lan5
```

NOTE: When APA is used in `LAN MONITOR` mode, the following rules must be met:

- The primary interface, `lan0`, must be assigned to the proper compartment.
- The secondary interface, `lan1`, is either not assigned to any compartment or is assigned to the same compartment as `lan0`.
- The aggregate interface, `lan900`, is either not assigned to any compartment or is assigned to the same compartment as `lan0`. HP recommends that you leave `lan900` unassigned in case APA changes the naming scheme.

In this example, `lan0` and `lan1` are aggregated into `lan900`.

For more information on APA, see *apa*(7).

Privilege Limitation Rules A privilege limitation rule controls privilege inheritance. Any privilege named in a privilege limitation rule cannot be obtained when calling `execve(2).`

The syntax for privilege limitation rules is:

```
disallowed privileges  privilege[,privilege[...]]
```

where:

`disallowed privileges`	Specifies this as a privilege limitation rule.
privilege[,privilege[...]]	A comma-separated list of privileges. You can use the following additional keywords: • `all`: disallows all privileges • `none`: allows all privileges • `!`: denotes except

For example:

```
/* Disallow all privileges except mount. */
disallowed privileges all,!mount

/* Disallow mount only. */
disallowed privileges none,mount
```

If privilege limitation rules are not specified for a compartment, the default privilege limitation is `basicpolicy,mknod` for every compartment except the `INIT` compartment. The `INIT` compartment default privilege limitation is `none`.

When multiple disallowed privilege rules are defined, the rules will be aggregated. Refer to *priv_str_to_set*(3) for information on how the privileges string will be aggregated to the privilege set.

6.4.6 Example Rules File

An example rules file is located in `/etc/cmpt/examples/sendmail.example`.

6.5 Configuring Compartments

This section discusses the following topics:

- Activating compartments (Section 6.5.1)
- Defining a compartment configuration (Section 6.5.2)
- Running an application in a compartment (Section 6.5.3)
- Login directly in a compartment (Section 6.5.4)

6.5.1 Activating Compartments

To activate compartment rules on the system, follow these steps:

1. Plan the compartment rules. See Section 6.2 for more information.

   TIP: HP recommends you plan the compartment rules configuration carefully. After you have edited the configuration and implemented it on a production system, it becomes difficult to change. When you change a compartment configuration, you must make changes to user procedures, scripts, and tools.

2. Create compartment rules. See Section 6.4 for instructions on completing this step and for a complete description of compartment rules syntax.
3. (Optional) Preview the compartment rules by entering the following command:

   ```
   # setrules -p
   ```

   The `-p` option parses the configured rules list and reports any discrepancies in syntax and semantics. HP recommends that you follow this step before enabling compartment rules on the system.

4. (Optional) Make backup copies of the compartment configuration files. Either put these files outside the `/etc/cmpt` directory or omit the `.rules` suffix. Doing this lets you easily revert to the starting point if an editing problem occurs.
5. Enable the compartments feature by entering the following command:

   ```
   # cmpt_tune -e
   ```

6. Reboot the system. This step is mandatory.

TIP: Keep the backup files; this makes it easier to revert to a prior configuration.

6.5.2 Defining a Compartment Configuration

You can create new compartments and modify existing compartments without rebooting the system. If you enable or disable the compartment feature, or completely remove a compartment, you must reboot the system. However, if you remove all rules associated with a compartment and all references to that compartment, you can leave the compartment on the system until the next reboot.

See Section 6.5.2.2 for more information about the implications of changing the name of a compartment.

You can add new compartment rules, delete unneeded rules, and modify existing rules. You can also change the names of existing compartments.

The application containment wizard, `contain`, can be used to simplify this configuration process. See *compartment_login*(5) for more information.

To following sections describe how to modify compartment configuration.

6.5.2.1 Changing Compartment Rules

1. (Optional) Make temporary backup copies of the configuration files you plan to modify. Either put these files outside the `/etc/cmpt` directory or omit the `.rules` suffix. Doing this lets you easily revert to the starting point if an editing problem occurs.
2. Use the following command to examine the current compartment rules:

   ```
   # getrules
   ```
3. Create or modify compartment rules. See Section 6.4 for instructions on completing this step and for a complete description of compartment rules syntax.
4. (Optional) Preview the compartment rules by entering the following command:

   ```
   # setrules -p
   ```

   The `-p` option parses the configured rules list and reports any discrepancies in syntax and semantics. HP recommends that you follow this step before enabling compartment rules on the system.
5. (Optional) Make backup copies of the compartment configuration files.
6. Run the `setrules` command to load the configured rules:

   ```
   # setrules
   ```

6.5.2.2 Changing Compartment Names

You can change the names of compartments. However, changing the name of a compartment can affect applications that are already configured with the existing compartment names. If you change the name of a compartment, you must reconfigure any applications configured in that compartment as well.

NOTE: If you rename a compartment, you have essentially created a new compartment and removed the compartment with the old name. You must change all references to see the new compartment. The old compartment continues to exist on the system until a reboot.

6.5.3 Running an Application in a Compartment

You can configure an application to run in a particular compartment by using one of the following options:

- The `setfilexsec` command to configure the compartment attribute of a binary file. For example, to configure the application `apple` into the compartment `fruit`, enter the following command:

  ```
  # setfilexsec -c fruit apple
  ```

- HP-UX RBAC,see Section 8.5.5.

6.5.4 Login Directly to a Compartment

The compartment login configuration enables users and administrators to login directly to a compartment. It provides a mechanism to set controls on those users that are allowed to login to a service running in a specified compartment or prevent access to the system based on previously configured authorization information.

NOTE: The Compartment Login feature is only supported on standard systems, it is not supported on trusted systems.

For more information, see *HP-UX Compartment Login using Secure Shell (SSH)*:

www.hp.com/go/hpux-security-docs

Click **HP-UX 11i Security Containment Software**.

6.6 Troubleshooting Compartments

If something is not working on the system and you suspect the problem is occurring because of the compartment structure, you can check the compartment rules as follows.

Problem 1: Access is not being controlled according to the compartment rules I configured. Solution: the rules may not be set in the kernel. To check whether the rules are set in the kernel, follow these steps:

1. Use the following command to list the valid compartment rules in the kernel.

   ```
   # getrules
   ```

2. Use the following command to list all rules configured on the system, including rules that have not been loaded into the kernel.

   ```
   # setrules -p
   ```

3. Compare the output of the two commands. If they are the same, all rules are loaded into the kernel. If the output differs, you need to load rules into the kernel.
4. Use the following command to load rules into the kernel. :

   ```
   # setrules
   ```

Problem 3: Access to a file is not functioning properly. Solution: If multiple hard links point to this file, the compartment rules configuration may contain inconsistent rules for accessing the file. To check for inconsistencies, follow these steps:

1. Execute the following command:

   ```
   # vhardlinks
   ```

   If the output shows an inconsistency, go on to step 2.

2. Modify the rules to remove the inconsistency. Follow the procedure described in Section 6.5.2.

Problem 4: Network server rules do not appear in getrules output. Solution: Because of the way rules are managed internally, network server rules for a given compartment can be listed in the target compartment output of the `getrules` command.

For example:

```
/* telnet compartment rule to allow incoming telnet requests through compartment labeled ifacelan0
*/
grant server tcp port 23 ifacelan0
```

If this rule is specified, it appears listed under the `ifacelan0` compartment output of `getrules`.

```
ACCESS              PROTOCOL     SRCPORT     DESPORT      DESCMPT
Grant client        tcp          0           23           telnet
```

6.7 Using Discover Mode to Generate Initial Compartment Configuration

A compartment definition can be tagged with the keyword **discover**. See Section 6.4.1. The `discover` keyword instructs the system to discover all of the rules necessary to make the application function correctly. This feature is intended to only be used in a test environment.

To use `discover` mode, mark a compartment as `discover` and run the application as you normally would. The system identifies all resource accesses and creates the required rules.

After the initial execution of the application, use the `getrules -m` *`compartment_name`* command to generate a machine readable version of rules.

NOTE: The system does not discover `nread` and `grant-local`. The system discovers `read` rules for `nread` access and discovers `grant` rules for `grant-local` access.

The system generated rules are required to make the application function successfully in the test environment, but may need to be generalized. For example, the system may generate a rule that involves a port number in anonymous port range, where the kernel, not the application, selects the port number. When the application is run again, it may end up with a different port number, requiring a different rule. The rule may need to be generalized such that either all ports or at least the port numbers in the anonymous port range are specified.

6.8 Compartments in HP Serviceguard Clusters

If you use compartments with HP Serviceguard, you must configure all Serviceguard daemons in the default `INIT` compartment. However, you can configure Serviceguard packages in other compartments. See the latest editions of *Managing Serviceguard* and *Using Serviceguard Extension for RAC* for daemons required in Serviceguard and Serviceguard extensions for Oracle Real Application Cluster (RAC).

Serviceguard packages can belong to specific compartments. Applications monitored as part of a Serviceguard package can also be configured in specific compartments. When you set up the compartment for a package, be sure that the resources required by that package (such as volume groups, file systems, network addresses, and so on) are accessible by that compartment. Compartment rules are node-specific and do not get carried over during Serviceguard failover operations. To ensure proper operation after a failover, all nodes in the cluster must have identical compartment configurations.

When a primary LAN interface fails over to a standby LAN interface, the compartment label of the primary interface is automatically copied over to the standby interface as long as the standby is not online. If the standby interface is already configured online, the standby interface and the primary interface must be configured in the same compartment to fail over successfully. If the standby interface is configured in a different compartment from the primary interface, but is offline at the time of the failover, the standby interface is updated to the primary interface compartment configuration when the interface fails over.

To maintain proper Serviceguard operations when deploying compartments in HP Serviceguard nodes or packages:

- Do not modify the `INIT` compartment specifications in any way.
- Ensure `inetd` runs in the `INIT` compartment.
- Ensure that all Serviceguard daemons in a cluster run in the INIT compartment. For example, the daemons for Serviceguard Version A.11.16 include `cmclconfd`, `cmcld`, `cmlogd`, `cmlvmd`, `cmomd`, and `cmsnmpd`. See *Managing Serviceguard* for a list of all Serviceguard daemons.
- Ensure that all Serviceguard cluster requirements are met for Serviceguard Extensions for RAC clusters. Additionally, clusters with Serviceguard Extension for RAC Version A.11.16 need the `cmsmgd` daemon to run in the `INIT` compartment. RAC processes must have access to the `libnmapi2` library, and must communicate with `cmsmgd`. See *Using Serviceguard Extension for RAC* for required daemons and libraries.
- Do not configure standby LAN interfaces in a compartment.
- Set up the compartments and rules identically on all nodes in the cluster. Compartments and rules are specific to a system and do not get carried over when a system fails over.

NOTE: If a standby interface is configured in a compartment, running the `setrules` command applies this compartment to the standby interface even if it has been successfully switched from a primary interface. If the configured standby interface compartment does not match the primary interface compartment, the primary interface compartment is overwritten when you run `setrules`. This can cause security violations.

There are no changes made to the Serviceguard scripts to facilitate the use of compartments, fine-grained privileges, or RBAC.

7 Fine-Grained Privileges

This chapter describes the fine-grained privileges feature of HP-UX 11i . This chapter addresses the following topics:

- Overview (Section 7.1)
- Fine-grained privileges components (Section 7.2)
- Available privileges (Section 7.3)
- Configuring applications with fine-grained privileges (Section 7.4)
- Security implications of fine-grained privileges (Section 7.5)
- Fine-grained privileges in HP Serviceguard Clusters (Section 7.6)
- Troubleshooting fine-grained privileges (Section 7.7)

7.1 Overview

The UNIX operating system traditionally uses an "all or nothing" privilege model, in which superusers (those with effective `UID=0`, such as the root user) have virtually unlimited power, and other users have few or no special privileges.

HP-UX provides several legacy methods of delegating limited powers, including restricted *smh*(1M), the privilege groups described in *privgrp*(4), the `shutdown.allow` file described in *shutdown*(1M), and the `cron.allow` file described in *crontab*(1).

These legacy methods can be replaced by the use of fine-grained privileges and the HP-UX RBAC access control framework.

The HP-UX fine-grained privilege model splits the powers of superusers into a set of privileges. Fine-grained privileges are granted to processes. Each privilege grants a process that possesses that privilege the right to a certain set of restricted services provided by the kernel.

See *privileges*(5) for more information.

7.2 Fine-Grained Privileges Components

The fine-grained privileges feature of HP-UX 11i include configuration files, commands, and manpages. You can use these components to configure and administer fine-grained privileges.

7.2.1 Commands

Table 7-1 briefly describes the fine-grained privileges commands.

Table 7-1 Fine-Grained Privileges Commands

Commands	Description
`setfilexsec`	Sets security attributes of binary files. The attributes include retained privileges, permitted privileges, compartment, and the privilege start flag.
`getfilexsec`	Displays security attributes associated with binary executable files. The attributes include retained privileges, permitted privileges, compartment, and security attribute flags.
`getprocxsec`	Displays security attributes associated with a running processes. The attributes include the effective privilege set, retained privilege set, permitted privilege set, `euid`, and the compartment name.

7.2.2 Manpages

Table 7-2 briefly describes the fine-grained privileges manpages.

Table 7-2 Fine-Grained Privileges Manpages

Manpage	Description
privileges(5)	Overview of HP-UX privileges.
privileges(3)	Describes fine-grained privileges interfaces.
setfilexsec(1M)	Describes `setfilexsec` functionality and syntax.
getfilexsec(1M)	Describes `getfilexsec` functionality and syntax.
getprocxsec(1M)	Describes `getprocxsec` funtionality and syntax.

7.3 Available Privileges

Fine-grained privileges are primarily targeted for developers. However, an administrator may still need to understand the privileges to understand how such applications work and to find if any unauthorized applications have gained privileges.

Table 7-3 lists the privileges and their primary purposes.

Table 7-3 Available Privileges

Privilege	Description
`PRIV_ACCOUNTING`	Allows a process to control the process accounting system.
`PRIV_AUDCONTROL`	Allows a process to start, modify, and stop the auditing system.
`PRIV_CHANGECMPT`	Grants a process the ability to change its compartment.
`PRIV_CHANGEFILEXSEC`	Allows a process to grant privileges to binaries.
`PRIV_CHOWN`	Allows a process to access the `chown()` system calls.

Table 7-3 Available Privileges *(continued)*

Privilege	Description
`PRIV_CHROOT`	Allows a process to change its root directory.
`PRIV_CHSUBJIDENT`	Allows a process to change its UIDs, GIDs, and group lists. Also allows a process to leave the `suid` or `sgid` bits set on the file when the `chown()` system call is used.
`PRIV_CMPTREAD`	Allows a process to open a file or directory for reading, executing, or searching, bypassing compartment rules that otherwise would not permit these operations.
`PRIV_CMPTWRITE`	Allows a process to write to a file or directory, bypassing compartment rules that otherwise would not permit this operation.
`PRIV_COMMALLOWED`	Allows a process to override compartment rules in the IPC and networking subsystems.
`PRIV_CORESYSATTR`	Enables a process to manage system attributes including the setting of tunables and modifying user quotas. This privilege is valid only when the HP-UX ContainmentPlus product (version B.11.31.02 or later) is installed on the system.
`PRIV_DACREAD`	Allows a process to override all discretionary read, execute, and search access restrictions.
`PRIV_DACWRITE`	Allows a process to override all discretionary write access restrictions.
`PRIV_DEVOPS`	Allows a process to do device administrative operations that are not specific to streams-based or pseudo terminals. **NOTE:** If the HP-UX ContainmentPlus product (version B.11.31.02 or later) is installed on the system, the `PRIV_DEVOPS` privilege is divided into `PRIV_RDEVOPS` and `PRIV_PTYOPS`. See "Compatibility Information for Divided Privileges" (page 135).
`PRIV_DLKM`	Allows a process to load a kernel module, get information about a loaded kernel module, and change global search paths for a dynamically loadable kernel module.
`PRIV_FSINTEGRITY`	Allows a process to perform disk operations such as removing or modifying the size or boundaries of disk partitions, or to import and export an LVM volume group across the system.
`PRIV_FSMOUNT`	Allows a process to mount and unmount a file system using the `mount`() and `umount`() system calls. This privilege is valid only when the HP-UX ContainmentPlus product (version B.11.31.02 or later) is installed on the system.
`PRIV_HOSTATTR`	Enables a process to modify the host name and domain name. This privilege is valid only when the HP-UX ContainmentPlus product (version B.11.31.02 or later) is installed on the system.

Table 7-3 Available Privileges *(continued)*

Privilege	Description
PRIV_LIMIT	Allows a process to set resource and priority limits beyond the maximum limit values.
PRIV_LOCKRDONLY	Allows a process to use the lockf() system call to lock files opened with read-only permission.
PRIV_MKNOD	Allows a process to create character or block special files using the mknod() system call.
PRIV_MLOCK	Allows a process to access the plock system call.
PRIV_MOUNT	Allows a process to mount and unmount a file system using the mount() and umount() system calls. **NOTE:** If the HP-UX ContainmentPlus product (version B.11.31.02 or later) is installed on the system, the PRIV_MOUNT privilege is divided into PRIV_FSMOUNT and PRIV_SWAPCTL. See "Compatibility Information for Divided Privileges" (page 135).
PRIV_MPCTL	Allows a process to change processor binding, locality domain binding, or launch policy.
PRIV_NETADMIN	Allows a process to perform network administrative operations including configuring the network routing tables and querying interface information.
PRIV_NETPRIVPORT	Allows a process to bind to a privileged port. By default, port numbers 0-1023 are privileged ports.
PRIV_NETPROMISCUOUS	Allows a process to configure an interface to listen in promiscuous mode.
PRIV_NETRAWACCESS	Allows a process to access the raw internet network protocols.
PRIV_OBJSUID	Allows a process to set the suid or sgid bits on any file if the process has the OWNER privilege. It also allows a process to change the ownership of a file without clearing the suid or sgid bits, provided that the process is allowed to change the ownership of the file.
PRIV_OWNER	Allows a process to override all restrictions with respect to UID matching the owner of the file or resource.
PRIV_PSET	Allows a process to change the system pset configuration.
PRIV_PTYOPS	Allows the process to do administrative operations that are streams-based or pseudo terminal specific. This privilege is valid only when the HP-UX ContainmentPlus product (version B.11.31.02 or later) is installed on the system.

Table 7-3 Available Privileges *(continued)*

Privilege	Description
`PRIV_RDEVOPS`	Allows the process to do device administrative operations that are non-pseudo terminal specific. This privilege is valid only when the HP-UX ContainmentPlus product (version B.11.31.02 or later) is installed on the system.
`PRIV_REBOOT`	Allows a process to perform reboot operations.
`PRIV_RTPRIO`	Allows a process to access the `rtprio`() system call.
`PRIV_RTPSET`	Allows a process to control RTE psets.
`PRIV_RTSCHED`	Allows a process to set POSIX.4 real-time priorities.
`PRIV_RULESCONFIG`	Allows a process to add and modify compartment rules on the system.
`PRIV_SELFAUDIT`	Allows a process to generate auditing records for itself using `audwrite`() system call.
`PRIV_SERIALIZE`	Allows a process to use the `serialize`() system call force a target process to run serially with other processes marked for serialization.
`PRIV_SPUCTL`	Allows a process to do certain administrative operations in the Instant Capacity product.
`PRIV_SWAPCTL`	Allows a process to manage swap space using the `swapctl`() system call. This privilege is valid only when the HP-UX ContainmentPlus product (version B.11.31.02 or later) is installed on the system.
`PRIV_SYSATTR`	Allows a process to manage system attributes, including the setting of tunables, modifying the host name, domain name, and user quotas. **NOTE:** If the HP-UX ContainmentPlus product (version B.11.31.02 or later) is installed on the system, the `PRIV_SYSATTR` privilege is divided into `PRIV_CORESYSATTR` and `PRIV_HOSTATTR`. See "Compatibility Information for Divided Privileges" (page 135).
`PRIV_SYSNFS`	Allows a process to perform NFS operations like exporting a file system, the `getfh`() system call, NFS file locking, revoking NFS authentication, and creating an NFS kernel daemon thread.
`PRIV_TRIALMODE`	Allows a process to log trial mode information to the `syslog` file.

7.3.1 Compatibility Information for Divided Privileges

If the HP-UX ContainmentPlus product (version B.11.31.02 or later) is installed on the system, the `PRIV_SYSATTR`, `PRIV_MOUNT` and `PRIV_DEVOPS` privileges are each divided into two privileges. The `PRIV_SYSATTR` privilege is divided into

PRIV_CORESYSATTR and PRIV_HOSTATTR. The PRIV_MOUNT privilege is divided into PRIV_FSMOUNT and PRIV_SWAPCTL. The PRIV_DEVOPS privilege is divided into PRIV_RDEVOPS and PRIV_PTYOPS.

This new privilege model allows applications, when explicitly developed to be aware of HP-UX privileges (see *privileges*(5)), to have finer control over the administrative capabilities that were controlled by the PRIV_SYSATTR, PRIV_MOUNT and PRIV_DEVOPS privileges.

System calls that manage a system's host and domain names (see *setdomainname*(2), *sethostname*(2), and *setuname*(2)) now require the PRIV_HOSTATTR privilege.

System calls that manage a system's swap space (see *swapctl*(2) and *swapon*(2)) now require the PRIV_SWAPCTL privilege.

System calls that manage streams-based terminals (see *ldterm*(7)) now require the PRIV_PTYOPS privilege.

The above system calls will return -1 with errno set to either EPERM or EACCESS if the required privilege is not possessed by the calling process.

To maintain backward compatibility for HP-UX privileges aware applications in the new privilege model, the string representation of the PRIV_SYSATTR, PRIV_DEVOPS, and PRIV_MOUNT privileges will continue to be supported as compound privileges [PRIV_CORESYSATTR and PRIV_HOSTATTR], [PRIV_RDEVOPS and PRIV_PTYOPS], and [PRIV_SWAPCTL and PRIV_FSMOUNT] in the user space. All HP-UX core kernel modules and commands have been updated to support the new privileges. This ensures standard and typical HP-UX privileges aware applications to continue to work in the new privilege model without requiring any changes unless you want to take advantage of the new privilege model to gain finer control.

7.4 Configuring Applications with Fine-Grained Privileges

Applications that are written or modified to support fine-grained privileges are called privilege-aware applications. You must register privilege-aware applications using the setfilexsec command. Once registered, the security attributes associated with a binary file are stored in a configuration file and maintain persistence across reboot. This is normally done for you when you install and configure privilege-aware applications using the SD-UX utilities.

Older HP-UX applications, or legacy applications, are not privilege-aware. You can configure legacy applications that run with UID=0 to run with fine-grained privileges. To configure legacy applications using HP-UX RBAC, see Section 8.5.4.

TIP: HP recommends you use HP-UX RBAC to configure applications that require variable privileges to run.

NOTE: Some of the fine-grained privileges are divided into more granularity. If the HP-UX ContainmentPlus product (version B.11.31.02 or later) is installed on the system, the `PRIV_SYSATTR` , `PRIV_MOUNT`, and `PRIV_DEVOPS` privileges are each divided into two privileges. By using the new privileges, a process can now allow a subset of the operations while disallowing the other. See *privileges*(5) and "Compatibility Information for Divided Privileges" (page 135).

To configure security attributes for a privilege-aware application, use the `setfilexsec` command as follows:

```
# setfilexsec [options] filename
```

The `setfilexsec` command is meant to assign privileges to binaries on a local file system. Binaries that are obtained from a network file systems (NFS) should not be assigned privileges because if the file is modified by a different system (directly on the NFS server), the extended attributes set by `setfilexsec` are not removed.

The options for `setfilexsec` are as follows:

`-d`	Deletes any security information for this file from the configuration file and the kernel.
`-D`	Deletes any security information for this file from the configuration file only. Used to clear security information for a deleted file.
`-r`	Add or change minimum retained privileges.
`-R`	Add or change maximum retained privileges.
`-p`	Add or change minimum permitted privileges.
`-P`	Add or change maximum permitted privileges.
`-f`	Sets the security attribute flags.

The `getfilexsec` command displays the extended attributes of a binary file, set with the `setfilexsec` command.

```
# getfilexsec filename
```

7.4.1 Privilege Model

Each process has three privilege sets associated with it:

- Permitted Privilege Set

  The maximum set of privileges a process can raise. The process can drop any privilege from this set, but cannot add any privileges to this set. Privileges from this set can be added to the effective privilege set of the process.

- Effective Privilege Set

  The set of currently active privileges for a process. A privilege-aware process can modify effective privilege set to keep only the necessary privileges in this set at any given time. The process can remove any privilege from the effective privilege set, but can only add privileges from the permitted privilege set.

  The effective privilege set is always a subset of the permitted privilege set.

- Retained Privilege Set

  The set of privileges retained when a process calls the `execve` system call. The process can remove any privilege from this set, but cannot add privileges to this set.

  The retained privilege set is always a subset of the permitted privileges set.

The first process, `init`, starts with a small set of privileges. It then creates other processes that execute other binaries using `exec` family calls (`execv`, `execve`, and so on). During this `exec` call, the extended attributes of the binary, the attributes set with `setfilexsec` command, may cause these processes to gain privileges that their parent process do not have, or lose the privileges that the parent process had. For instance, if a binary has a permitted minimum of `DACREAD` (`setfilexsec –p DACREAD` has been performed on the binary), the new process will have the `DACREAD` privilege whether or not the parent process had that privilege. On the other hand, if process already has the `DACREAD` privilege, but if the binary it executes does not have this privilege in permitted max (for example, `setfilexsec -P none ….` has been performed on the file already), it would lose the privilege as a side-effect of executing the binary.

7.4.2 Compound Privileges

Compound privileges are a shorthand way of specifying a predefined set of simple privileges.

The following are compound privileges:

- `BASIC`

  Basic privileges available to all processes by default. Processes may drop one or more privileges from this set.

- `BASICROOT`

  Basic and privileges and privileges that provide powers usually associated with UID=0.

- `POLICY`

  Policy override privileges and policy configuration privileges. Policy override privileges override compartment rules. Policy configuration privileges control how privileges are configured.

For a complete list of the privileges in each of the compound privileges, see *privileges*(5).

7.5 Security Implications of Fine-Grained Privileges

Fine-grained privileges are not propagated across distributed systems; they are applied only on the local system. For example a process on one system that has `PRIV_DACREAD` and `PRIV_DACWRITE` cannot override discretionary restrictions on another system to read or write to a file.

7.5.1 Privilege Escalation

In certain situations, if you grant a process a certain privilege or set of privileges, that process can gain additional privileges that were not explicitly granted to it. This is called privilege escalation. For example, a process with the `PRIV_DACWRITE` privilege can overwrite critical operating system files and, in the process, can grant itself additional fine-grained privileges.

7.6 Fine-Grained Privileges in HP Serviceguard Clusters

Privilege-aware applications can be monitored by HP Serviceguard. There are no changes to Serviceguard package configuration files or Serviceguard package management to support fine-grained privileges. No changes were made in Serviceguard scripts to facilitate the use of fine-grained privileges.

To maintain proper Serviceguard operations when deploying HP-UX 11i fine-grained privileges to Serviceguard nodes or packages:

- Ensure root (`UID=0`) has full privileges in the `INIT` compartment.
- Ensure fine-grained privileges implementations do not create security risks for Serviceguard clusters.

7.7 Troubleshooting Fine-Grained Privileges

If something is not working on the system and you suspect the problem is occurring because of fine-grained privileges, you can check the fine-grained privileges configuration as follows.

Problem 1: Even though fine-grained privileges are assigned to a binary file, processes that use exec() to access the binary are not receiving the assigned fine-grained privileges. Solution: Check for one of the following situations.

- Is the file in question a script?

  Any fine-grained privileges assigned to shell scripts are ignored.

- Has the file changed since the fine-grained privileges were assigned?

  When a file is modified, its fine-grained privilege attributes are lost. Run the following command either before or after you modify the file:

  ```
  # setfilexsec -d filename
  ```

  Next, add the privilege attributes you want assigned to the file.

See *setfilexsec*(1M) for more information about troubleshooting fine-grained privileges.

Problem 2: A process has privileges it should not have, or does not have privileges it should have. Solution: Use the `getprocxsec` command to determine what privileges a process has:

```
# getprocxsec -per pid
```

This command displays the permitted, effective, and retained privilege sets for the process. For more information, see *getprocxsec*(1M)

If the process does not have the correct privileges, configure the binary file that created this process with the correct privileges. See "Configuring Applications with Fine-Grained Privileges" for more information.

Part III Protecting Identity

In modern day global enterprise companies, managing identity is not an easy task, especially as identity management requirements grow to include employees, contractors, partners and suppliers across many countries with various privacy protection laws and regulation. HP-UX 11i simplifies user authentication and access management, while auditing all privileged actions that take place.

This section discusses the following topics:

- HP-UX Role-Based Access Control (Chapter 8)
- Audit administration (Chapter 9)

8 HP-UX Role-Based Access Control

The information in this chapter describes HP-UX Role-Based Access Control (HP-UX RBAC). This chapter addresses the following topics:

- Overview (Section 8.1)
- Access control basics (Section 8.2)
- HP-UX RBAC components (Section 8.3)
- Planning the HP-UX RBAC deployment (Section 8.4)
- Configuring HP-UX RBAC (Section 8.5)
- Using HP-UX RBAC (Section 8.6)
- Troubleshooting HP-UX RBAC (Section 8.7)

8.1 Overview

Security, especially platform security, has always been an important issue for enterprise infrastructure. Even so, many organizations often neglected or overlooked such security concepts as individual accountability and least privilege in the past. However, recently introduced legislation in the United States including the Health Insurance Portability and Accountability Act (HIPAA) and the Sarbanes-Oxley Act has helped to highlight the importance of these security concepts.

Most enterprise environments have systems administered by multiple users. Typically, this is accomplished by providing the administrators with the password to a common, shared account, known as root. While the root account simplifies access control management by enabling administrators with the root password to perform all operations the root account also presents several inherent obstacles for access control management, for example:

- After providing administrative users with the root password, there is no easy way to further constrain those users.
- In the best case, revoking access for a single administrator requires changing the common password and notifying other administrators. More realistically, simply changing the password is probably not sufficient to effectively revoke access because alternative access mechanisms might have already been implemented.
- Individual accountability with a shared root account is virtually impossible to achieve. Consequently, proper analysis after a security event becomes difficult, and in some cases impossible.

The HP-UX Role-Based Access Control (RBAC) feature resolves these obstacles by providing the capability to assign sets of tasks to ordinary, but appropriately configured, user accounts. HP-UX RBAC also mitigates the management overhead associated with assigning and revoking individual authorizations on a per-user basis.

HP-UX RBAC offers the following features:

- Predefined configuration files specific to HP-UX, for a quick and easy deployment
- Flexible re-authentication via Plugable Authentication Module (PAM), to allow restrictions on a per command basis
- Integration with HP-UX audit system, to produce a single, unified audit trail
- Pluggable architecture for customizing access control decisions

8.2 Access Control Basics

The goal of an access control system is to limit access to resources based on a set of constraints. Typically, these constraints and their associated attributes fit into the following categories:

- Subject: The entity attempting to access the resource. In the context of an operating system, the subject is commonly a user or a process associated with a user.
- Operation: An action performed on a resource. An operation can correspond directly to an application or a command. In the case of HP-UX RBAC, the operation is a dot-separated, hierarchical string, such as `hpux.user.add`.
- Object: The target of the operation, which is often the same as the end resource, but which can be different.

An access control request can be thought of as a question combining the previous elements, where the response to the question (usually allow or deny) determines whether access to the resource is granted. For example:

Is the user `ron` authorized to perform the operation `hpux.fs.mount` on the object`/dev/dsk/c0t1d0`?

Often, the term authorization is used as a synonym for access control. In HP-UX RBAC, authorization refers to the ability to perform an operation on an object. As shown in Table 8-1, a user can have a set of authorizations, each of which allows access to a resource.

Table 8-1 Example of Authorizations Per User

Operation Component of Authorization	**Users**			
	ron	*lisa*	*jim*	*liz*
`hpux.user.add`				
`hpux.user.delete`				
`hpux.user.modify`				
`hpux.user.password.modify`	•	•	•	•
`hpux.network.nfs.start`	•			

Table 8-1 Example of Authorizations Per User *(continued)*

Operation Component of Authorization	Users			
hpux.network.nfs.stop	•			
hpux.network.nfs.config	•			
hpux.fs.backup	•	•		
hpux.fs.restore	•	•		

NOTE: Table 8-1 shows only the operation element of the authorizations—not the object element of the authorizations.

8.2.1 Simplifying Access Control with Roles

In addition to the basic principals of access control discussed in the preceding overview, this section addresses how access control policy is represented and how decisions are made.

The preceding overview of access control does not address how access control policy is represented and how decisions are made. One approach is to simply maintain a list of users and the authorizations (operation, object pairs) assigned to each of them. This approach has the advantage of being flexible, because each user's set of authorizations can be completely different from those of the other users.

Unfortunately, this approach is also difficult to manage because as you add users, you must determine exactly which authorizations each user requires. Also, when performing audits, you must examine each user individually to determine his or her associated authorizations.

HP-UX RBAC addresses these issues by grouping users with common authorization needs into roles. Roles serve as a grouping mechanism to simplify authorization assignment and auditing. Rather than assigning an authorization directly to a user, you assign authorizations to roles. As you add users to the system, you assign them a set of roles, which determine the actions they can perform and the resources they can access.

Compare Table 8-2, which lists authorizations assigned to roles, with Table 8-1, which lists authorizations assigned to each user. By comparing these two tables, you can see how roles simplify authorization assignment.

Table 8-2 Example of Authorizations Per Role

Operation Component of Authorization	Role			
	UserAdmin	*NetworkAdmin*	*BackupOper*	*Admin*
hpux.user.add	•			•

Table 8-2 Example of Authorizations Per Role *(continued)*

Operation Component of Authorization	Role			
`hpux.user.delete`	•			•
`hpux.user.modify`	•			•
`hpux.user.password.modify`				•
`hpux.network.nfs.start`		•		•
`hpux.network.nfs.stop`		•		•
`hpux.network.nfs.config`		•		•
`hpux.fs.backup`			•	•
`hpux.fs.restore`			•	•

NOTE: Table 8-2 shows only the operation element of the authorizations—not the object element of the authorization.

8.3 HP-UX RBAC Components

Following is a list of the primary HP-UX RBAC components:

`privrun` wrapper command	Based on authorizations associated with a user, `privrun` invokes existing legacy applications with privileges after performing authorization checks and optionally re-authenticating the user and without modifying the application.
`privedit` command	Based on the authorizations associated with a user, `privedit` allows users to edit files they usually would not be able to edit because of file permissions or Access Control Lists (ACLs).
Privilege shells	Privilege shells (`privsh`, `privksh`, and `privcsh`) that automatically invoke the access control subsystem to run commands with privileges when appropriate.
management commands	Edits and validates HP-UX RBAC database files.
Access Control Policy Switch (ACPS)	Determines whether a subject is authorized to perform an operation on an object.
Access Control Policy Module (ACPM)	Evaluates HP-UX RBAC databases files and applies mapping policies to service access control requests.

SMH integration	RBAC System Management Homepage (SMH) integration to allow the graphical management of the RBAC databases through a Web interface.

The following sections discuss the HP-UX RBAC components in more detail.

8.3.1 HP-UX RBAC Access Control Policy Switch

The HP-UX RBAC Access Control Policy Switch is a customizeable interface between applications that must make access control decisions and the access control policy modules that provide decision responses after interpreting policy information in RBAC databases. As shown in Figure 8-1, from its location in the HP-UX RBAC architecture, the ACPS provides an interface between the access control policy modules and the applications that make access control decisions.

The ACPS has the following interfaces, described in detail in their respective manpages:

- ACPS application programming interface (API)
- ACPS service provider interface (SPI)
- `/etc/acps.conf`

The administrative interface for the ACPS is the `/etc/acps.conf` configuration file. The `/etc/acps.conf` configuration file determines which policy modules the ACPS consults, the sequence in which the modules are consulted, and the rules for combining the module's responses to deliver a result to the applications that need access control decisions. This ACPS implementation allows you to create a module to enforce custom policy without modifying existing role-based access control applications.

NOTE: Refer to *acps*(4), *acps.conf*(4), *acps_api*(3), and *acps_spi*(3) for more information on the ACPS and its interfaces.

8.3.2 HP-UX RBAC Configuration Files

Table 8-3 lists and briefly describes the HP-UX RBAC files.

Table 8-3 HP-UX RBAC Configuration Files

Configuration File	Description
`/etc/rbac/auths`	Database file containing all valid authorizations.
`/etc/rbac/cmd_priv`	`privrun` database file containing command and file authorizations and privileges.
`/etc/rbac/role_auth`	Database file defining the authorizations for each role.
`/etc/rbac/roles`	Database file defining all configured roles.
`/etc/rbac/user_role`	Database file defining the roles for each user.

Table 8-3 HP-UX RBAC Configuration Files *(continued)*

Configuration File	Description
`/etc/acps.conf`	Configuration file for the ACPS.
`/etc/rbac/aud_filter`	Audit filter file identifying specific HP-UX RBAC roles, operations, and objects to audit.

8.3.3 HP-UX RBAC Commands

Table 8-4 lists and briefly describes the HP-UX RBAC commands.

Table 8-4 HP-UX RBAC Commands

Command	Description
`privrun`	Invokes legacy application with privileges after performing authorization checks and optionally re-authenticating the user.
`privedit`	Allows authorized users to edit files that are under access control.
`roleadm`	Edits of role information in the `/etc/rbac/user_role`, `/etc/rbac/role_auth`, and `/etc/rbac/roles` files.
`authadm`	Edits authorization information in the `/etc/rbac/role_auth` and `/etc/rbac/roles` files.
`cmdprivadm`	Edits command authorizations and privileges in the `/etc/rbac/cmd_priv` database.
`rbacdbchk`	Verifies authorizations and syntax in the HP-UX RBAC and `privrun` database files.

8.3.4 HP-UX RBAC Manpages

Table 8-5 lists and briefly describes the HP-UX RBAC manpages.

Table 8-5 HP-UX RBAC Manpages

Manpage	Description
rbac(5)	Describes the HP-UX RBAC feature.
acps(3)	Describes the ACPS and its interfaces.
acps.conf(4)	Describes the ACPS configuration file and its syntax.
acps_api(3)	Describes the ACPS Application Programming Interface.
aacps_spi(3)	Describes the ACPS Service Provider Interface.
privrun(1m)	Describes `privrun` functionality and syntax.
privedit(1m)	Describes `privedit` functionality and syntax.
roleadm(1m)	Describes `roleadm` functionality and syntax.

Table 8-5 HP-UX RBAC Manpages *(continued)*

Manpage	Description
authadm(1m)	Describes `authadm` functionality and syntax.
cmdprivadm(1m)	Describes `cmdprivadm` functionality and syntax.
rbacdbchk(1m)	Describes `rbacdbchk` functionality and syntax.
privsh(5m)	Overview of various privileged system shells.
rbac.conf(4m)	Configuration file for Role Based Access Control.
key_filter(4m)	Configuration file for the keystroke logging module.

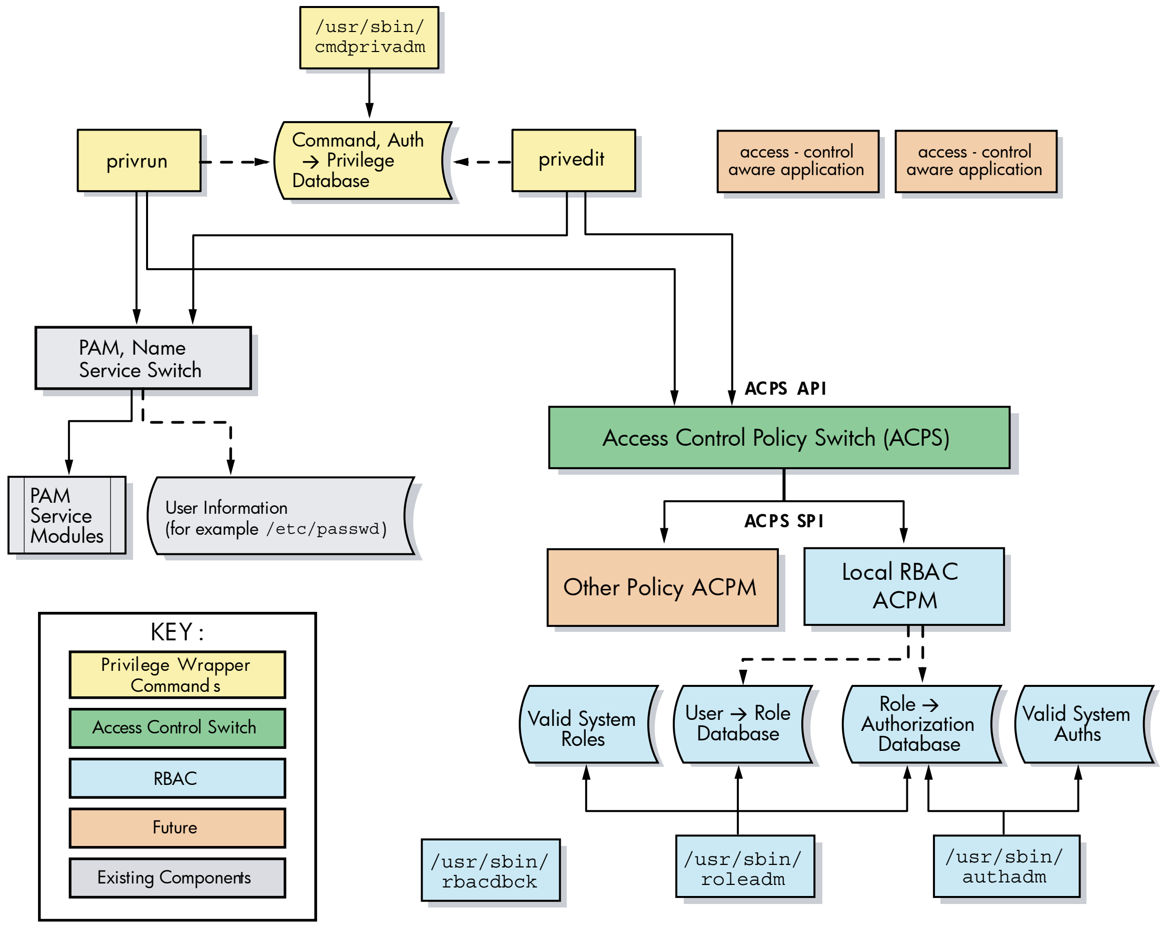
8.3.5 HP-UX RBAC Architecture

The primary component of HP-UX RBAC is the `privrun` command, which invokes existing commands, applications, and scripts. The `privrun` command uses the ACPS subsystem to make access control requests. An access request is granted or denied based on a set of configuration files that define user-to-role and role-to-authorization mappings.

If the access request is granted, `privrun` invokes the target command with additional privileges, which can include one or more of either a UID, GID, fine-grained privileges, and compartments. The privileges are configured to enable the target command to run successfully.

Figure 8-1 shows the HP-UX RBAC architecture.

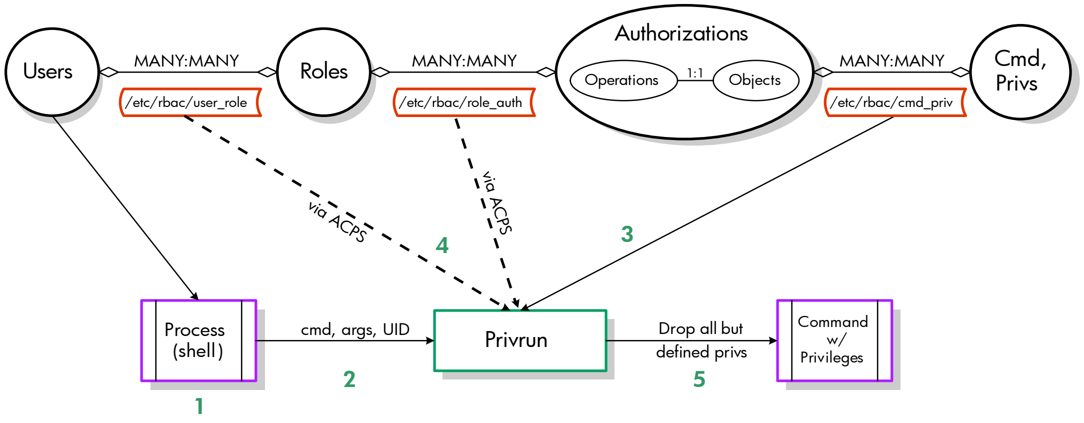
Figure 8-1 HP-UX RBAC Architecture



8.3.6 HP-UX RBAC Example Usage and Operation

Figure 8-2 and the subsequent footnotes show a sample invocation of `privrun` and the configuration files that `privrun` uses to determine whether a user is allowed to invoke a command.

Figure 8-2 Example Operation After Invoking privrun



1. A process, specifically a shell, associated with the user executes `privrun` with the goal of executing a target command with elevated privilege.
2. The target command line (command and arguments) is explicitly passed to `privrun`, and the UID of the invoking user is implicitly passed by the process context.
3. `privrun` attempts to find a match (or set of matches) within the `/etc/rbac/cmd_priv` database for the specified command line. Each matching entry also specifies a required authorization (operation, object pair) and the resulting privileges if the user has the specified authorization.
4. `privrun` makes a call (for each matching `/etc/rbac/cmd_priv` entry) to the ACPS. The HP-UX RBAC back end of the ACPS consults the `/etc/rbac/user_role` and `/etc/rbac/role_auth` databases to determine whether the user has the specified authorization, and passes this result back to `privrun`.
5. Assuming that the user associated with the process has the required authorization specified in the `/etc/rbac/cmd_priv` database for the requested command, `privrun` will drop all privileges except those specified in the `/etc/rbac/cmd_priv` entry and execute the requested command. The `privrun` command is set to `UID=0` and starts with all necessary privileges.

8.4 Planning the HP-UX RBAC Deployment

Follow these planning steps before deploying HP-UX RBAC:

1. Plan roles for users.
2. Plan authorizations for the roles.
3. Plan the authorization-to-command mappings.

The following sections describe these steps in more detail.

8.4.1 Planning the Roles

Planning an appropriate set of roles for the users of a system is a critical first step in deploying HP-UX RBAC. In some enterprises, this set of roles already exists, and you can reuse it when configuring HP-UX RBAC. More commonly, you must design the roles based on the existing tasks associated with administrative users on the system.

Consider the following guidelines when designing roles:

- There should be considerably fewer roles than the total number of users of the system. If each user requires a special role, then all of the simplified management associated with the use of roles is no longer in place.
- Roles should have some relation to the actual business roles of the users.
- Users can have multiple roles, and therefore you can design some roles simply to group authorizations common to multiple business roles. Using this approach, you can design roles hierarchically to include different roles by including their authorizations.

8.4.2 Planning Authorizations for the Roles

After defining roles, you can plan the authorizations associated with each role. If the roles align with the pre-existing operation hierarchy, then assigning the authorizations is straightforward. Enter the following command to list all the system-defined authorizations:

```
# authadm list sys
```

If the existing authorization hierarchy does not align with your roles, defining the authorizations associated with each role is more complex. You can use the following steps to help:

1. List the system commands commonly used by each role.
2. Compare these commands to the commands in the `/etc/rbac/cmd_priv` database.
3. If you find matching entries after performing the previous steps, use those entries as a guide for assigning authorizations.

For example, assume one of the desired roles is UserOperator, which commonly runs such commands as `useradd`, `usermod`, `userdel`, and so on. To determine what authorizations might be appropriate for this role, enter the following command:

```
 # grep useradd /etc/rbac/cmd_priv
/usr/sbin/useradd:dflt:(hpux.user.add,*):0/0//:dflt:dflt:dflt:
```

In this example, the `/usr/sbin/useradd` command requires the `hpux.user.add` authorization. You could assign this authorization directly, or assign `hpux.user.*` as the authorization.

Be careful using wildcards when assigning authorizations. Assigning this authorization actually assigns multiple authorizations:

```
# grep hpux.user. /etc/rbac/cmd_priv
/usr/sbin/pwgrd:dflt:(hpux.user.cache.admin,*):0/0// :dflt :dflt :dflt :
/usr/sbin/userdel:dflt:(hpux.user.delete,*):0/0// :dflt :dflt :dflt :
/usr/sbin/groupdel:dflt:(hpux.user.group.delete,*):0/0// :dflt :dflt :dflt :
/usr/sbin/useradd:dfl:(hpux.user.add,*):0/0//:dflt:dflt:dflt:
/usr/sbin/usermod:dflt:(hpux.user.modify,*):0/0// :dflt :dflt :dflt :
/usr/sbin/groupadd:dflt:(hpux.user.group.add,*):0/0// :dflt :dflt :dflt :
/usr/sbin/groupmod:dflt:(hpux.user.group.modify,*):0/0// :dflt :dflt :dflt :
/usr/sbin/vipw:dflt:(hpux.user.modify,*):0/0// :dflt :dflt :dflt :
```

8.4.3 Planning Command Mappings

Define any commands that are commonly used by any of the defined roles but do not exist in the predefined `/etc/rbac/cmd_priv` file that is provided. The `/etc/rbac/cmd_priv` file defines the mapping between authorizations and commands. Determine the following for each command:

- The full path of the command
- The necessary authorization to check before running the command
- Any special privileges needed by the command, for example, `euid=0`

The strings of text that constitute the operation and object entries in the `/etc/rbac/cmd_priv` file are arbitrary, but they should correspond logically to a command or set of commands. Consider the following guidelines when planning the authorization to command mappings in `/etc/rbac/cmd_priv`:

- Define operations into logical groups to easily assign the operations to roles.
- Do not create operation branches with too many (more than 10) or too few (1) child elements. The overall tree should not be overly wide, making it difficult to assign groups of operations, or overly tall, with individual operation names that are long and hard to use.
- End the last element of an operation name with an action (verb).
- Define operations so that new commands can be clearly placed when added.

See "Configuring Additional Command Authorizations and Privileges" for the procedure to configure additional commands.

8.4.4 HP-UX RBAC Limitations and Restrictions

Following is a list of items to consider before deploying HP-UX RBAC:

- HP-UX RBAC does not support single user mode, therefore the root account should be available during situations when single user mode is needed.
- Serviceguard does not support the use of HP-UX RBAC and `privrun` to grant access to Serviceguard commands. See Section 8.6.1.1 for more information about HP-UX RBAC and Serviceguard clusters.
- As with all applications, HP-UX RBAC is subject to the rules that govern compartments (see Chapter 6). Remember the following when using HP-UX RBAC with Compartments:

— You cannot run `privedit` on a file that is restricted by a compartment definition.

— To provide a different application with fine-grained privileges, the `privrun` command must be running with those same privileges it wants to provide to the application. By default, `privrun` is configured to run with all privileges (see getfilexsec(1M) for more information). However, sometimes this default privilege set may be restricted. For example, if a compartment is configured to disallow privileges, this specification prevents `privrun` from providing the privileges to the application in that compartment because `privrun` does not have the privileges itself. Note that by default, sealed compartments are configured to disallow the `POLICY` compound privilege.

— For `privrun` to invoke another application in a compartment, `privrun` must assert the `CHANGECMPT` privilege. If `privrun` cannot assert the `CHANGECMPT` privilege, for example, if the compartment is configured to disallow privileges, `privrun` will fail. This behavior is intentional and designed to reinforce the concept of a sealed compartment.

8.5 Configuring HP-UX RBAC

Configuring HP-UX RBAC is a three-step process:

1. Configure the roles.
2. Configure the authorizations.
3. Configure any additional commands.

IMPORTANT: Authorizations are built-in (hard-coded) to the HP-UX RBAC administration commands and cannot be configured. However, you can configure which roles and users have the required HP-UX RBAC administration command authorizations.

HP-UX RBAC administration commands do not need to be wrapped with the `privrun` command because they are `setuid=0`. The HP-UX RBAC administration commands run with privileges equal to root regardless of who invokes them. Access control checks limit who can use the HP-UX RBAC administrative commands.

See the Authorization section in each of the HP-UX RBAC administrative commands manpages for more information about their authorizations.

This Section 8.5 uses the example planning results and users in Table 8-6 to demonstrate the HP-UX RBAC administrative commands and configuration process.

Table 8-6 Example Planning Results

Users	Roles	Authorizations (Note: Objects Assumed to Be *)	Typical Commands
chandrika, rwang	UserOperator	`hpux.user.*` `hpux.security.*`	`/usr/sbin/useradd` `/usr/sbin/usermod`
bdurant, prajessh	NetworkOperator	`hpux.network.*`	`/sbin/init.d/inetd`
luman	Administrator	`hpux.*` `company.customauth`	`/opt/customcmd`

8.5.1 Configuring Roles

Configuring roles for users is a two-step process:

1. Create roles.
2. Assign roles to users or groups.

8.5.1.1 Creating Roles

Use the `roleadm` command to create roles and assign them to users or groups. You must first add roles that do not already exist, and then assign users to those roles. The following shows the `roleadm` command syntax:

```
roleadm add role [comments]
        | delete role
        | modify oldrolename newrolename
        | assign user role
        | assign "&group" role
        | revoke user [role]
        | revoke "&group" [role]
        | list [user=username] [role=rolename] [sys]
```

Following is a list and brief description of the `roleadm` command arguments:

`add`	Adds the role to the system list of roles in `/etc/rbac/roles`.
`delete`	Deletes the role from the system list of roles in `/etc/rbac/roles`.
`modify`	Changes role names in all three role-related database files: `/etc/rbac/roles`, `/etc/rbac/user_role`, and `/etc/rbac/role_auth`.
`assign`	Assigns a role to a user or group, and updates the `/etc/rbac/user_role`.
`revoke`	Revokes a role from a user or group, and removes the entry from `/etc/rbac/user_role`.
`list`	Lists the valid system roles (sys), or the user-to-role mappings.

NOTE: See the *roleadm*(1m) manpage for more information.

Following are two examples of the `roleadm` command adding new roles:

```
# roleadm add UserOperator
roleadm: added role UserOperator

# roleadm add NetworkOperator
roleadm: added role NetworkOperator
```

NOTE: The default configuration files delivered with HP-UX RBAC contain a single preconfigured role: Administrator. By default, the Administrator role is assigned all HP-UX system authorizations (`hpux.*, *`) and is associated with the root user.

After defining valid roles, you can assign them to one or more users or groups. Attempting to assign a role that has not been created to users will display an error message indicating that the role does not exist.

8.5.1.2 Assigning Roles to Users

Separating role creation from role assignment offers the following advantages:

- Requiring that roles be created before they are assigned ensures that any typographical errors are caught when specifying role names during role assignment.
- Allows different users to perform each task. For example, the same user is not required to both create the roles and assign the roles.

After creating valid roles, use the `roleadm` command to assign them to the appropriate users, as shown in the following examples:

```
# roleadm assign luman Administrator
roleadm assign done in /etc/rbac/user_role

# roleadm assign rwang UserOperator
roleadm assign done in /etc/rbac/user_role
```

After using the `roleadm assign` command to assign roles to users, you can use the `roleadm list` command to verify that the roles were assigned correctly, for example:

```
# roleadm list
root: Administrator
luman: Administrator
rwang: UserOperator
```

NOTE: HP-UX RBAC offers the ability to add a special user named DEFAULT to the /etc/rbac/user_role database. Assigning a role to the DEFAULT user means any user that does not exist on the system is assigned that role.

8.5.1.3 Assigning Roles to Groups

HP-UX RBAC also enables you to assign roles to groups. You can use the roleadm command options that use the *user* value, such as roleadm assign *user role* and roleadm revoke *user role* to administer groups and roles.

Assign, revoke, or list group and role information using the roleadm command by inserting an ampersand (&) at the beginning of the user value and enclosing the user value in quotations. The group name value and ampersand (&) must be shell escaped or enclosed in quotations to be interpreted by roleadm. For example:

```
# roleadm assign "&groupname" role
```

8.5.2 Configuring Authorizations

Configuring authorizations is similar to creating and assigning roles. However, authorizations contain two elements: an operation and an object. The * wildcard—the most commonly used object—is the implicit object used if you do not specify an object while invoking the authadm command. In many cases, the object is purposely left unspecified, so that the operation applies to all objects. Leaving the object unspecified is often used for authorizations that apply to wrapped commands because it can be difficult to determine the target of an action from the command name.

An example of this object ambiguity is the /usr/sbin/passwd command. The passwd command can operate on a number of repositories, for example, the /etc/passwd file, an NIS table, and an LDAP entry. You cannot determine the actual object by looking at the command line, so it is typically easiest to require that the user have the operation on all objects, for example: (hpux.security.passwd.change, *).

NOTE: You can configure a value for the default object. By default, if you do not specify an object, HP-UX RBAC will use the * wildcard as the object. However, if you have configured a value for the RBAC_DEFAULT_OBJECT= parameter in /etc/default/security, HP-UX RBAC will use this value instead of the * wildcard as the default object.

Use the authadm command to edit authorization information in the HP-UX RBAC databases. The authadm syntax is similar to the roleadm syntax. Following is the authadm command syntax:

```
authadm add operation[object[comments]]
        | delete operation[object]
        | assign role operation[object]
        | revoke [role=name][operation=name[object=name]]
        | list [role=name][operation=name[object=name][sys]
```

The following is a list and brief description of the `authadm` command arguments:

`add`	Adds an authorization to the system list of valid authorizations in `/etc/rbac/auths`.
`delete`	Deletes an authorization from the system list of valid authorizations in `/etc/rbac/auths`.
`assign`	Assigns an authorization to a role and adds an entry to `/etc/rbac/role_auth`.
`revoke`	Revokes an authorization from a role and updates `/etc/rbac/role_auth`.
`list`	Lists valid authorizations per system or role, and lists roles associated with the specified operation.

IMPORTANT: Be aware that when you assign an authorization that contains the asterisk * character, you must surround the wildcard character with quotation marks to prevent shell interpretation, as shown in the following examples.

The following are examples of authorization creation and assignment based on Table 8-6:

```
# authadm add 'company.customauth.*'
authadm added auth: (company.customauth.*,*)

# authadm assign Administrator 'company.customauth.*'
authadm added auth for role Administrator
```

Use the `list` argument with the `authadm` command to verify the authorization assignment, for example:

```
# authadm list
Administrator: (hpux.*, *) (company.customauth.*, *)
```

8.5.3 Configuring Additional Command Authorizations and Privileges

You must define any additional commands that are not provided in the default configuration. The authorizations needed to run the commands must already exist and must be assigned to a role. If you have not done this, the command will be configured, but no user will be appropriately authorized to use the command.

Use the `cmdprivadm` command to edit a command's authorization and privilege information. The `cmdprivadm` command works in a similar fashion to `roleadm` and `authadm`, but only allows addition and removal of a command privilege and authorization in the privrun database.

The following shows the `cmdprivadm` command syntax:

```
cmdprivadm add cmd=full_path_name_of_a_command | full_path_name_of_a_file
                        |[op=operation]|[object=object]
                        |[ruid=ruid]|[euid=euid]
                        |[rgid=rgid]|[egid=egid]
                        |[compartment=compartment_label]
                        |[privs=comma_separated_privilege_list]
```

```
                            |[re-auth=pam_service_name]
                            |[flags=comma_separated_flags_list]
cmdprivadm delete cmd=full_path_name_of_a_command | full_path_name_of_a_file
                            |[op=operation]|[object=object]
                            |[ruid=ruid]|[euid=euid]
                            |[rgid=rgid]|[egid=egid]
                            |[compartment=compartment_label]
                            |[privs=comma_separated_privilege_list]
                            |[re-auth=pam_service_name]
                            |[flags=comma_separated_flags_list]
```

The following is a list and brief description of the two main `cmdprivadm` command arguments:

`add`	Adds command (or file) authorization information to the `/etc/rbac/cmd_priv` database.
`delete`	Deletes command (or file) authorization information in the `/etc/rbac/cmd_priv` database.

The following example demonstrates the most common `cmdprivadm` arguments:

```
# cmdprivadm add cmd=/opt/customcmd \
op=companyname.customcommand ruid=0 euid=0 flags=edit \
/opt/customcmd::(companyname.customcommand,*):0/0/-1/-1::::edit
cmdprivadm added the entry to /etc/rbac/cmd_priv
```

As shown in the previous example, the `cmd_priv` file database file contains a field for flag values. Be sure to consider the value of the `cmdprivadm flags` when configuring command or file authorization and privilege information.

The `privrun` command recognizes one defined flag, `KEEPENV`. If the `KEEPENV` flag is set in the `cmd_priv` file for a particular command, none of the environment variables will be scrubbed when `privrun` wraps that particular command.

For `privedit`, you can specify flag values to indicate whether or not `privedit` can edit a file. Additional flag values can be specified to indicate whether `privrun` can execute a command. The following are the supported flag values:

`flag=`empty or any other token	Indicates the file can only be executed and cannot be edited.
`flag=edit`	Indicates the file can be both edited and executed. This flag is mainly intended for scripts.
`flag=noexec`	Indicates the file cannot be executed and can only be edited with `privedit`.

NOTE: See cmdprivadm(1M) for information on all of the `cmdprivadm` arguments. Most arguments are optional and are filled in with reasonable defaults if nothing is specified.

NOTE: To modify an existing entry in the `/etc/rbac/cmd_priv` file, you must first delete the entry and then add the updated version back in. When you use `cmdprivadm` to delete entries, arguments act as filters. For example, specifying the `cmdprivadm delete op=foo` command removes all entries where the operation is `foo`. As a result of this, when you use `cmdprivadm` to delete entries, be careful to ensure that you specify sufficient arguments to uniquely identify the entries to be removed.

8.5.4 Configuring HP-UX RBAC with Fine-Grained Privileges

Applications communicate with the system's resources using system calls, allowing the operating system access to system resources. Certain system calls require special, elevated privileges for the application to access the operating system and system hardware.

Before fine-grained privileges were available, `UID=0` would satisfy as a special, elevated privilege for certain system calls. If the UID was not 0, the system call was denied and an application error returned.

HP-UX RBAC and specifically the `privrun` wrapper command allows non-root users to acquire the level of special privileges or `UID=0` required for running certain applications. In addition to providing `UID=0` to a non-root user in certain circumstances to run a particular application, HP-UX RBAC can also use the fine-grained privileges to run applications with additional privileges, but without `UID=0`.

You can use HP-UX RBAC to configure commands to run with only a select set of privileges and with different sets of privileges for different users, all without UID=0. For example, an administrator might need to run the `foobar` command with several privileges, and a normal user might need far fewer privileges to run `foobar`.

Think of fine-grained privileges as "system call access control check keys." Rather than checking for `UID=0`, the system call checks for a particular privilege. These fine-grained privileges provide the ability to "lock" system calls and to control application access to the operating system and hardware resources. Also, by splitting privileges into finely-grained privileges, applications do not require all privileges to run—only a specific privilege or set or privileges. Should an application process running with a particular set of privileges be compromised, the potential damage is far less than it would be if the process was running with `UID=0`.

NOTE: See privileges(5) for more information fine-grained privileges.

Use the `cmdprivadm` command and the `privs` option to configure commands for `privrun` to wrap and run only with the specified privileges. The following is an example `cmdprivadm` command that configures the `/usr/bin/ksh` command to run with the

`BASICROOT` compound privilege and that requires the `(hpux.adm.mount, *)` authorization:

```
# cmdprivadm add cmd=/etc/mount op=hpux.adm.mount object='*' privs=BASICROOT
```

The preceding `cmdprivadm` command creates an entry in the `/etc/rbac/cmd_priv` file as follows:

```
#------------------------------------------------------------------------------------------------------
# Command        : Args   :Authorizations       :U/GID :Cmpt   :Privs     :Auth  :Flags
#---------------:--------:--------------------:------:-------:---------:------:-------------------
/etc/mount       :dflt    :(hpux.adm.mount,*)   :///   :dflt   :BASICROOT :dflt  :
```

After you create the entry using `cmdprivadm` and using `privrun` to wrap the command,`/etc/mount` will run with the elevated privilege of the `BASICROOT` compound fine-grained privilege and without `UID=0` if the user has the `(hpux.adm.mount, *)` authorization.

As described in Section 8.6.1, the `privrun -p` command option matches only the entries in the `/etc/rbac/cmd_priv` database file that have the privileges specified by the `-p` option. Be aware when you specify a privilege using the `privrun -p` option that `privrun` will match all entries that contain the specified privilege—including groups of privileges and compound privileges that include the `-p` specified privilege. The `privrun` command will execute according to the first match in `/etc/rbac/cmd_priv`. For example, the following is an example `privrun -p` command and a list of entries the command will match in `/etc/rbac/cmd_priv`:

The command:

```
# privrun -p MOUNT /etc/mount
```

matches the following `/etc/rbac/cmd_priv` entries:

```
#------------------------------------------------------------------------------------------------------
# Command        : Args   :Authorizations     :U/GID :Cmpt  :Privs
 :Auth :Flags
#---------------:--------:------------------:------:------:-------------------------------------:-----:-----
/etc/mount       :dflt    :(hpux.adm.mount,*) :///   :dflt  :PRIV_CHOWN, MOUNT
 :dflt :
/etc/mount       :dflt    :(hpux.*,nfs)       :///   :dflt  :MOUNT, PRIV_RTPRIO, PRIV_MLOCK
 :dflt :
/etc/mount       :dflt    :(hpux.adm.*,*)     :///   :dflt  :BASICROOT
 :dflt :
```

NOTE: The `privrun -p MOUNT /etc/mount` command matches the `BASICROOT` privilege because the MOUNT simple privilege is part of the predefined `BASICROOT` compound privilege. See the *privileges*(5) manpage for more information about simple and compound privileges.

IMPORTANT: The sequence of the entries in `/etc/rbac/cmd_priv` is important because `privrun` will execute according to the first explicit match it finds. In the preceding example, while all three entries are considered matches to the `privrun` command, `privrun` would execute the first entry. Keep the sequence of the entries in mind when configuring commands and authorizations. The `cmdprivadm` tool adds entries to the bottom of the `/etc/rbac/cmd_priv` file.

8.5.5 Configuring HP-UX RBAC with Compartments

HP-UX RBAC can also use compartments to configure applications to run in a particular compartment. With compartments, you can logically partition a system into compartments so that a process cannot communicate or access resources outside of its compartment (unless a compartment rule is set up to allow this).

The following is an example `cmdprivadm` command that configures the `/sbin/init.d/hpws_apache` command to run only in the `apache` compartment, which is defined by the `/etc/cmpt/apache.rules` compartment rule:

```
# cmdprivadm add cmd='/sbin/init.d/hpws_apache -a start' \
op=hpux.network.service.start object=apache compartment=apache
```

The preceding `cmdprivadm` command creates an entry in the `/etc/rbac/cmd_priv` file, as follows:

```
#---------------------------------------------------------------------------------------------------
# Command                 : Args   :Authorizations                     :U/GID   :Cmpt   :Privs :Auth
   :Flags
#-----------------------:-------:--------------------------------:------------:-------:------:------
/sbin/init.d/hpws_apache  :start   :(hpux.network.service.start,apache) :///    :apache :dflt  :dflt
   :
```

After you create the entry using `cmdprivadm` and using `privrun` to wrap the command, authorized users can execute the `/sbin/init.d/hpws_apache -start` command, and it will run only in the `apache` compartment. The compartment tag for the process is changed to `apache`, and properties of the process will follow the defined `apache` compartment rules.

NOTE: Use only the `cmdprivadm` command to configure compartments for commands. Do not edit the `/etc/rbac/cmd_priv` database file without using `cmdprivadm`.

To modify an existing entry in the `/etc/rbac/cmd_priv` file, you must first delete the entry and then add the updated version back in. When you use `cmdprivadm` to delete entries, arguments act as filters. For example, specifying the `cmdprivadm delete op=foo` command removes all entries in which the operation is `foo`. As a result of this, when you use `cmdprivadm` to delete entries, be careful to ensure that you specify sufficient arguments to uniquely identify the entries to be removed.

8.6 Using HP-UX RBAC

This section explains how to run the `privrun` and `privedit` commands to operate HP-UX RBAC.

8.6.1 Using the privrun Command to Run Applications with Privileges

The `privrun` command enables a user to run legacy applications with different privileges, according to the authorizations associated with the invoking user. The user invokes `privrun`, specifying the legacy application as command line arguments. Next, `privrun` consults the `/etc/rbac/cmd_priv` database to determine what authorization is required to run the command with additional privileges. If the user has the necessary authorization, `privrun` invokes the specified command after changing its `UID` and or `GID` as specified in the `/etc/rbac/cmd_priv` database.

The following is the `privrun` command syntax:

```
privrun [options] command [args]
        | [-u eUID|username]
        | [-g eGID|groupname]
        | [-U rUID|username]
        | [-G rGID|groupname]
        | [-a (operation, object)]
        | [-c compartment]
        | [-p privilege[,privilege,privilege...]]
        | [-x]
        | [-v [-v]]
        | [-h]
        | [-t]
```

The following list explains each of the `privrun` command options:

`-u`	Matches only those entries containing the effective user ID (`EUID`) corresponding to the specified `EUID` or the `EUID` associated with the username.
`-g`	Matches only those entries containing the effective group ID (`EGID`) corresponding to the specified `EGID` or the `EGID` associated with the group name.
`-U`	Matches only those entries containing the real user ID (`RUID`) corresponding to the specified `RUID` or the `RUID` associated with the username.

-G	Matches only those entries containing the real group ID (RGID) corresponding to the specified RGID or the RGID associated with the group name.
-a	Matches only those entries requiring the specified authorization. Authorization is defined as (operation, object) pairs in the /etc/rbac/cmd_priv database file. The specified authorization must exactly match the authorization present in the /etc/rbac/cmd_priv file—wildcards are not supported.
-c	Matches the specified compartment in the /etc/rbac/cmd_priv database file. The specified compartment must exactly match the compartment present in /etc/rbac/cmd_priv.
-p	Matches the specified privileges with the privileges in the /etc/rbac/cmd_priv database file. You can specify more than one privilege. When specifying multiple privileges, separate each privilege with a comma. Be aware when you specify a privilege using the privrun -p option that privrun will match all entries that contain the specified privilege—including groups of privileges and compound privileges that include the -p specified privilege. The privrun command will execute according to the first match in /etc/rbac/cmd_priv.
-x	Uses a fall-through mode that modifies the behavior of privrun only when an authorization or authentication check fails. Rather than exiting with an error message, the target command runs, but without any additional privileges. The target command executes as though the user ran the command directly without privrun.
-v	Invokes privrun in verbose mode. The verbose level increases if two -v options are specified. An increased verbose level prints more information.
-h	Prints privrun help information.
-t	Uses a test mode that performs all the normal authorization and authentication checks according to the configuration files to see if the desired privrun invocation will succeed. The only difference is that instead of executing the command, upon success, privrun -t just returns. Use this to preview whether a given privrun invocation will succeed.

The following is an example of the most basic privrun usage—wrapping a legacy application. In this case, the ipfstat command runs as a privrun command argument in order to run according to the authorizations associated with the invoking user:

```
# privrun ipfstat
```

As long as the user logged in has the necessary authorization, defined in /etc/rbac/cmd_priv, the privrun wrapper command will execute the legacy command with the privileges (UID and GID) defined in the /etc/rbac/cmd_priv entry.

Multiple entries can exist for the same command, potentially with different required authorizations and different resulting privileges. In this case, privrun iterates sequentially through the /etc/rbac/cmd_priv database, executing the first command the user is authorized for.

In some cases, this may not be ideal. For example, all users may be allowed to run the `passwd` command to change their own password but if a user administrator runs it, they need the privileges to change other users' passwords. If the entry for all the normal users is listed before the entry for the user administrators, it is executed first, and this might prevent the user administrators from running the more privileged version.

For cases like this, `privrun` has options that allow users to specify the desired privileges. Only entries matching the specified privileges (for example, `UID`) are used. If no entries match the desired privileges, `privrun` returns an error message.

The following is an example invocation of `privrun` that matches only entries where the effective `UID` is set to 0:

```
# privrun -u 0 ipfstat
```

NOTE: See the *privrun*(1M) and *rbac*(5) manpages for more about using the `privrun` command.

8.6.1.1 HP-UX RBAC in Serviceguard Clusters

Serviceguard does not support the use of HP-UX RBAC and `privrun` to grant access to Serviceguard commands. Serviceguard version A.11.16 implemented its own Role-Based Access Control by specifying Access Control Policies through package and cluster configuration files, providing cluster-aware policies for Serviceguard operations. The Serviceguard mechanism must be used for Role Based Access Control of Serviceguard operations. See the latest *Managing Serviceguard* document for additional details on Serviceguard Access Control Policies.

HP-UX RBAC can be used with non-Serviceguard commands in a Serviceguard cluster. The same HP-UX RBAC rules should be applied to all nodes in the cluster.

8.6.2 Using the privedit Command to Edit Files Under Access Control

The `privedit` command allows authorized users to edit files they usually would not be able to edit because of file permissions or ACLs. After you invoke the command and identify the file you want to edit as an argument, `privedit` checks the `/etc/rbac/cmd_priv` database, just as `privrun` does, to determine the authorization required to edit the specified file. If the invoking user is authorized to edit the file, `privedit` invokes an editor on a copy of the file.

NOTE: When you use `privedit` to invoke an editor to edit a file, the editor does not run with any elevated privileges. Because the editor `privedit` invokes does not run with elevated privileges, any attempted actions, such as shell escapes, run with the user's typical (non-elevated) privilege set.

You can specify which editor `privedit` uses to edit the file by setting the `EDITOR` environment variable. If you do not set the `EDITOR` variable, `privedit` uses the default editor, `vi`. You cannot pass arguments to the editor via the `privedit` command line.

However, the editor recognizes and supports editor-specific environment variables if you set them before invoking `privedit`.

Use a fully qualified file name as a `privedit` argument to identify which file to edit. If you do not use a fully qualified file name, `privedit` adds the current working directory to the beginning of the file name you specify. Regardless of how you specify the file to edit, all file names are fully qualified after you invoke `privedit`. The `privedit` command also recognizes and supports files that are symbolic links.

The `privedit` command can edit only one file at a time. If you specify multiple file names as `privedit` arguments, `privedit` edits the first file specified and ignores the subsequent file names. The following shows the `privedit` command syntax:

```
privedit [option] fully-qualified-file-name
         |   [-a (operation, object)]
         |   [-v]
         |   [-h]
         |   [-t]
         |   [-x]
```

The following is a list and brief description of the `privedit` command options:

`-a` *authorization*	Match only the `/etc/rbac/cmd_priv` file entries with that have the specified authorization.
`-v`	Invokes `privedit` in verbose mode.
`-h`	Prints `privedit` help information.
`-t`	Checks if the user has the required authorization to edit the file and reports the results.
`-x`	If the authorization check fails, the file will be edited with the caller's original privileges.

The following is an example of using a `privedit` command to edit the `/etc/default/security` file with the specific authorization of (`hpux.sec.edit, secfile`):

```
# privedit -a "(hpux.sec.edit, secfile)" /etc/default/security
```

NOTE: Remember that the flag values for each entry in the `cmd_priv` database dictate whether or not `privedit` can edit a file. See "Configuring Additional Command Authorizations and Privileges" and the *privedit*(1M) manpage for more information about flags and using the `privedit` command.

8.6.3 Customizing privrun and privedit Using the ACPS

The HP-UX RBAC feature provides the ability to customize how `privedit` and `privrun` check user authorizations. The ACPS module is a customizeable interface that provides responses to applications that must make authorization decisions. The ACPS configuration file, `/etc/acps.conf`, controls the following aspects of the ACPS:

- which modules are consulted for making access decisions
- the sequence in which the modules are consulted
- the rules for combining module responses to return results to applications

See Section 8.3.1, and *acps.conf*(4), *acps*(3), and *rbac*(5) for more information about the ACPS.

8.6.4 Generating Keystroke and Command Logs

An authorized user can generate "keystroke logs" for selected users, as well as generate a log of commands invoked through RBAC without the need for the HP-UX audit system. This section describes these features:

- Keystroke logging
- Alternate logging

8.6.4.1 Keystroke Logging

In many situations, it is sufficient to simply log the set of privilege commands invoked by a user. RBAC has supported this functionality since its initial release with the HP-UX audit system. There are some situations, however, where this coarse level of logging is insufficient. For example, there are some legislative compliance regulations that require that all actions performed by an administrator are logged, not just the privileged actions. There are situations where it is desirable to only log in the event that certain files or objects are accessed. And there are situations where selected users are granted "unconstrained root privileges", such as a `root` shell under the caveat that all of their actions are logged. These uses are granted maximum administrative flexibility.

Keystroke logging enhances the logging capability. RBAC provides a PAM module that you can configure to log a user's entire terminal session, or relevant parts of a session based on keyword "triggers". You can customize this keystroke logging policy to capture session logs for particular users, roles, and groups. In order to enable this functionality, an administrator must perform the following steps after installing the RBAC product depot:

1. Create an entry (or entries) in the PAM configuration file (`/etc/pam.conf`) including the keystroke library as a session module:

   ```
   login    session optional       libpam_keystroke.so.1
   dtlogin  session optional       libpam_keystroke.so.1
   sshd     session optional       libpam_keystroke.so.1
   rcomds   session optional       libpam_keystroke.so.1
   OTHER    session optional       libpam_keystroke.so.1
   ```

   Note that this module may be configured for one or more services, depending on the intended effect of the logging. For more information on `pam.conf` and the syntax of the entries, see *pam.conf*(4).
2. Enable keystroke logging in `/etc/rbac/rbac.conf`:

```
KEY_STROKE_LOGGING = 1
```

3. Create a keyfilter file under `/etc/rbac` specifying what users to log. For more information on customizing specific policies, see *key_filter*(4m).

Once these steps are completed, subsequent access by the targeted users will cause a keystroke log file to be generated and stored in the location specified in `/etc/rbac/rbac.conf` file. Note that in the event that a user has privileged access to this location (for example, they are granted a `root` shell), they may be able to modify these files. In this situation, HP recommends that modification of the files be monitored (for example, by HP-UX Host IDS) or that they periodically be transferred off-host.

NOTE: The keystroke logging feature does not currently work with Secure Shell (SSH) login.

8.6.4.2 Alternate Logging

The alternate logging feature enables you to log access control events and RBAC-invoked commands. It is no longer necessary to enable HP-UX auditing to generate RBAC logs. An administrator can enable RBAC logging and specify the location of the alternate logging files simply by editing the `/etc/rbac/rbac.conf` file. For more information on the specific keyword/value pairs, see *rbac_conf*(4m).

Alternate logging works in an identical fashion to the audit logging and may be configured using the `/etc/rbac/aud_filter` file. The traditional RBAC audit log generation continues to work. If both auditing and logging are enabled, two sets of logs will be generated.

8.7 Troubleshooting HP-UX RBAC

The following is a list of the primary mechanisms used to troubleshoot and debug HP-UX RBAC:

- The `rbacdbchk` utility verifies HP-UX RBAC database syntax.
- The `privrun -v` command reports additional and relevant information.

8.7.1 The rbacdbchk Database Syntax Tool

The most common bugs are caused by manual editing of the HP-UX RBAC databases, resulting in syntactically invalid configurations or in configurations that are inconsistent between databases (for example, a role in `/etc/rbac/user_role` that is not defined in `/etc/rbac/roles`). To assist in diagnosing these common mistakes, HP-UX RBAC includes an `rbacdbchk` command. This command reads through the HP-UX RBAC databases and prints warnings where incorrect or inconsistent configuration entries are found:

```
# rbacdbchk
[/etc/rbac/user_role] chandrika: UserOperator
        invalid user
         The value 'chandrika' for the Username field is bad.
```

```
[/etc/rbac/cmd_priv]
/opt/cmd:dflt:(newop,*):0/0//:dflt:dflt:dflt:
        invalid command: Not found in the system
        The value '/opt/cmd' for the Command field is bad.

[Role in role_auth DB with no assigned user in user_role DB]
Rebooter:(hpux.admin.*, *)

[Invalid Role in user_role DB. Role 'UserOperator' assigned to user 'chandrika' does not exist in
the roles DB]
```

On a correctly configured system, the `rbacdbchk` command produces no output, indicating no errors are present.

8.7.2 privrun -v Information

The second method for detecting problems is to run the `privrun` command with the `-v` option (verbose mode). In verbose mode, `privrun` provides additional information about the entries that the input command matched and the status of the authorization checking, as well as other relevant data. In many cases, this output clarifies the issue causing `privrun` to fail. Specify the `-v` option multiple times for additional levels of verbose output. The following is an example of the `privrun -v` output with the `ipfstat` command:

```
# privrun -v /sbin/ipfstat
privrun: user root intends to execute command /sbin/ipfstat
privrun: input entry: '/sbin/ipfstat:dflt:(,):///:dflt:dflt::'
privrun: found matching entry: '/sbin/ipfstat:dflt:(hpux.network.filter.readstat,*):0/0//:dflt:dflt::'
privrun: passed authorization check
privrun: attempting to set ruid/euid/rgid/egid to 0/0/-1/-1
privrun: current settings for ruid/euid/rgid/egid are 0/0/3/3
privrun: executing: /sbin/ipfstat
```

9 Audit Administration

The purpose of auditing is to selectively record events for analysis and detection of security breaches. The audit data is recorded in log files. Thus, the auditing system acts as a deterrent against system abuses and exposes potential security weaknesses.

The auditing system records instances of access by subjects to objects on the system; it detects any (repeated) attempts to bypass the protection mechanism and any misuses of privileges; it also helps in exposing potential security weaknesses in the system.

When a user logs in, a unique audit session ID called "audit tag" is generated and associated with the user's process. The audit tag remains the same during each login session. Even if a user changes identity within a single session, all events are still recorded with the same audit tag and accountable under the original login user's name.

Audit records are generated for selective security related system events. Each audit record contains information about the event, such as what the event was, when it occurred, the ID of the user who caused it, the ID of the process that caused it and so on.

Audit records are collected in audit logs/files in binary format. The HP-UX Auditing system on the HP-UX 11i v3 release is capable of using more than one writer thread to log data. Each writer thread writes to one file, allowing an audit trail to be written in parallel by multiple kernel threads and hence potentially increasing the throughput of the system. As a result, an audit trail is present on the file system as a directory with multiple audit files in it.

The records in the audit trail are compressed to save disk space. When a process is audited the first time, a process identification record (PIR) is written into the audit trail containing information that remains constant throughout the lifetime of the process. The PIR includes the process ID, the parent process' ID, audit tag, real user ID, real group ID, effective user ID, effective group ID, group ID list, effective, permitted, and retained privileges, compartment ID, and the terminal ID. The PIR is entered only once per process per audit trail.

This chapter discusses the following topics:

- Auditing components (Section 9.1)
- Auditing your system (Section 9.2)
- Auditing users (Section 9.3)
- Auditing events (Section 9.4)
- Audit trails (Section 9.5)
- Audit filtering tools (Section 9.6)
- Using filter.conf (Section 9.7)
- Audit reporting tools (Section 9.8)
- Viewing audit logs (Section 9.9)

- Self-auditing (Section 9.10)
- HP-UX RBAC auditing (Section 9.11)

9.1 Auditing Components

The auditing feature of HP-UX 11i contains configuration files, commands, and manpages. These are listed in the following sections.

9.1.1 Commands

Table 9-1 contains a brief description of each auditing command.

Table 9-1 Audit Commands

Command	Description
`audevent`	Changes or displays event or system call status.
`audfilter`	Loads, clears, and displays the audit filtering policy.
`auditdp`	Selectively reads and writes audit data, converting the data format in the process.
`audisp`	Displays the audit records.
`audomon`	Sets the audit file monitoring and size parameters.
`audsys`	Starts and stops auditing; sets and displays audit file or directory information.
`userdbset`	Selects users to be audited.

9.1.2 Audit Configuration Files

Table 9-2 contains a brief description of each configuration file associated with the audit feature.

Table 9-2 Audit Configuration Files

File	Description
`/etc/audit/audit.conf`	File containing pre-defined event classification information.
`/etc/audit/audit_site.conf`	File containing site-specific event classification information.
`/etc/default/security`	File containing system-wide auditing defaults.
`/var/adm/userdb`	Database containing per-user audit information.
`/etc/rc.config.d/auditing`	File containing configuration information directing audit to start at system reboot.
`/etc/audit/filter.conf`	File containing rule-based audit filtering policy.

9.1.3 Audit Manpages

Table 9-3 contains a brief description of each manpage associated with the auditing feature.

Table 9-3 Audit Manpages

Manpage	Description
audevent(1M)	Describes `audevent` functionality and syntax.
audisp(1M)	Describes `audisp` functionality and syntax.
audomon(1M)	Describes `audomon` functionality and syntax.
audsys(1M)	Describes `audsys` functionality and syntax.
userdbset(1M)	Describes `userdbset` functionality and syntax.
audit.conf(4)	Describes the `/etc/audit/audit.conf` file.
audit(5)	Gives introductory information about HP-UX auditing.
audfilterd(1M)	Describes the service daemon, `audfilterd`, that manages the audit pre-filtering policy.
audfilter(1M)	Describes the tool that configures the audit pre-filtering policy.
filter.conf(4)	Describes the file containing the rule-based audit pre-filtering policy.
auditdp(1M)	Describes the audit data processing tool.
audit_dpms_api(3)	Describes the Audit DPMS application programming interface.
audit_dpms_spi(3)	Describes the Audit DPMS Service Provider Interface.
audit_dpms_filter(4)	Describes the configuration file for filtering Audit DPMS data.
audit_dpms(5)	Describes the Audit Data Process Module Switch.
audit_hpux_portable(5)	Describes the Audit DPMS service module for managing portable format audit data.
audit_hpux_raw(5)	Describes the Audit DPMS service module for managing HP-UX raw audit data.
audit_hpux_xml(5)	Describes the Audit DPMS service module for generating XML format audit data.

9.2 Auditing Your System

Use the following procedures to plan, enable, and monitor auditing on your system.

9.2.1 Planning the Auditing Implementation

To plan the auditing implementation, follow these steps:

1. Determine which users to audit. By default, all users are selected for auditing.
2. Determine which events or system calls to audit. Use the `audevent` command to display a list of events and system calls that are currently selected for auditing.

   Events and system calls can be grouped into profiles. For more information on profiles, see Section 9.4.
3. Decide where you want to place the audit log files (audit trails) on the system. For more information on configuring the audit log files, see Section 9.5.
4. Create a strategy to archive and back up audit files. Audit files can take up a considerable amount of disk space and can overflow the file system partition if you do not carefully plan file management. Use the `-X` option with the `audomon` command to automate archiving.

For additional information about auditing system performance and administration that can help you plan the auditing implementation, see Section 9.2.5 and Section 9.2.6.

9.2.2 Enabling Auditing

To enable auditing on the system, follow these steps:

1. Configure the users you want to audit using the `userdbset` command. For more information on configuring auditing for users, see Section 9.3.
2. Configure the events you want to audit using the `audevent` command. For example, to audit according to MySitePolicy, enter the following command:

   ```
   # audevent -P -F -r MySitePolicy
   ```

   MySitePolicy must be defined in the `/etc/audit/audit_site.conf` file.

   Use the `audevent` command with no options to display a list of events and system calls that are currently configured for auditing.

   For more information on configuring auditing for events, see Section 9.4.
3. Set the `audevent` argument parameters in the `/etc/rc.config.d/auditing` file to enable the auditing system to retain the current configuration parameters when the system is rebooted. For example to retain the parameters configured in step 2, set the parameters as follows:

   ```
   AUDEVENT_ARGS1 = -P -F -r MySitePolicy
   ```
4. Start the auditing system and define the audit trail(s) using the `audsys` command:

   ```
   #audsys -n -c primary_audit_file -s 1000
   ```

   For more information on configuring audit trails, see Section 9.5.1.
5. Set up the log files and log file switch parameters in the `/etc/rc.config.d/auditing` file. Follow these steps:
   a. Set `PRI_AUDFILE` to the name of the primary audit log file.
   b. Set `PRI_SWITCH` to the maximum size of the primary audit log file (in KB), at which audit logging switches to the auxiliary log file.

c. Set `SEC_AUDFILE` to the name of the auxiliary log file.
d. Set `SEC_SWITCH` to the maximum size of the secondary audit log file (in KB).

For more information about setting up primary and auxiliary audit log files, see Section 9.5.

6. Start the `audomon` daemon if it has not yet been started. The `audomon` daemon monitors the growth of the current audit trail and switches to an alternative audit trail whenever necessary. For example:

```
#audomon -p 20 -t 1 -w 90 -X "/usr/local/bin/rcp_audit_trail hostname"
```

   For more information about configuring the `audomon` daemon, see Section 9.5.2
7. Set the *`audomon`* argument parameter in the `/etc/rc.config.d/auditing` file to retain the current settings across system reboots.
8. Set the `AUDITING` flag to `1` in the `/etc/rc.config.d/auditing` file to enable the auditing system to automatically start when the system is booted.

9.2.3 Disabling Auditing

To disable auditing on the system, follow these steps:

1. Stop system auditing using the following command:

```
# audsys -f
```

2. Set the `AUDITING` flag to `0` in the `/etc/rc.config.d/auditing` file to prevent the auditing system from starting when the system is rebooted.
3. (Optional) To stop the `audomon` daemon, enter:

```
# kill `ps -e | awk '$NFS~ /audomon/ {print $1}'`
```

   Only use this step if you want to reconfigure the `audomon` daemon. To reconfigure and restart the `audomon` daemon, follow `step 6` and `step 7` as described in Section 9.2.2.

9.2.4 Monitoring Audit Files

To view, monitor, and administer the audit files, follow these steps:

1. View the audit log files with the `audisp` command:

```
# audisp audit_file
```

   See "Viewing Audit Logs" for details on using the `audisp` command.
2. Set the audit log file monitor arguments in the `/etc/rc.config.d/auditing` file.
3. (Optional) Stop system auditing using the following command:

```
#audsys -f
```

The `audsys -f` command lets you stop the system auditing while keeping the `audomon` daemon running.

4. (Optional) Set the `AUDIT` flag to `0` in the `/etc/rc.config.d/auditing` file to keep the auditing system from starting at the next system reboot.

9.2.5 Performance Considerations

Auditing increases system overhead. When performance is a concern, be selective about what events and users are audited. This can help reduce the impact of auditing on performance.

9.2.6 Guidelines for Administering the Auditing System

Use the following guidelines when administering the system:

- Check the audit logs according to the security policy. For example, a security policy might state that an online audit file should be retained for at least 24 hours and all audit records stored offline should be retained for a minimum of 30 days.
- Review the audit log for unusual activities, such as: late hours login, login failures, failed access to system files, and failed attempts to perform security-relevant tasks.
- Prevent the overflow of the audit file by archiving daily.
- Revise current selectable events periodically, especially after installing new releases of HP-UX, since new system calls are often introduced in new releases.
- Revise audited users periodically.
- Do not follow any pattern or schedule for event or user selection.
- Set site guidelines. Involve users and management in determining these guidelines.
- If the audit data volume is expected to be high, configure audit trails on a logical volume consisting of multiple physical disks and multiple physical I/O cards. Use the -N option with `audsys` command to split the audit trail into multiple files.

9.3 Auditing Users

By default, when system auditing is on, all users are audited. New users added to the system are automatically audited.

You can monitor what users are doing on HP-UX systems by inspecting the audit trail. To change which users are audited, choose one of the following options:

- Audit all users.

  Perform the following steps to enable auditing for all users:

  1. Set `AUDIT_FLAG=1` in the `/etc/default/` security file to enable auditing globally for all users.
  2. Run the command `userdbget -a | grep AUDIT_FLAG=0` to determine for which users, if any, auditing is disabled on a per-user basis. Then run the

command `userdbset -u user AUDIT_FLAG=1` or `userdbset -d -u user AUDIT_FLAG` for each of those users.

By default, auditing is enabled for all users when the audit system is turned on. New users added to the system are automatically audited.

If auditing is turned off for all users, set `AUDIT_FLAG=1` in the `/etc/default/security` file.

- Do not audit any users.

  Perform the following steps to disable auditing for all users:

  1. Set `AUDIT_FLAG=0` in the `/etc/default/` security file to disable auditing globally for all users.
  2. Run the command `userdbget -a | grep AUDIT_FLAG=1` to determine for which users, if any, auditing is enabled on a per-user basis. Then run the command `userdbset -u user AUDIT_FLAG=0` or `userdbset -d -u user AUDIT_FLAG` for each of those users.

- Audit specific users.

  Perform the following steps to enable auditing for only certain users:

  1. Set `AUDIT_FLAG=0` in the `/etc/default/` security file to disable auditing globally for all users.
  2. Run the command `userdbget -a | grep AUDIT_FLAG=1` to determine for which users, if any, auditing is enabled.

     To disable auditing for any of those users listed as being audited, run the command `userdbset -u user AUDIT_FLAG=0` or `userdbset -d -u user AUDIT_FLAG`.

     To enable auditing for those users with no per-user `AUDIT_FLAG` attribute set, run the command `userdbset -u user AUDIT_FLAG=1`.

If the audit system is not already enabled, use the `audsys -n` command to start the auditing system. Auditing changes take effect at the user's next login.

9.4 Auditing Events

An event is an action with security implications, such as creating a file, opening a file, or logging in to the system. You can audit events on an HP-UX system to enhance security by detecting possible breaches. However, the more events you choose to audit, the more system resources are used and the greater the impact on system performance. The security architect must determine which events to audit based on business needs and any applicable government regulations.

The `audevent` command is used to specify system activities (auditable events) that are to be audited. Auditable events are classified into event categories and profiles for easier configuration. A profile consists of a set of operations (event categories, self-auditing events, and system calls) that affect a particular type of system. An event category consists of a set of operations (self-auditing events and system calls) that affect a particular aspect of the system. Once an event category or a profile is selected, all system calls and self-auditing events associated with the event category or profile are selected. When the auditing system is installed, a default set of event classification information is provided in the `/etc/audit/audit.conf` file. Additional, site-specific classifications and profiles may also be defined in the `/etc/audit/audit_site.conf` file.

NOTE:
HP recommends that you audit the following event categories at a minimum:

- admin event
- login event
- moddac self-auditing event
- execv, execve
- pset event

These event categories are predefined as the *basic* profile in the `/etc/audit/audit.conf` file.

Configure the events you want to audit before you turn on the auditing system. The syntax for the `audevent` command is as follows:

```
# audevent [options]
```

Changes made by running the `audevent` command take effect immediately.

The following options are commonly used with the `audevent` command:

Table 9-4 audevent Command Options

audevent options	Description
`-e` *event*	Specifies an event to log.
`-F`	Logs unsuccessful event operations.
`-l`	Displays a complete list of event categories and associated system calls.
`-P`	Logs successful event operations.
`-r` *profile*	Specifies the profile of events to log. Profiles are defined in the `/etc/audit/audit.conf` file.
`-S` or `-s` *system_call*	Changes event or system call audit status.
no option	Displays the current status of the selected events or system calls.

To configure the `admin`, `login`, and `modaccess` event categories for auditing, enter the following command:

```
# audevent -P -F -e admin -e login -e moddac
```

To configure the events associated with the basic profile for auditing, use the following command:

```
# audevent -P -F -r basic
```

Both `Audit Success` and `Audit Failure` are set as event types for monitoring successful and failed events or system calls. Monitoring these three event categories is the minimum event type selection recommended for running a system.

Generally, a record is written only if both the event is selected for auditing, and the user initiating the event has been selected for auditing. However, it is expected that some records may still be generated at the time user starts a session and ends a session, even if the user is not selected for auditing. Those records are considered system-wide information that are based on event selection instead of user selection. Programs that do self-auditing can choose to ignore the user selection, but this is not recommended.

9.5 Audit Trails

All auditing data is written to an audit trail. In regular mode, an audit trail is stored on a file system in one or more log files that reside in the same directory. The number of log files is directly proportional to the number of kernel threads that are configured for logging audit records (see the `audsys -N` option). All the files in the directory are needed for meaningful analysis or display. Contrary to regular mode, a compatibility mode is also provided in the HP-UX 11i version 3 release to generate audit trail that is stored in a single file. The compatibility mode is solely supported for backward compatibility and will be obsolete in releases after HP-UX 11i Version 3. See *audsys*(1M) for more information.

When the auditing system is enabled, there must be at least one audit trail pathname specified. The trail pathname and various attributes for the trail can be specified using the `audsys` command. When the current trail exceeds a predefined capacity (its Audit File Switch (AFS) size), or when the auditing 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 is reached, the auditing subsystem looks for an auxiliary trail. If one is available, recording is switched to the auxiliary trail. If no auxiliary trail is specified, the auditing subsystem creates a new audit trail with the same base name but a different timestamp extension and begins recording to it. The `audomon` command can be invoked with an option (`-X`) that specifies a command line to run after a successful audit trail switch to perform some action. Depending on site-specific needs, the command may perform audit trail backup, archival, off site transfer, cleaning up or data reporting. If the audit trail switch is unsuccessful, warning messages are sent to request appropriate administrator action and the current audit trail continues to grow.

NOTE:

1. With HP-UX 11i version 3, an auxiliary audit trail does not need to be specified; the auditing system does switching of audit trails automatically.
2. If autoswitching failed and the current audit trail continues to grow past the FSS point, all auditable actions are suspended for regular users. The system can be restored by archiving the audit data, or specifying a new audit log file on a file system with space.
3. If other activities consume space on the file system, or the file system chosen has insufficient space for the AFS size chosen, the File Space Switch point can be reached before the Audit File Switch point.

Choose a file system with adequate space for the audit log files. You can assess the size of the file systems using the `bdf` command. HP recommends you configure the log files to reside on a file system with at least 5 MB of available space and with at least 20% of its total file space available.

The growth of audit log files is closely monitored by the audit overflow monitor daemon, `audomon`, to insure that no audit data is lost.

9.5.1 Configuring Audit Trails

Use the `audsys` command to specify the primary audit log file and the (optional) auxiliary audit log file to collect auditing data:

```
#audsys -n -N2 -c my_audit_trail -s 5000
```

This example starts the audit system and records data in the `my_audit_trail` directory, using two writer threads. The AFS size is set to 5000K bytes.

The `audsys` command recognizes the following options:

`-c` *file|directory*	Specifies a "current" trail.
`-f`	Turns off the auditing system.
`-n`	Turns on the auditing system.
`-N` *num*	Specifies the number of active files that comprise an audit trail.
`-s` *cafs*	Specifies `cafs`, the "current" trail's AuditFileSwitch (AFS) size (in kbytes).
`-x` *file|directory*	Specifies the "next" audit trail.
`-z` *xafs*	Specifies `xafs`, the "next" trail's AuditFileSwitch (AFS) size (in kbytes).

For more information, see *audsys*(1M) .

9.5.2 Monitoring and Managing Audit Trails

The audit overflow monitor daemon (`audomon`) is used to monitor and manage audit trails. The `audomon` daemon is started automatically when auditing is started at system boot time (`AUDITING=1` in `/sbin/init.d/auditing`). The `audomon` daemon can also be started by a privileged user. Once started, the `audomon` daemon monitors the capacity of the current audit trail and the file system it resides on. The following is an example of how to start the `audomon` daemon:

```
# audomon -p 20 -t 1 -w 90 -X "/user/local/bin/rcp_audit_trail hostname"
```

This command starts the `audomon` daemon with the following behavior, assuming the auditing system was started with the following command:

```
# audsys -n -N 2 -c /var/.audit/my_trail -s 400
```

- `audomon` sleeps at least one minute intervals
- When the size of the current audit trail reaches 4500 Kb, or the file system that the audit trail resides becomes 80% full, the `audomon` daemon stops recording data to the current audit trail and starts recording a new audit trail: `/var/.audit/my_trail.`*`yyyymmddHHMM`*
- After the switch to the new audit trail succeeds, the `audomon` daemon invokes the following command:

  ```
  sh -c "/usr/local/bin/rcp_audit_trail hostname /var/.audit/my_trail"
  ```

  This script is site specific and may be used to copy the old audit trail, perform data backup or archival functions, and create audit reports. For more information about the `audomon` daemon, see *audomon*(1).

CAUTION:

- If the file system containing the audit trail is full, any non root process that generates audit data will block inside the kernel. Also, if a non root process is connected to the system terminal, it will be terminated. For details see the WARNINGS section of *audsys*(1M).
- The root file system should not be used for storing audit trails to avoid filling up the root file system and causing processes to fail or hang.

TIP: HP recommends that you write a script to carry out your long term strategy for data storage and pass it to the `audomon` daemon using the `-X` option.

The `audomon` command recognizes the following options:

`-p` *`fss`*	The minimum percentage of space left on the file system that contains the primary audit log file before the auditing system switches to the auxiliary log file. The default *`fss`* value is 20%.

`-t` *`sp_freq`*	The minimum wakeup interval, in minutes, at which the system prints warning messages on the console for audit log file switch points. The default *`sp_freq`* value is 1 minute.
`-w` *`warning`*	The percentage of audit log file space used or minimum file system free space used after which warning messages are sent to the console. The default *`warning`* value is 90%.
-X *`command`*	The *`command`* is executed each time the `audomon` switches the audit trail.

For more information, see *audomon*(1M).

9.6 Using the Audit Filtering Tools

The **audit filtering tools** are a set of tools that helps customize and enforce the audit data pre-filtering policy on the system. A good pre-filtering policy is an efficient way to control the size and quality of the raw data and therefore minimizes the performance impact of auditing and reduces the operational cost associated with audit data management. The audit filtering tools consist of the following main components:

- A configuration tool, `audfilter`, that interprets the filtering policy as specified in the configuration file, `filter.conf`, and puts the policy into effect. You can also use `audfilter` to display or clear out the filtering policy that is currently in effect.
- A service daemon, `audfilterd`, that handles service requests from `audfilter`. It also tracks the mounted file system changes and makes sure the filtering policy is up to date with the new mounted file system information.
- A dynamic loadable kernel module, `audit_filters`, that makes filtering decisions and enforces the filtering policy in the kernel.

The following options are available with the `audfilter` command:

`-c`	Puts the current rule-based audit filtering policy as specified in `/etc/audit/filter.conf` into effect. Rules are parsed into an efficient internal format. Note that a given set of rules may be expressed in many different ways, but they are all parsed into the same internal format. A success or failure status will be reported for the request.
`-C` *`compartment`*	Only displays the filtering rules for the specified compartment. This option must be specified with the `-p` or `-P` option.
`-c` *`system_call`*	Displays the selected system call.
`-m` *`mntpnt`*	Only displays the filtering rules for the specified mount point. This option must be specified with the `-p` or `-P` option.
`-p`	Displays the audit filtering policy currently in effect. The rules are not displayed the same way as they were written, but in the order they are evaluated (that is, in the internal format).

`-P`	Displays audit filtering policy in preview mode as specified in the `/etc/audit/filter.conf` file. This option parses the `/etc/audit/filter.conf` file, checking for syntax and semantic errors, but makes no changes to the system. The rules will not be displayed the same way as they are written, but in the order they will be evaluated (that is, in the internal format).
`-s` *`syscall`*	Restricts the display to the given system call. This option must be used with the `-p` or `-P` option.
`-z`	Clears the audit filtering policy currently in effect. Upon success, it effectively disables finer grained audit filtering feature.

For more information, see *audfilter*(1M)

9.7 Using filter.conf

The `filter.conf` file contains rule-based audit filtering policy that the auditing subsystem uses to determine what activities to audit on the system. Each rule consists of two parts: a scope and a condition. All rules together represent a policy.

A scope is an identifier for a mounted file system (also called a partition) for file operations, or an identifier for non-file operations. The scope can be any of the following forms:

a. The absolute pathname of a mount point that matches one of those in the `/etc/mnttab` file.

b. A pair of major and minor device numbers.

c. Special file name or a pair of hostname and pathname of the directory on the remote host.

d. Scope in the form of a), b), or c), followed by the keyword *`all`*.

e. Scope in the form of a), b), or c), followed by the keyword *`other`*.

f. Scope in the form of the keyword *`other-objects`*.

See *filter.conf*(4) for more information.

9.8 Using the Audit Reporting Tools

The **audit reporting tools** are a set of tools that facilitates the processing of previously collected HP-UX raw audit data and extracts useful information for compliance reporting purposes. The audit reporting tools consist of the following main components:

- An audit data processing tool, `auditdp`, that selectively extracts audit data from a data source in one of several possible formats and writes the data to the target, in the same or different format.
- An Audit Data Process Module Switch (Audit DPMS) framework that offers the ability to selectively access audit data in various formats through a set of common programming interfaces.

- An Audit DPMS service module, `audit_hpux_raw`, that reads raw audit data collected by the HP-UX auditing system.
- An Audit DPMS service module, `audit_hpux_portable`, that handles audit data that is portable across HP-UX systems, and good for retention purpose. Also a sample script, `audit_p2l`, that demonstrates how to convert the portable data into syslog-like messages.
- An Audit DPMS service module, `audit_hpux_xml`, that converts audit data into XML format. Also a sample script, `audreport_generator`, that demonstrates how to use the `auditdp` command and the XSLT stylesheets to generate a collection of web-based audit reports for regulation compliance purposes.
- Sample audit report configuration templates for PCI and SOX that work with the `audreport_generator` script.

The following options are available with the `auditdp` command:

`-M` *`module [target]`*	Write audit data to the target using the specified Audit DPMS service module. The target is the pathname of a file where to write the data. The file must not already exist. If the target is omitted, auditdp writes the audit data to the standard output.
`-O` *`option`*	Specify the options (case insensitive) to be passed to the Audit DPMS framework when writing to the target.
`-P` *`[target]`*	Write audit data in portable format. Portable format audit data can be ported from system to system and is the recommended format for retention purposes. The target is the pathname of a file where to write the data. The file must not already exist. If the target pathname is not absolute, then the target is assumed to be relative to the current directory. If the target is omitted, `auditdp` writes the audit data to the standard output.
`-S` *`filter_file`*	Selectively process audit data based on the Audit DPMS configuration file specified in `filter_file`. Only the audit data matching the filtering criteria will be included in the target output.
`-X` *`[target]`*	Write audit data in Extensible Markup Language (XML) format. The target is the pathname of a file where to write the data. The file must not already exist. If the target pathname is not absolute, the pathname is assumed to be relative to the current directory. If the target is omitted, `auditdp` writes the audit data to the standard output. Applying Extensible Stylesheet Language Transformations (XSLT) stylesheets to the resulting XML document generates "human-readable" web-based audit reports.

`-m` *`module[source]`*	Read audit data from the source using the specified Audit DPMS service module. The source is the pathname of a file where to read the data. If the source is omitted, `auditdp` reads the audit data from the standard input.
`-n` *`nevents`*	Specify the number of events to display. If `nevents` is positive, process only the first `nevents` events. If `nevents` is negative, process only the last `nevents` events. If `-n` is not specified, all events are processed.
`-o` *`options`*	Specify the option (case insensitive) to be passed to the Audit DPMS framework when reading from the source. To specify more than one option, use `-o` multiple times, or set option to a quoted string containing a list of options separated by spaces.
`-p` *`[source]`*	Read portable format audit data. The source is the pathname of a file where to read the data. If the source pathname is not absolute, the pathname is assumed to be relative to the current directory. If the source is omitted, `auditdp` reads the audit data from the standard input.
`-r` *`[source]`*	Read HP-UX raw audit data that was collected by the HP-UX auditing system (see *audit*(5)). The source specifies the pathname to a file if the data was collected in compatibility mode, or to a directory if the data was collected in regular mode. If the source pathname is not absolute, the pathname is assumed to be relative to the current directory.
`-s` *`filter_string`*	Selectively process audit data based on the filter expression specified in the `filter_string`.

For more information, see *auditdp*(1M)

9.8.1 Examples of Using the auditdp Command

The following examples show audit information displayed using the `auditdp` command:

- Read raw data from `audit_trail` and write portable data to `./portable`.

  ```
  #auditdp -r /var/.audit/audit_trail -P portable
  ```

- Read raw data and write data in the `audisp` display format to stdout (see Section 9.9).

  ```
  #auditdp -r /var/.audit/audit_trail
  ```

- Read portable data and display only the last four events to stdout.

  ```
  #auditdp -p portable -n -4
  ```

- Read and then write portable data, saving only the login events.

```
#auditdp -p portable -P portable2 -s "+event=login"
```

- Extract exec events from a particular session and write to stdout:

  ```
  #auditdp -r /var/.audit/audit_trail -s "+sid=1234" -P | \
  auditdp -p -s "+event=exec"
  ```

  or

  ```
  #auditdp -r /var/.audit/audit_trail -s "+sid=1234;+event=exec"
  ```

9.9 Viewing Audit Logs

Auditing can generate a significant amount of data. Use the `audisp` command to select the data that you want to view:

```
#/usr/sbin/audisp audit_trail
```

NOTE: The `audisp` command will be obsolete in a future release. Invoking `/usr/sbin/auditdp -r audit_trail` produces the same output as `/usr/sbin/audisp audit_trail`.

The following options are available with the `audisp` command:

`-f`	Displays failed events only.
`-p`	Displays successful events only.
`-c` *system_call*	Displays the selected system call.
`-t`	Display events that occurred after the given time.
`-s`	Displays events that occurred before the given time.
`-u` *user-name*	Displays information for a specific user.
`-l` *terminal-name*	Displays information for a specific terminal.
`-e` *event-name*	Displays information for the given event.
`>` *file-name*	Writes output to specified file.

It can take a few minutes to prepare the record for viewing when working with large audit logs. When viewing the audit data, be aware of the following anomalies:

- Audit data can appear inaccurate when programs that call auditable system calls supply incorrect parameters. The audit data shows what the user program passed to the kernel. For example, calling the `kill` system call with no parameters produces unpredictable values in the parameter section of the audit record.
- System calls that take file name arguments may not have device and inode information properly recorded. The values will be `-1` if the call does not complete successfully.
- Auditing the superuser while changing the event or system call audit parameters will result in a long audit record. For example, when you add an event type to be audited,

a record will be produced for each event type and system call that has been enabled for audit, not just for the new event type being added.

9.9.1 Examples of Using the audisp Command

The following examples show audit information displayed using the `audisp` command:

- Display the log output on the screen:

  ```
  #/usr/sbin/audisp audit_trail
  ```

- Direct the log output to `/tmp/mylogoutput`:

  ```
  #/usr/sbin/audisp audit_trail > /tmp/mylogoutput
  ```

- View successful events only:

  ```
  #/usr/sbin/audisp -p audit_trail
  ```

- View activities owned by user joe:

  ```
  #/usr/sbin/audisp -u joe audit_trail
  ```

- View activities on terminal, `ttypa`:

  ```
  #/usr/sbin/audisp -l ttypa audit_trail
  ```

- View login events only:

  ```
  #/usr/sbin/audisp -e login audit_trail
  ```

9.10 Self-Auditing

Some processes invoke a series of actions that can be audited. 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; using self-auditing programs streamlines audit log data.

NOTE: The list of self-auditing processes varies from system to system.

Self-auditing processes

The following processes have self-auditing capabilities:

`chfn`	Change `finger` entry
`chsh`	Change login shell
`login`	The login utility
`newgrp`	Change effective group
`passwd`	Change password

`audevent`	Select events to be audited
`audisp`	Display the audit data
`audsys`	Start or halt the auditing system
`audusr`	Select users to be audited
`init`	Change run levels, users logging off
`lpsched`	Schedule line printer requests
`fbackup`	Flexible file backup
`ftpd`	File transfer protocol daemon
`remshd`	Remote shell server daemon
`rlogind`	Remote login server daemon
`telnetd`	Telnet server daemon
`privrun`	Invokes legacy application.[1]
`privedit`	Allows authorized users to edit files.[1]
`roleadm`	Edits role information.[1]
`authadm`	Edits authorization information.[1]
`cmdprivadm`	Edits command authorizations and privileges.[1]

Most self-auditing programs generate audit data under a single event category. For example, the `audsys` command generates the audit data under the `admin` event. Some commands generate audit data under multiple event categories. For example, the `init` command generates data under the `login` and `admin` events.

9.11 HP-UX RBAC Auditing

The `privrun`, `privedit`, `roleadm`, `authadm`, and `cmdprivadm` HP-UX RBAC commands each generate audit records. The following attributes are included in each audit record:

- User name
- UID
- Role
- Authorizations (operation, object)
- Time of event
- Result of event (success or failure)

9.11.1 Auditing Based on HP-UX RBAC Criteria and the /etc/rbac/aud_filter File

HP-UX RBAC Version B.11.23.02 and later support the use of an audit filter file to identify specific HP-UX RBAC criteria to audit. You can create a filter file named `/etc/rbac/aud_filter` to identify specific roles, operations, and objects for which

1. See Chapter 8 for more information.

to generate audit records. Audit records are generated only if the attributes of a process match all three entries (role, operation, and object) found in `/etc/rbac/aud_filter`. If a user's role and associated authorization are not found in the file or do not explicitly match, then no audit records specific to role-to-authorization are generated.

Authorized users can edit the `/etc/rbac/aud_filter` file using a text editor and specify the role and authorization to be audited. Each authorization is specified in the form of operation, object pairs. All authorizations associated with a role must be specified in a single entry. Only one authorization can be specified per role on each line; however, the `*` wildcard is supported. The following are the supported entries and format for the `/etc/rbac/aud_filter` file:

`role`, *`operation`*, *`object`*

The following list explains each of the `/etc/rbac/aud_filter` entries:

`role`	Any valid role defined in `/etc/rbac/roles`. If `*` is specified, all roles can be accessed by the operation.
`operation`	A specific operation that can be performed on an object. For example, `hpux.printer.add` is the operation of adding a printer. Alternatively, `hpux.printer.*` is the operation of either adding or deleting a printer. If `*` is specified, all operations can be accessed by the operation.
`object`	The object the user can access. If `*` is specified, all objects can be accessed by the operation.

The following are example `/etc/rbac/aud_filter` entries that specify how to generate audit records for the role of `SecurityOfficer` with the authorization of (`hpux.passwd, /etc/passwd`), and for the `Administrator` role with authorization to perform the `hpux.printer.add` operation on all objects.

```
SecurityOfficer, hpux.passwd, /etc/passwd
Administrator, hpux.printer.add, *
```

NOTE: Use an editor such as `vi` to directly edit the `/etc/rbac/aud_filter` file. The HP-UX RBAC administrative commands do not provide an interface to configure `/etc/rbac/aud_filter`.

For detailed information about RBAC, roles, operations, and objects, see Chapter 8

9.11.2 Procedure for Auditing HP-UX RBAC Criteria

The following steps describe how to configure an audit process to audit HP-UX RBAC criteria on the system:

1. Configure the system to audit Passed or Failed events for the Administrator events by using the following command:

   ```
   # audevent -PFe admin
   ```

2. Configure the location and name of the audit output file and enable auditing on the system by using the following command:

   ```
   # audsys -n -c /tmp/aud.out -s 2048
   ```

3. Execute an HP-UX RBAC command, for example:

   ```
   # /usr/sbin/authadm add newauth
   ```

4. Open the audit output file and search for the records on the `authadm` command by using the following command:

   ```
   # audisp /tmp/aud.out |fgrep authadm
   ```

NOTE: For more information, see *audit*(5), *audevent*(1M), *audsys*(1M), and *audisp*(1M) to learn more about auditing HP-UX systems.

A Trusted Systems

This appendix describes how to set up and manage a trusted system. This appendix discusses the following topics:

- Setting up a trusted system (Section A.1)
- Auditing a trusted system (Section A.2)
- Managing trusted passwords and system access (Section A.3)
- Guidelines for trusted backup and recovery (Section A.4)

NOTE: Trusted Systems has been depreciated. HP-UX 11i v3 is the last release that supports this product.

A.1 Setting Up a Trusted System

To set up a trusted system, follow these steps:

1. Establish an overall security policy appropriate to your work site.
2. Inspect all existing files on the system for security risks, and remedy them. This is important before you convert to a trusted system. Thereafter, examine the files regularly, or when you suspect a security breach. See Section 5.9 in Chapter 5
3. Back up the file system for later recovery of user files. You should also back up the `/etc/passwd` file to tape before the conversion.

   You can use any of the backup and recovery programs provided by HP-UX for the initial backup and recovery. After security features are implemented, however, use only `fbackup` and `frecover`, which preserve and restore access control lists (ACLs). For more information, see *fbackup*(1M) and *frecover*(1M).
4. Convert to a trusted system. Conversion to a trusted system is a reversible operation.

   To convert to a trusted system, run HP SMH and click **System Security Policies**. It will get to the `Convert to trusted system` prompt. You might receive a confirmation prompt. Press **`Y`** to begin the conversion process.

   When you convert to a trusted system, the conversion program does the following actions:

   - Creates a new, protected password database in `/tcb/files/auth/`.
   - Moves encrypted passwords from the `/etc/passwd` file to the protected password database and replaces the password field in `/etc/passwd` with an asterisk (`*`).
   - Forces all users to use passwords.
   - Creates an audit ID number for each user. The audit ID remains unchanged throughout a user's history. It uniquely identifies a user. Note that audit ID is getting deprecated along with Trusted System in HP-UX 11i v3, and is being replaced by audit tag that is dynamically assigned each time a user successfully starts a new login session. See Chapter 9 for more information about audit tags.

- Turns on the audit flag for all existing users.
- Converts the `at`, `batch`, and `crontab` input files to use the submitter's audit ID.

5. Verify that the audit files are on the system:
   1. Use `swlist -l fileset` to list the installed file sets. Look for the fileset called `SecurityMon`, which contains the auditing program files. To reduce the listing, enter the following command:# **`swlist -l fileset | grep Security`**
   2. In addition, verify that the following files (not specified in `SecurityMon`) also exist:
      - `/etc/rc.config.d/auditing` contains parameters to control auditing. You can modify this file with SMH or by hand with a text editor.
      - `/sbin/rc2.d/S760auditing` is the script that starts auditing. Do not modify this file.
6. After converting to a trusted system, you can use the audit subsystem and run the HP-UX system as a trusted system.

NOTE: On HP-UX 11i v3, an auditing system also works on a system without converting to a trusted system.

See Chapter 9 for more information.

If you need to convert from a trusted system back to a standard system, run HP SMH and use the **Auditing and Security** window. The **Audited Events**, **Audited System Calls**, and **Audited Users** screens all provide an unconvert option.

TIP: One way to determine if the system has been converted to a trusted system is to look for`/tcb` files. If they exist, then you have a trusted system.

A.2 Auditing a Trusted System

Auditing a trusted system is very similar to auditing a system that has not been converted to trusted mode. See Chapter 9 for the information on auditing. The only difference is how to select audited users. On a system that has not been converted to trusted mode, the `userdbset` command is used to specify those users who are to be audited. See *userbdset*(1M) and *userdb*(4). The associated attribute is called `AUDIT_FLAG` and is described in *security*(4). On a trusted system, the `audusr` command specifies those users who are to be audited. See *audusr*(1M) for more information.

A.3 Managing Trusted Passwords and System Access

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

Also see Chapter 2 for password information.

Security Administrator's Responsibilities

The security administrator and every user on the system must share responsibility for password security. The security administrator performs the following security tasks:

- Generates temporary passwords for new users. This password must be used for first login. When this number has been verified, the new user is prompted for a new password.
- Maintains proper permissions on all system files, including the standard password file, `/etc/passwd`, and the trusted database files, `/tcb/files/auth/*`.
- Establishes password aging.
- Manages password reuse.
- Deletes or nullifies expired passwords, user IDs, and passwords of users no longer eligible to access the system.

User's Responsibilities

Every user must observe the following rules:

- Remember the password and keep it secret at all times.
- Change the initial password immediately; thereafter, change the password regularly.
- Report any changes in status and any suspected security violations.
- Make sure no one is watching when you enter the password.
- Choose a different password for each machine on which you have an account.

A.3.1 Password Files

A trusted system maintains multiple password files: the `/etc/passwd` file and the files in the protected password database `/tcb/files/auth/` (see "The /tcb/files/auth/ Database"). Each user has an entry in two files, and `login` looks at both entries to authenticate login requests.

All passwords are encrypted immediately after entry and stored in `/tcb/files/auth/`*`user-char`*`/`*`user-name`*, the user's protected password database file. Only the encrypted password is used in comparisons.

Do not permit any empty (null) password fields in either password file. On trusted systems, the password field in `/etc/passwd` is ignored. A user with an empty password will be forced to set a password upon login on a trusted system. However, even this leaves a potential for a security breach, anyone logging in to this account is required to set the password.

Do not edit the password files directly. Use HP SMH, `useradd`, `userdel`, or `usermod` to modify password file entries.

A.3.1.1 The /etc/passwd File

A trusted system uses the `/etc/passwd` file to identify a user at login time. The file contains an entry for every account on the HP-UX system. Each entry consists of seven fields, separated by colons. A typical entry for `/etc/passwd` in a trusted system looks like this:

```
robin:*:102:99:Robin Hood,Rm 3,x9876,408-555-1234:/home/robin:/usr/bin/sh
```

The fields contain the following information (listed in order), separated by colons:

1. User (login) name, consisting of up to 8 characters. (In the example, `robin`)
2. Unused password field, held by an asterisk instead of an actual password. (`*`)
3. User ID, an integer ranging from 0 to `MAXINT-1`, equal to 2,147,483,646 or 2^{31} -2. (`102`)
4. Group ID, from `/etc/group`, an integer ranging from 0 to `MAXINT-1`. (`99`)
5. Comment field, used to identify such information as the user's full name, location, and phone numbers. For historic reasons, this is also called the `gecos` field. (`Robin Hood,Rm 3,x9876,408-555-1234`)
6. Home directory, the user's initial login directory. (`/home/robin`)
7. Login program path name, executed when the user logs in. (`/usr/bin/sh`)

The user can change the comment field (fifth field) with the `chfn` command and the login program path name (seventh field) with the `chsh` command. The system administrator sets the remaining fields. The user ID should be unique. For more information, see *chfn*(1), *chsh*(1), *passwd*(1), and *passwd*(4). The user can change the password in the protected password database with `passwd`.

A.3.1.2 The /tcb/files/auth/ Database

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 protected password database, and an asterisk holds its place in the `/etc/passwd` file.

Protected password database files are stored in the `/tcb/files/auth/` hierarchy. User authentication profiles are stored in these directories based on the first letter of the user account name. For example, the authentication profile for user `david` is stored in the file `/tcb/files/auth/d/david`.

On trusted systems, key security elements are held in the protected password database, accessible only to superusers. Use HP SMH to set password data entries. Password data that is not set for a user will default to the system defaults stored in the file `/tcb/files/auth/system/default`.

The protected password database contains many authentication entries for the user. See *prpwd*(4) for more information on these entries, which include the following:

- User name and user ID
- Encrypted password
- Account owner
- Boot authentication to allow specified users to boot the system; see *security*(4).
- Audit ID and audit flag for the user (whether audit is on or not)
- Minimum time between password change
- Password maximum length
- Password expiration time, after which the password must be changed
- Password lifetime, after which the account is locked

- Time of last successful and unsuccessful password changes
- Absolute time (date) when the account will expire
- Maximum time allowed between logins before the account is locked
- Number of days before expiration when a warning will appear
- Whether passwords are user-generated or system-generated
- Password triviality check to prevent common words or well-known terms from being used as passwords
- Type of system-generated passwords
- Null passwords
- User ID of last person to change password, if not the account owner
- Time periods when this account can be used for login
- Identification of terminal or remote hosts associated with the last successful and unsuccessful logins to this account
- Number of unsuccessful login attempts; cleared upon successful login
- Maximum number of login attempts allowed before account is locked

A.3.2 Password Selection and Generation

On trusted systems, the following password generation options are available:

- User-generated passwords.

  A password screening option is available to check for the use of login and group names, login and group name permutations, and palindromes.

  A new password must differ from the old password by at least 3 characters.

- System-generated passwords using a combination of letters only.
- System-generated passwords using a combination of letters, numbers, and punctuation characters.
- System-generated passwords using pronounceable meaningless syllables.

You can set password generation options for a system. Alternately, you can set password generation options on a per-user basis, overriding the system default.

You must set at least one password generation option for each user. If more than one option is available to a user, a password generation menu is displayed when the user changes the password.

A.3.3 Password Aging

You can enable or disable password aging for each user. When password aging is enabled, the system maintains the following for the password:

Minimum time	The minimum time required between password changes. This prevents a user from changing the password and then changing it back immediately to avoid memorizing a new one.
Expiration time	A time after which a user must change that password at login.
Warning time	The time before expiration when a warning will be issued.

Lifetime	The time at which the account associated with the password is locked if the password is not changed. Once an account is locked, only the system administrator can unlock it. Once unlocked, the password must still be changed before the user can log into the account.

The expiration time and lifetime values are reset when a password is changed. A lifetime of zero specifies no password aging; in this case, the other password aging times have no effect.

A.3.4 Password History and Password Reuse

You can enable the password history feature on a systemwide basis to discourage users from reusing previous passwords.

You enable the password reuse check by defining the `PASSWORD_HISTORY_DEPTH` attribute in the `/etc/default/security` file:

```
PASSWORD_HISTORY_DEPTH=n
```

where *n* is an integer specifying the number of previous passwords to check.

When a user changes the password, the new password is checked against the previous *n* passwords, starting with the current password. If the system finds a match, it rejects the new password. An *n* of `2` prevents users from alternating between two passwords.

For more information, see *passwd*(1) and *security*(4).

A.3.5 Time-Based Access Control

On trusted systems, you can specify times-of-day and days-of-week that are allowed for login for each user. When a user attempts to log in outside the allowed access time, the event is logged (if auditing is enabled for login failures and successes) and the login is terminated. A superuser can log in outside the allowed access time, but the event is logged. The permitted range of access times is stored in the protected password database for users and can be set with HP SMH. Users that are logged in when a range ends are not logged out.

A.3.6 Device-Based Access Control

For each MUX port and dedicated DTC port on a trusted system, you can specify a list of users allowed for access. When the list is null for a device, all users are allowed access.

The device access information is stored in the device assignment database, `/tcb/files/devassign`, which contains an entry for each terminal device on the trusted system. A field in the entry lists the users allowed on the device.

Terminal login information on a trusted system is stored in the terminal control database, `/tcb/files/ttys`, which provides the following data for each terminal:

- Device name
- User ID of the last user to successfully log into the terminal
- Last successful login time to the terminal

- Last unsuccessful login time to the terminal
- Number of consecutive unsuccessful logins before terminal is locked
- Terminal lock flag

Only superusers can access these trusted system databases and can set the entries using HP SMH. See *devassign*(4) and *ttys*(4).

A.3.7 Manipulating the Trusted System Databases

Use the library routines described in the following manpages to access information in the password files and in other trusted system databases:

getdvagent(3)	Manipulates device entries in `/tcb/files/devassign`
getprdfent(3)	Manipulates system defaults in `/tcb/files/auth/system/default`
getprpwent(3)	Gets password entries from `/tcb/files/auth/`
getprtcent(3)	Manipulates terminal control database, `/tcb/files/ttys`
getpwent(3C)	Gets password entries from `/etc/passwd`
putpwent(3C)	Writes password file entries to `/etc/passwd`
getspwent(3X)	Gets password entries from `/tcb/files/auth/` (provided for backward compatibility)
putspwent(3X)	Writes password entries to `/tcb/files/auth/` (provided for backward compatibility)
putprpwnam(3)	Writes password file entries to `/tcb/files/auth/`

A.4 Guidelines for Trusted Backup and Recovery

Following are guidelines for backup and recovery on a trusted system:

- Use only `fbackup` and `frecover` to back up and recover files selectively. Only `fbackup` and `frecover` retain access control lists (ACLs). Use the `-A` option of these commands when backing up and recovering files for use on systems that do not implement ACLs. For more information, see *fbackup*(1M) and *frecover*(1M).
- If you plan to recover the files to another system, be sure that the user's user name and group name on both systems are consistent.
- Remember that the 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 a minimum of 6 months, then recycle the media.
- Use appropriate procedures to clean magnetic media to remove data before reuse.
- Perform daily incremental and full weekly backups.

  Synchronize the backup schedule with the information flow in the organization. For example, if a major database is updated every Friday, you might want to schedule the weekly backup on Friday evenings.

- If all files must be backed up on schedule, request that all users log off before you perform the backup. However, `fbackup` warns you if a file is changing while the backup is being performed.
- Examine the log file of latest backups to identify problems occurring during backup. Set restrictive permissions for the backup log file.
- The `frecover` command allows you to overwrite a file. However, the file retains the permissions and ACLs set when the file was backed up.
- You must test the recovery process beforehand to make sure you can fully recover data in the event of an emergency.
- When recovering files from another machine, you might have to execute the `chown` 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`. If owner and group names have different meanings on different systems, recovery results might be unexpected.
- 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` 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` prevents you from reading the file if the permissions on the file forbid it.
- Never recover in place any critical files such as `/etc/passwd` or the files in `/tcb/files`. Instead, restore the file to a temporary directory (do not use `/tmp`) and give this directory permissions `drwx------`, preventing anyone else from using it. Compare the restored files with those to be replaced. Make any necessary changes.
- Auditing is not enabled automatically when you have recovered the system. Be sure to turn auditing on with the `audsys` command.

B Other Security Products

This appendix includes additional security products available for HP-UX, for the following three security categories:

- "Protecting Systems" (page 199)
- "Protecting Data" (page 200)
- "Protecting Identity" (page 203)

You can download these products for free from the HP Software Depot at:

http://www.hp.com/go/softwaredepot

B.1 Protecting Systems

In addition to the security products that are discussed in *Part I Protecting Systems*, the following security products offer additional system protection.

B.1.1 HP-UX Bastille

HP-UX Bastille is a system hardening and reporting program that enhances the security of the HP-UX operating system by consolidating essential hardening and lock-down checklists from industry and government security organizations, and making them accessible to administrators in an easy to use package.

For more information, see the HP-UX Bastille documentation:

http://www.hp.com/go/hpux-security-docs

Click **HP-UX Bastille Software**.

B.1.2 HP-UX HIDS

HP-UX Host Intrusion Detection System (HIDS) enables security administrators to proactively monitor, detect, and respond to attacks within a network, as follows:

- Protects against both existing attack scenarios and against some as of yet unknown scenarios. It seeks out patterns that might suggest security breaches or misuses by examining information about system activity from a variety of data sources. Such illicit activities might include: a hacker attempting to break into or disrupt your system, subversive "insider" activities, or someone trying to spread a virus
- Detects product enhances local host-level security within your network. It automatically monitors each configured host system within the network for possible signs of unwanted and potentially damaging intrusions. If unchecked it can lead to the loss of availability of key systems or can compromise system integrity. HP-UX HIDS generate alerts for many types of exploits.
- Provides continuous protection against both existing attack scenarios and unknown scenarios unlike other intrusion detection systems. It detects intrusions by using detection templates. Detection templates are the building blocks used to identify the

basic types of unauthorized system activity or security attacks frequently found on enterprise networks.

- Provides notification in the event of suspicious activity that might precede an attack. By contrast, other intrusion detection systems rely entirely on an operator-instigated analysis of the system log files. Typically the operator analyses the system log files at the end of the day. This delay in the analysis of the attack provides considerable time to damage the system.

For more information, see the HP-UX HIDS documentation:

http://www.hp.com/go/hpux-security-docs

Click **HP-UX Host Intrusion Detection System Software**.

B.1.3 HP-UX IPFilter

HP-UX IPFilter is a system firewall that filters IP packets to control packet flow in or out of a machine. It works as a security defense by cutting down on the number of exposure points on a machine.

For more information, see the HP-UX IPFilter documentation:

http://www.hp.com/go/hpux-security-docs

Click **HP-UX IPFilter Software**.

B.1.4 Security Patches

HP-UX Software Assistant (SWA) is a command-line based tool that consolidates and simplifies patch management and security bulletin management on HP-UX systems. The SWA tool is new for HP-UX releases as of January 2007, replaces Security Patch Check (SPC), and is the HP-recommended utility to use to maintain currency with HP-published security bulletins for HP-UX software.

HP provides up-to-date software patches to known security problems that allow unauthorized root access to your system.

For more information, see the HP-UX SWA documentation:

http://www.hp.com/go/hpux-security-docs

Click **HP-UX Software Assistant (SWA) Software**.

B.2 Protecting Data

In addition to the security products that are discussed in *Part II Protecting Data*, the following security products offer additional data protection.

B.2.1 HP-UX Containers (SRP)

HP-UX Containers, formerly Secure Resource Partitions (SRP), allows you to deploy multiple isolated container-based environments on a single server platform. This allows the enterprise to host multiple workloads in a single operating system instance, thereby better utilizing server resources (CPU, memory, network access) and data center resources (power, cooling, footprint), and reducing the overall number of operating system instances that must be managed. HP-UX Containers is a component of the Virtualization Continuum for HP-UX and offers high efficiency in resource utilization and performance.

For more information, see the HP-UX Containers (SRP) documentation:

www.hp.com/go/virtualization-manuals

Click **HP-UX Containers (SRP) Software**.

B.2.2 HP-UX Encrypted Volume and File System (EVFS)

EVFS (Encrypted Volume and File System) is an application-transparent technology providing protection of data at rest.

With EVFS, critical files and data at rest (on disk) are stored in encrypted form on disk. EVFS safeguards against compromised use of and unauthorized access to data due to physical theft of storage devices. The data encryption is based on a secret-key cryptosystem and runs as an integrated kernel service transparent to the user.

With HP-UX EVFS, disks and volumes can be configured to be used in one of two modes - volume-level encryption (EVS) or file-level encryption (EFS).

For more information, see the HP-UX EVFS documentation:

http://www.hp.com/go/hpux-security-docs

Click **HP-UX Encrypted Volume and File System Software**.

B.2.3 HP-UX IPSec

HP-UX IPSec provides an infrastructure to allow secure communications (authentication, integrity, confidentiality) over IP-based networks between systems and devices that implement the IPsec protocol suite.

For more information, see the HP-UX IPSec documentation:

http://www.hp.com/go/hpux-security-docs

Click **HP-UX IPSec Software**.

B.2.4 HP-UX OpenSSL

HP-UX 11i operating systems implement the Secure Sockets Layer (SSL V2 and V3) and Transport Layer Security (TLS V1) protocols using the OpenSSL Toolkit developed by the OpenSSL Project (http://www.openssl.org/). Secure Sockets Layer (SSL) is the open standard security protocol for the secure transfer of sensitive information over the Internet. SSL provides three basic security services: privacy through encryption, server authentication, and message integrity. Client authentication is available as an optional function.

Protecting communication links to HP-UX applications over a TCP/IP connection can be accomplished through the use of SSL. The OpenSSL APIs establish private, authenticated, and reliable communications links between applications.

For more information, see the HP-UX OpenSSL documentation:

http://www.hp.com/go/hpux-security-docs

Click **HP-UX OpenSSL Software**.

B.2.5 HP-UX Secure Shell

HP-UX Secure Shell uses hashing to ensure data integrity and provides secure tunneling features, port forwarding, and an SSH agent to maintain private keys on the client.

HP-UX Secure Shell enables you to securely log into another system over a network, to execute commands on a remote system, and to move files from one system to another. HP-UX Secure Shell provides a set of commands that replace insecure commands such as `rlogin`, `rsh`, `rcp`, `ftp`, and `telnet`.

HP-UX Secure Shell also protects a network from the following security hazards:

IP Spoofing	A technique used to gain unauthorized access to computers. An intruder sends messages to a computer with an IP address indicating that the message is coming from a trusted host.
Eavesdropping	Searching a system for passwords, credit card numbers, or business secrets.
Hijacking	A technique used to take over network communication in such a way that the attacker can inspect and modify data transmitted between the communicating parties.

For more information, see the HP-UX Secure Shell documentation:

http://www.hp.com/go/hpux-security-docs

Click **HP-UX 11i Secure Shell Software**.

B.2.6 HP-UX Trusted Computing Services

HP-UX Trusted Computing Services (TCS) provides software support for the Trusted Platform Module (TPM) option currently available on certain HP blade servers, the BL860C and BL870C being two examples. Each TPM chip contains a unique, hidden RSA private key and algorithms for applying the key to standard cryptographic operations. By cryptographic wrapping, private keys can be rendered usable only on a specific platform with a specific embedded TPM. This is useful for ensuring against unauthorized use of private keys on platforms other than those intended by the key owners. A TCS-generated key is effectively restricted for use on a single platform.

The TCS package provides an extensive set of library functions for application development. These library functions have been specified by the Trusted Computing Group for implementation on a wide range of platform architectures. The TCS package also includes commands for generating and maintaining TCS keys, and for bulk encryption of user data. You can find more information on TPM and Trusted Computing at: https://www.trustedcomputinggroup.org/home.

With TCS installed, TPM protection of private keys becomes available to a number of applications:

- HP-UX Encrypted Volume File System (EVFS) volumes can be configured to use TCS keys. With TCS, these volumes can only be decrypted on a specific server with the correct TPM chip. Procedures are provided for encrypted volume backup and configuration of ServiceGuard clustering when TCS keys are employed.

- HP-UX SecureShell now contains support for utilization of TCS keys for servers establishing encrypted sessions with remote clients. This prevents a SecureShell server from being easily transferred to another platform.
- With HP-UX OpenSSL, TCS key protection can be easily integrated into applications that rely on OpenSSL for cryptographic operations. The Stunnel product available with Internet Express provides a solid example of how TCS keys can be integrated through OpenSSL. An application server employing Stunnel to establish encrypted sessions can utilize TCS keys through Stunnel.

For more information, see the HP-UX TCS documentation:

http://www.hp.com/go/hpux-security-docs

Click **HP-UX Trusted Computing Services (TCS) Software**.

B.2.7 HP-UX Whitelisting

HP-UX Whitelisting (WLI) protects the system from unexpected downtime and denial-of-service by preventing inadvertent or illegitimate changes to the critical system files. It also protects files from unauthorized access by granting permissions only to the authorized applications, irrespective of the user (`uid`) executing the application. WLI is a cryptographic key-based product. Whitelisting security features are based on RSA key ownership and encryption technology. The authorization is provided by policies along with the traditional Discretionary Access Control(DAC). WLI security features are imposed through RSA signatures and enforced through signature verification. Therefore, regular files and directories may be protected from access by any user including super user.

For more information, see the HP-UX Whitelisting documentation:

http://www.hp.com/go/hpux-security-docs

Click **HP-UX Whitelisting**.

B.3 Protecting Identity

In addition to the security products that are discussed in *Part III Protecting Identity*, the following security products offer additional identity protection.

B.3.1 HP-UX AAA Server (RADIUS)

The HP-UX AAA Server utilizes the industry standard Remote Authentication Dial-In User Service (RADIUS) protocol and Extensible Authentication Protocol (EAP) to provide standards-based user authentication, authorization, and accounting services to network devices and software applications.

The HP-UX AAA Server can be utilized for securing wired and wireless LAN access, provide authentication and accounting for Virtual Private Network (VPN) gateways, firewalls and other network devices, and to enhance the security of RADIUS-enabled software applications in Enterprise and Service Provider environments.

For more information, see the HP-UX AAA Server documentation:

http://www.hp.com/go/hpux-security-docs

Click **HP-UX AAA Server (RADIUS) Software**.

B.3.2 HP-UX Directory Server

A global directory service, HP-UX Directory Server (HPDS) provides an industry-standard, centralized directory service on which to build your intranet or extranet. Your HP-UX servers and other directory-enabled applications use the directory server as a common, network-accessible location for storing shared data such as user and group identification, server identification, and access control information. In addition, you can extend the HP-UX Directory Server to support your entire enterprise with a global directory service that enables centralized management of all enterprise resource information.

HP-UX Directory Server includes enterprise-class features, including multi-master replication, encryption, authentication and access control, remote administration, on-line backup, as well as numerous other features.

For more information, see the HP-UX Directory Server documentation:

http://www.hp.com/go/hpux-security-docs

Click **HP-UX Directory Server**.

B.3.3 HP-UX LDAP-UX Integration

With an LDAP-enabled directory server, LDAP-UX Integration provides your HP-UX system centralized user, group, and system management, along with centralized authentication and access control. LDAP-UX supports standard LDAP directory servers as well as Windows Active Directory, for which HP-UX can use the same management groups and policies as in a Windows domain. In addition, users from multiple domains can authenticate to HP-UX. To simplify authentication and access control, LDAP-UX can defer to centralized password and account policies as well as define highly customizable access control policies for HP-UX services.

LDAP-UX includes numerous integration features: centralized configuration, flexible group management that includes support for standard LDAP groups or dynamic groups (role-based membership), command-line and GUI-based (through HP SMH) user and group management, host and ssh key management, off-line mode, and more.

For more information, see the HP-UX LDAP-UX Integration Software documentation:

http://www.hp.com/go/hpux-security-docs

Click **HP-UX LDAP-UX Integration Software**.

Glossary

3DES	Triple Data Encryption Standard. A symmetric key block encryption algorithm that encrypts data three times, using a different 56-bit key each time (168 bits used for keys). 3DES is suitable for bulk data encryption.
AAA server	Authentication, Authorization, and Accounting server. An AAA server provides authentication, authorization, and accounting services of user network access at the entry points to a network. HP-UX provides AAA servers based on the RADIUS protocol and Diameter Base protocol.
ACL	Access Control List. A list or database that defines what resources users or other principals can access, and the type of access allowed.
AES	Advanced Encryption Standard. A symmetric key block encryption algorithm. HP-UX IPSec supports AES with a 128-bit key. AES is suitable for bulk data encryption.
AH	Authentication Header. The AH provides data integrity, system-level authentication and can provide antireplay protection. AH is part of the IPsec protocol suite.
asymmetric key cryptography	*See* public key cryptography.
auditing	The selective recording of events for the analysis and detection of security breaches. The HP-UX auditing system provides a mechanism to audit users and processes.
authentication	The process of verifying the identity of a subject (a user, host, device or other entity in a computer network). Authentication is often a prerequisite to allowing access to resources in a system. Alternatively, the process of verifying the integrity of data, or the identity of the party that sent data.
Authentication Header	*See* AH.
authorization	The process of evaluating access control information and determining if a subject (a user, host, device, or other entity in a computer network) is allowed to perform an operation on a particular resource, or object. Authorization is typically performed after a subject's identity is authenticated. In the context of RBAC, authorization specifically refers to the pairing of an operation with an object, and is also referred to as **permission**. See RBAC.
Bastille	HP-UX Bastille is a system hardening and reporting program that enhances the security of the HP-UX operating system by consolidating essential hardening and lock-down checklists from industry and government security organizations, and making them accessible to administrators in an easy to use package.
bastion host	A computer system that protects an internal network from intruders. See also firewall and hardened system.
buffer overflow attack	A method to attack a system by causing process errors, or by causing a process to execute malicious code. This is typically achieved by overflowing an input buffer in the stack. This causes a memory violation or other error that causes the process to terminate, or causes the process to execute malicious code. See also stack buffer overflow attack.
CA	Certificate Authority. A trusted third-party that authenticates users and issues certificates. In addition to establishing trust in the binding between a user's public key and other security-related information in a certificate, the CA digitally signs the certificate information using its private key.

certificate	A security certificate associates (or binds) a public key with a principal—a particular person, system, device, or other entity. The security certificate is issued by an entity, in whom users have put their trust, called a Certificate Authority (CA), which guarantees or confirms the identity of the holder (person, device, or other entity) of the corresponding private key. The CA digitally signs the certificate with the CA's private key, so the certificate can be verified using the CA's public key.The most commonly used format for public-key certificates is the International Organization for Standardization (ISO) X.509 standard, Version 3.
Certificate Authority	*See* CA.
Certificate Revocation List	*See* CRL.
challenge-response authentication	A form of authentication where the authenticator sends a random value, the challenge, to the user or principal being authenticated. The user sends back a response based on the challenge value and a shared secret value previously established with the authenticator, such as an MD5 hash value.
	Unlike a regular password exchange, the challenge-response dialog varies, so an intruder cannot replay the user's response to gain authentication.
chroot jail	A method restricting the files and directories accessible by a process and users of that process. The process starts in a specified base directory (the root), and cannot access any directories or files above the root directory.
compartments	A method of isolating various components of the system from one another. When configured properly, components are an effective method to safeguard the HP-UX system and the data that resides upon it.
containment	A mechanism or set of mechanisms to restrict the access rights of processes.
	In the context of RBAC, containment is a combination of mandatory access control and fine-grained privileges. See RBAC.
CRL	Certificate Revocation List. Certificates are issued with a specific lifetime, defined by a start date/time and an expiration date/time. However, situations can arise, such as a compromised key value, that necessitate the revocation of the certificate. In this case, the certificate authority can revoke the certificate. This is accomplished by including the certificate's serial number on a CRL updated and published on a regular basis by the CA and made available to certificate users. See CA.
cryptography	The process of encoding normal data (or cleartext) data so it can only be decoded by holders of specific information.
Data Encryption Standard	See DES.
denial of service attack	An attack where a system is prevented from responding to network packets so the system cannot service requests. Denial of service attacks may be implemented by flooding a vulnerable system with false requests that consume a large number of resources. Denial of service attacks are often used with host spoofing to keep the spoofed host (the host with the IP address the spoofer is assuming) from participating in the exchange between the spoofer and the system the spoofer is trying to access.
DES	Data Encryption Standard. Uses a 56-bit key for symmetric key block encryption. DES is suitable for bulk data encryption.
	DES has been cracked (data encoded using DES has been decoded by a third party).

Diameter Base	A protocol that provides authentication, authorization, and accounting (AAA) services based on the RADIUS protocol. The Diameter protocol provides the same functionality as RADIUS, with improved reliability, security and infrastructure. See also RADIUS.
Diffie-Hellman	A public-key method to generate a symmetric key where two parties can publicly exchange values and generate the same symmetric key. Start with prime `p` and generator `g`, which may be publicly known (typically these numbers are from a well-known Diffie-Hellman Group). Each party selects a private value (`a` and `b`) and generates a public value (`g**a mod p`) and (`g**b mod p`). They exchange the public values. Each party then uses its private value and the other party's public value to generate the same symmetric key, `(g**a)**b mod p` and `(g**b)**a mod p`, which both evaluate to `g**(a*b) mod p` for future communication. The Diffie-Hellman method must be combined with authentication to prevent man-in-the-middle or third-party attacks (spoofing) attacks. For example, Diffie-Hellman may be used with certificate or preshared key authentication.
Digital Signature	Digital signatures are a variation of keyed hash algorithms that use public/private key pairs. The sender uses its private key and the data as input to create a Digital Signature value.
EAP	Extensible Authentication Protocol. A protocol that provides a framework for using multiple authentication methods and protocols, including passwords, Kerberos, and challenge-response protocols.
Encapsulating Security Payload	*See* ESP.
encryption	The process of converting data from a readable format to nonreadable format for privacy. Encryption functions usually take data and a cryptographic key (value or bit sequence) as input.
ESP	Encapsulating Security Payload. This is part of the IPsec protocol suite. The ESP provides confidentiality (encryption) and an antireplay service. It should be used with authentication, either with the optional ESP authentication field (authenticated ESP) or nested in an authentication header message. Authenticated ESP also provides data origin authentication and connectionless integrity. When used in tunnel mode, ESP also provides limited traffic flow confidentiality.
event	An action, such as creating a file, opening a file, or logging in to the system.
Extensible Authentication Protocol	*See* EAP.
filter	A mechanism for screening unwanted objects, or the parameters that specify the objects allowed or denied access. Typically, a filter is used to screen unwanted network packets (a packet filter).
fine-grained privilege	A permission to perform a specific, low-level operation (for example, permission to execute a specific system call).
firewall	One or more devices or computer systems used as a barrier to protect a network against unwanted users or harmful, intrusive applications. See also bastion host and hardened system.
hardened system	A computer system with minimal operating system features, users, and applications that is used as a barrier to protect a network against unwanted users or harmful, intrusive applications. Also referred to as a bastion host.
HMAC	Hashed Message Authentication Code. See also MAC.
IKE	The Internet Key Exchange (IKE) protocol is part of the IPsec protocol suite. IKE is used before the IPsec ESP or AH protocol exchanges to determine which encryption and/or authentication services

	will be used. IKE also manages the distribution and update of the symmetric (shared) encryption keys used by ESP and AH. See also ESP and AH.
IPSec policy	IPSec policies specify the rules according to which data is transferred securely. IPSec policies generally contain packet filter information and an action. The packet filter is used to select a policy for a packet and the action is applied to the packets using the policy
Kerberos	A network authentication protocol designed to provide strong authentication for client or server applications. Kerberos allows users to authenticate themselves without transmitting unencrypted passwords over the network.
LDAP (Lightweight Directory Access Protocol)	The LDAP protocol provides network directory access. LDAP uses a directory structure similar to the OSI X.500 directory service, but stores data as strings and uses the TCP/IP network stack instead of the OSI network stack.
MAC	A message authentication code (MAC) is an authentication tag, also called a checksum, derived by application of an authentication algorithm, together with a secret key, to a message. MACs are computed and verified with the same key so they can only be verified by the intended receiver, unlike digital signatures. Hash function-based MACs (HMACS) use a key or keys in conjunction with a hash function to produce a checksum that is appended to the message. An example is the keyed-MD5 method of message authentication. MACs can also be derived from block ciphers. The data is encrypted in message blocks using DES CBC and the final block in the ciphertext is used as the checksum. The DES-CBC MAC is a widely used US and international standard.
man-in-the-middle attack	*See* third-party-attack.
manual keys	Manually configured cryptographic keys for IPSec. An alternative to using the Internet Key Exchange (IKE) protocol to generate cryptographic keys and other information for IPSec Security Associations (SAs).
MD5	Message Digest-5. Authentication algorithm developed by RSA. MD5 generates a 128-bit message digest using a 128-bit key. IPSec truncates the message digest to 96 bits.
NAT	Network Address Translation. A method to allow multiple systems in an internal, private network share one public internet IP address. A NAT gateway replaces (translates) internal IP addresses and ports to its public IP address when forwarding packets from the internal network to the public internet and performs the reverse translation for the return path.
object	A system or network resource such as a system, file, printer, terminal, database record. In the context of authorization, authorization is granted for a subject's operation on an object.
operation	A specific mode of access to one or more objects. For example, writing to a file. In the context of authorization, authorization is granted for a subject's operation on an object.
out-of-band key exchange	A key exchange using a secure communication channel that is outside of normal computer communication channels, such as a face-to-face meeting or telephone call.
packet filter	A filter used to select or restrict network packets. Packet filters specify network packet characteristics. Packet filters typically specify source and destination IP addresses, upper-layer protocols (such as TCP or UDP), and TCP or UDP port numbers. Packet filters may also define other packet fields, such as IPv6 header types, upper-layer message types (for example, ICMP message types), and TCP connection states.

PAM Pluggable Authentication Module. An authentication framework that allows system administrators to configure services for authentication, account management, session management, and password management for HP-UX utilities, such as the system login utility.

Perfect Forward Secrecy (PFS) With Perfect Forward Secrecy, the exposure of one key permits access only to data protected by that key.

Pluggable Authentication Module

See PAM.

preshared key A cryptographic value agreed upon by two systems for encryption or authentication. The key is exchanged prior to computer data communication, typically using an out-of-band key exchange (such as a verbal, face-to-face exchange). See also shared key cryptography.

principal A person, system, device or other entity.

private key cryptography *See* shared key cryptography.

privilege A permission to perform an action on a computer system.

public key cryptography A cryptographic method using two mathematically related keys (for example, k1 and k2) such that data encrypted with k1 can be decrypted only using k2. In addition, most algorithms provide assurance that only the holder of k1 can correctly encrypt data that can be decrypted by k2.

One key must be private (known only to the owner), but the second key can be widely known (public), which makes key distribution easy to manage. Public key encryption is computationally expensive, so it is impractical for bulk data encryption. Instead, public key cryptography is usually used to authenticate data.

Also referred to as asymmetric key cryptography (the two keys are not the same) or public-private key cryptography.

public-private key cryptography

See private key cryptography.

RADIUS The Remote Authentication Dial-In User Service (RADIUS) protocol is widely used and implemented to manage access to network services. It defines a standard for information exchange between a network access device and an authentication, authorization, and accounting (AAA) server for performing authentication, authorization, and accounting operations. A RADIUS AAA server can manage user profiles for authentication (verifying user name and password), configuration information that specifies the type of service to deliver, and policies to enforce that may restrict user access.

The RADIUS protocol provides only the framework for the authentication exchange and can be used with numerous authentication methods.

RBAC Role-Based Access Control. An HP-UX mechanism to provide fine-grained access to system resources, commands, and system calls. Users are assigned to roles and users are granted privileges for access according to roles.

role A job function, within the context of an organization, with associated semantics regarding the authority and responsibility given to users assigned to the role.

Role-Based Access Control *See* RBAC.

RSA Rivest, Shamir, and Adelman. Public-private key cryptosystem that can be used for privacy (encryption) and authentication (signatures). For encryption, system A can send data encrypted with system B's public key. Only system B's private key can decrypt the data. For authentication, system

	A sends data with a digital signature, a digest or hash encrypted with system A's private key. To verify the signature, system B uses system A's public key to decrypt the signature and compare the decrypted hash or digest to the digest or hash that it computes for the message.
SASL	Simple Authentication and Security Layer. A protocol used to add authentication services to connection-based network applications. The SASL API provides a flexible framework that allows programmers to use a common interface to access multiple authentication services.
secure shell	*See* SSH.
Secure Sockets Layer	*See* SSL.
Security Certificate	*See* certificate.
SHA1	Secure Hash Algorithm-1. An authentication algorithm that generates a 160-bit message digest using a 160-bit key.
shadow password	A structure to provide additional security for user passwords. The shadow password structure (spwd) contains encrypted user passwords and other information used with the passwd structure. The shadow password structure is stored in a file that is usually readable only by privileged users.
shared key cryptography	A cryptographic method where two parties use the same key (the two parties share the same key) for encrypting or authenticating data. To provide data privacy or authentication, only the two parties can know the key value (the key must be private). Shared key cryptography is more efficient than public-private key cryptography for encrypting data, so it is often used for bulk data encryption. However, distributing or establishing the shared key requires an out-of-band key exchange (such as a face-to-face verbal exchange), Diffie-Hellman exchange, or other mechanism. Also referred to as private key cryptography or symmetric key cryptography.
SSH	Secure Shell. A set of network services that provides secure replacements for remote login, file transfer, and remote command execution. SSH also provides secure tunneling features, port forwarding, and an SSH agent to maintain private keys on the client.
SSL	Secure Sockets Layer. A protocol used to encrypt network data. The SSL protocol is above TCP in the data stack. SSL uses public/private keys to authenticate principals and exchange a private (shared) key. SSL then uses the private key to encrypt data.
stack buffer overflow attack	A method to attack a system by causing a process to execute malicious code. This is typically achieved by overflowing an input buffer in the stack to insert malicious code and then modifying the stack pointer to execute the malicious code. See also buffer overflow attack.
stateful packet filter	A type of packet filtering that uses upper-layer protocol fields and state information, such as TCP connection states.
subject	A user, host, device or other entity in a computer network. In the context of authorization, the originator of an operation on an object requiring an authorization decision.
symmetric key cryptography	*See* shared key cryptography.
third-party attack	In a third-party attack, the attacker intercepts packets between two attacked parties, A and B. A and B assume they are exchanging messages with each other, but are exchanging messages with the third party. The attacker assumes the identity of A to exchange messages with B, and assumes the identity of A to exchange messages with B. Also referred to as man-in-the-middle attack.
transitive trust relationship	Extending a trust relationship through other trusted entities. If A and B both trust C, A and B can trust each other using a transitive trust relationship through C. In a hierarchical structure, A and B can establish a transitive trust relationship if they can establish a chain-of-trust to a common root.

VPN	Virtual Private Network. A private network within a public network, such as the global Internet. A VPN is virtual because it uses tunnels to effectively create a separate logical network within a physical network. A VPN is private because outside users cannot see or modify the data being transmitted. VPNs that use host identity authentication also provide protection against IP address spoofing.

Index

Symbols
/dev special device file
  security considerations for, 103
/etc/d_passwd file
  controlling access using, 56
/etc/default/security, 25
/etc/dialups file
  controlling access using, 56
/etc/ftpd/ftpusers file
  changing access with, 69
/etc/group file, 194
/etc/inetd.sec file, 72
/etc/pam.conf file, 35
  configuring systemwide with, 37
/etc/pam_user.conf file, 35
/etc/passwd file, 191, 192, 193, 194
  application user accounts, 30
  changing, 42
  example of pseudo-account in, 45
  format of, 43
  recovering, 27
  restricted account, 30
/etc/rbac/aud_filter, 188
/etc/rbac/cmd_priv, 160
  entries, 162
/etc/security.dsc file, 47
/etc/shadow shadow password file, 43
/sbin/rc2.d/S760auditing, 192
/tcb/files/auth/ protected password database, 192, 193
/tcb/files/auth/*/*, 191, 194, 196, 197
/tcb/files/ttys, 196
/tmp, 198
/var.adm/userdb file, 48, 63
/var/adm/inetd.sec file
  configuring, 72

A
access
  device-based access, 196
  password, 195
  terminal control, 195
  time-based access, 195, 196
access control list
  See ACL, 91
Access Control Policy Switch, 147
  customizing, 166
  interfaces, 147
ACL
  and NFS, 103
  comparison of JFS and HFS, 102
  default JFS entries, 99
  example of changing a minimal JFS, 98
  setting, 91
  setting HFS, 91
  setting JFS, 95
  trusted system backup/recovery, 197
administrative domain
  managing, 74
AES (Advanced Encryption Standard), 205
AH (Authentication Header)
  definition, 205
anonymous FTP
  securing, 69
at command, 192
audisp command
  viewing audit log output with, 186
audit event, 177
  type, 179
audit flag, 195
audit ID (aid), 192, 194, 195
audit log file, 179
  overwriting existing, 181
  streamlining data in, 187
  viewing, 186
auditing
  basic profile, 178
  commands, 172
  enabling, 172
  turning on after recovery, 27
  users, 171
authadm, 157
  examples, 158
  syntax, 157
authentication, 192
  during login, 31
  PAM login example, 39
  used by SSH, 80
  using boot, 25
  using PAM, 34
Authorization Number, 193
authorizations, 144
  configuring, 157
  object, 144
  operation, 144
auxiliary audit log file, 180

B
backup
  security guidelines for, 26
  trusted system, 191, 197
backup media
  security of, 197
Bastille (*see* HP-UX Bastille)
batch, 192

boot authentication
  using, 25
boot processs
  gaining, 24
booting
  preventing security breaches during booting, 23
btmp file
  tracking failed logins with, 33

C
CA (certificate authority)
  defined, 205
CDE Lock Manager
  configuring, 55
CDE Login Manager
  logging in with, 32
Certificate Revocation List (CRL), 206
chfn, 194
chmod command
  changing file access permissions with, 88
  effect on class entries, 97
chown, 27, 194, 198
chroot jail, 84
chsh, 194
cmdprivadm, 158
  examples, 159
  syntax, 158
command
  login, 193
  swlist, 192
compartments, 109
  activating, 124
  creating rules, 114
  file system rules, 116
  IPC rules, 117
  modifying rules, 114
  network interface rules, 122
  network rules, 119
  planning a structure, 111
  privilege limitation rules, 123
  troubleshooting, 126, 140
crontab, 192

D
DES (Data Encryption Standard), 206, 207
device assignment database
  trusted system, 197
device-based access control, 196
Diffie-Hellman, 207
  group, 207
directory access
  securing, 89
disk partition
  security considerations for, 104
domain
  managing an administrative, 74

E
encrypted password field, 194
encryption
  definition, 207
ESP (Encapsulating Security Payload)
  definition, 207
/etc/ftpd/ftpusers, 69
/etc/inetd.sec, 72
/etc/passwd, 27
expiration time
  password aging, 195

F
fbackup command, 26
  trusted backup, 197
file
  /etc/group, 194
  /etc/passwd, 191, 192, 193, 194
file corruption
  locating and correcting using fsck command, 90
file ownership
  setting, 89
file security
  considerations for /dev special files, 103
  controlling file access, 87
  controlling on a network, 106
  protecting disk partitions and logical volumes, 104
  protecting files related to user accounts, 90
  protecting NFS-mounted files, 107
file set
  SecurityMon, 192
file system
  security guidelines for mounting and unmounting, 104
fileaccess
  setting access permissions, 88
filter
  definition, 207
fine-grained privileges, 131
  configuring, 160
frecover command, 26
  trusted recovery, 197
fsck command
  correcting file corruption with, 90
FTP
  securing, 68
  securing anonymous, 69
ftpd server, 69
function
  getdvagent, 197
  getprdfent, 197
  getprpwent, 197
  getprtcent, 197
  getpwent, 197
  getspwent, 197
  putprpwnam, 197
  putpwent, 197

putspwent, 197

G
getacl command
viewing ACLs with, 97
getdvagent function, 197
getfilexsec command, 113, 132
getprdfent function, 197
getprocxsec command, 113, 132
getprpwent function, 197
getprtcent function, 197
getpwent function, 197
getspwent function, 197
group account
managing, 31
group ID (gid), 194
GSS-API
SSH, 80
guest account
monitoring, 30

H
HFS, 91
HFS ACL
and NFS, 103
commands and calls that work with, 93
compared with JFS ACL, 102
setting, 91
High Performance File System
See HFS, 91
history
password, 196
host-based authentication
and public key based authentication, 81
used by SSH, 81
HP-UX AAA Server (RADIUS), 203
HP-UX Bastille, 15, 25, 199
defined, 205
HP-UX Containers SRP, 200
HP-UX Directory Server, 204
HP-UX EVFS, 201
HP-UX HIDS, 199
HP-UX installation
installing security patches, 26
postinstallation security tips, 26
preventing security breaches during booting, 23
security considerations, 23
setting install-time security options, 25
HP-UX IPFilter, 200
HP-UX IPSec, 201
HP-UX LDAP-UX, 204
HP-UX RBAC
architecture, 149
auditing, 188
commands, 148
wrapping, 154
components, 146
configuration files, 147
configuring Compartments, 162
default user, 156
manpages, 148
operation, 150
troubleshooting, 168
HP-UX Secure Shell, 202
HP-UX Security Patches, 200
HP-UX TCS, 202

I
IKE (Internet Key Exchange)
protocol, 207
inetd daemon
overview of, 70
securing, 71
TCP wrappers and, 72
Install-Time Security, 25
installing HP-UX
installing security patches after, 26
postinstallation security tips, 26
preventing security breaches during booting, 23
security considerations, 23
setting install-time security options, 25
installing security patches
using Software Assistant, 26
Internet daemon
See inetd daemon, 70
Internet Services, 67
overview of, 67
IPSec policy
definition, 208

J
JFS, 91
JFS ACL
and NFS, 103
changing with setacl command, 100
compared with HFS ACL, 102
example of changing a minimal, 98
setting, 95
using default entries, 99
Journaled File System
See JFS, 91

L
last command
examples of using, 33
LDAP directory server
securing passwords stored in, 46
lifetime
password aging, 195
log file
audit, 179
logical volume

security considerations for, 104
Logical Volume Manager
See LVM, 104
login banners
securing, 57
login command, 32, 193
login process
explanation of, 32
login tracking file, 33
lost+found directory, 27, 198
LVM, 104

M
MAC, 208
managing file access, 87
managing passwords, 41
minimum time
password aging, 195
mobile connection
securing, 56
modem access
security guidelines for managing, 55
mounting a file system securely, 104

N
network administration, 75
controlling file security, 106
managing an administrative domain, 74
network control file
checking permissions on, 106
verifying permissions on, 75
NFS, 106
and ACLs, 103
protecting NFS-mounted files, 107
securing the client, 107
securing the server, 107
NIS
securing passwords stored in, 46

O
operations
guidelines for creating, 153

P
PAM
authenticating users with, 34
configuring systemwide, 37
overview of, 35
PAM authentication
login example, 39
PAM library, 36
PAM service module, 35
parameter
PASSWORD_HISTORY_DEPTH, 196
passwd command, 194
examples of, 42
password, 195
aging, 192, 194, 195
expiration time, 195
lifetime, 195
minimum time, 195
authentication
used by SSH, 81
criteria of a good, 42
database, 191, 192, 194
/tcb/files/auth/, 192, 193
encrypted field, 194
encryption, 193
entry
manipulating, 197
file
fields, 193
protected password database, 191, 192, 194
generation, 195
history, 196
integrity, 193
management, 41
reuse, 196
security, 192
shadow, 43
types of, 195
PASSWORD_HISTORY_DEPTH parameter, 196
patch installation
using Software Assistant, 26
Perfect Forward Secrecy (PFS)
defined, 209
permissions
checking network control file, 106
verifying for network control files, 75
power failure, 27, 198
file loss, 27
preshared keys
definition, 209
primary audit log file, 180
privedit, 165
options, 166
syntax, 166
privrun, 163
-p, 161
examples, 164
operation, 150
options, 163
syntax, 163
protected password database
/tcb/files/auth/, 192, 193
prpwd, 194
pseudo-account
example of, 45
public key based authentication
and host-based authentication, 81
used by SSH, 81
putprpwnam function, 197

putpwent function, 197
putspwent function, 197

R
random number generator, 83
recovery
  security guidelines for, 26
remote access
  security guidelines for managing, 55
Remote Access Services, 67
  overview of, 67
remote procedure call
  See RPC, 73
remote sessions
  securing using SSH, 76
reuse
  password, 196
roleadm, 155
  examples, 156
  syntax, 155
roles
  configuring, 155
  default, 156
  groups, 157
  guidelines for creating, 152
root
  drawbacks of, 143
root access
  gaining, 24
  monitoring, 58
  reviewing, 59
  using Restricted SMH Builder for limited, 58
root account
  protecting, 58
RPC
  and TCP wrappers, 73
RSA cryptosystem, 209
rsh command
  limiting system access with, 30
run level
  changing, 53
  controlling access with, 53

S
screen lock
  configuring, 54
Sec00Tools security level, 25
Sec20MngDMZ security level, 25
Sec30DMZ security level, 25
Secure Shell
  see SSH, 76
securing remote sessions, 76
security attribute
  defining, 46, 62
security level
  choosing during installation, 25
security patch
  installing, 26
SecurityMon file set, 192
selection and generation, 195
self-auditing program, 187
set group ID program
  See setgid programs, 50
set user ID program
  See setuid programs, 50
setacl command
  changing ACLs with, 97
  changing JFS ACLs with, 100
setfilexsec command, 113, 132
setgid programs, 27, 198
  managing, 50
setuid programs, 27, 198
  managing, 50
shadow password, 43
single-user mode
  booting into, 24
SIS, 73
Software Assistant
  using, 26
spoofing
  defined, 70
  protecting against using TCP wrappers, 72
SSH, 46
  associated technologies, 83
  authentication, 76, 80
  encryption, 76
  features, 76
  GSS-API, 80
  HP-UX system, 82
  password authentication, 81
  port forwarding, 77
  privileged mode execution, 79
  public key based authentication, 81
  running, 78
  running scp client, 79
  running sftp client, 79
  running ssh client, 78
  securing remote sessions, 76
  software components, 77
  strong random number generator, 83
  support for TCP wrappers, 83
SSH-1 protocol, 82
SSH-2 protocol, 82
stack buffer overflow protection, 52
sticky bit
  setting, 89
strong random number generator, 83
superuser access
  monitoring, 58
  protecting, 58
  reviewing, 59
  using Restricted SMH Builder for limited, 58
swlist command, 192

system access
    security guidelines for remote, 55
system administration
    auditing guidelines, 176
    auditing users, 171
    authenticating users during login, 31
    authenticating users using PAM, 34
    backup guidelines, 26
    controlling file security on a network, 106
    defining security attributes, 46, 62
    installing HP-UX securely, 23
    installing security patches, 26
    managing an administrative domain, 74
    managing passwords, 41
    managing remote access, 55
    managing setuid and setgid programs, 50
    managing user access, 29
    mounting and unmounting a file system securely, 104
    preventing stack buffer overflow attacks, 52
    protecting root acess, 58
    protecting unattended workstations and terminals, 53
    securing FTP, 68
    securing inetd, 71
    securing Internet Services, 67
    securing login banners, 57
    securing the HP-UX file system, 87
    security breaches , 23
    setting install-time security options, 25
    using boot authentication to prevent unauthorized access, 25
system run level
    changing, 53
    controlling access with, 53
system security
    defining security attributes, 46, 62

T
TCP wrappers
    and SSH, 83
    protecting against spoofing with, 72
telephone
    securing, 56
temporary account
    disabling, 30
terminal
    configuring screen lock for, 54
    protecting unattended, 53
terminal access, 195
terminal control database
    trusted system, 197
terminal device file
    protecting, 54
time-based access control, 196
TMOUT variable
    configuring, 54
trusted, 195
trusted password, 195
trusted password database, 197
trusted system
    converting from, 192
    converting to, 191
    databases, 197

U
umask command
    changing default file permissions with, 89
unique user name
    importance of, 32
unmounting a file system securely, 104
user access
    managing, 29
user account
    restricted, 30
user authentication
    during login, 31
    PAM login example, 39
    using PAM, 34
user ID (uid), 194, 195
user name
    creating unique, 32
user security
    managing, 29
userdbset command
    examples of defining user attributes with, 49

V
/var/adm/inetd.sec, 72
verifying permissions on network control files, 75

W
who command
    obtaining user login information with, 34
workstation
    protecting unattended, 53
wtmp file
    tracking successful logins with, 33
WU-FTPD, 69