HP-UX 11i Security Containment Administrator's Guide

HP-UX Servers and Workstations

HP-UX 11i v2

Manufacturing Part Number: 5991-1821
E0605

Printed in the US

Legal Notices

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

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

Warranty

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

U.S. Government License

Proprietary computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Trademark Notices

UNIX® is a registered trademark in the United States and other countries, licensed exclusively through The Open Group.

About This Document

1. HP-UX 11i Security Containment Introduction

2. Installation

3. HP-UX Role-Based Access Control

6. Standard Mode Security Extensions

About This Document

This document describes how to install, configure, and troubleshoot HP-UX 11i Security Containment on HP-UX 11i Version 2.

Intended Audience

This document is intended for system administrators responsible for installing, configuring, and managing HP-UX 11i Security Containment. Administrators are expected to have knowledge of HP-UX 11i v2 operating system concepts, commands, and configuration.

It is helpful to have knowledge of UNIX security concepts, commands, and protocols. Knowledge of HP-UX trusted mode systems is helpful, but not necessary.

This document is not a tutorial.

New and Changed Information in This Edition

This is a new document, published in parallel with the first release of HP-UX 11i Security Containment.

Publishing History

The document printing date and part number indicate the document's current edition. The printing date will change when a new edition is printed. Table 1 gives a history of printing dates for this document. Minor changes may be made at reprint without changing the printing date. The document part number will change when extensive changes are made.

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

The latest version of this document can be found online at http://www.docs.hp.com.

Table 1 Publishing History Details

Document Manufacturing Part Number	Operating Systems Supported	Supported Product Versions	Publication Date
599-1821	11i Version 2	HP-UX 11i Security Containment version B.11.23.01 HP-UX RBAC version B.11.23.01 HP-UX SMSE version B.11.23.01	May 2005

Document Organization

The *HP-UX 11i Security Containment Administrator's Guide* contains the following information about installing or configuring HP-UX 11i Security Containment:

Chapter 1	"HP-UX 11i Security Containment Introduction." Use this chapter to learn about the security containment features and how those features work together to secure your HP-UX 11i v2 system.
Chapter 2	"Installation." Use this chapter to plan and execute the installation of the full HP-UX 11i Security Containment product or individual security containment components.
Chapter 3	"HP-UX Role-Based Access Control." Use this chapter to learn how to configure and administer HP-UX RBAC.
Chapter 4	"Fine-Grained Privileges." Use this chapter to learn how to administer fine-grained privileges.
Chapter 5	"Compartments." Use this chapter to learn how to configure and administer compartments.
Chapter 6	"Standard Mode Security Extensions." Use this chapter to learn how to configure and administer the user database, per-user security attributes, and system auditing.

Typographic Conventions

This document uses the following conventions:

audit (5)	An HP-UX manpage. In this example, *audit* is the name and *5* is the section in the *HP-UX Reference*. On the Web and on the Instant Information CD, it may be a hot link to the manpage itself. From the HP-UX command line, you can enter "`man audit`" or "`man 5 audit`" to view the manpage. Refer to *man* (1).
Book Title	The title of a book. On the Web and on the Instant Information CD, 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.
Bold	Text that is strongly emphasized.
Bold	The defined use 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. If the contents are a list separated by |, you must choose one of the items.
`{ }`	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.

HP-UX Release Name and Release Identifier

Each HP-UX 11i release has an associated release name and release identifier. The *uname* (1) command with the `-r` option returns the release identifier. Table 2 lists the releases available for HP-UX 11i.

Table 2 HP-UX 11i Releases

Release Identifier	Release Name	Supported Processor Architecture
B.11.11	HP-UX 11i v1	PA-RISC architecture
B.11.20	HP-UX 11i v1.5	Intel® Itanium® architecture
B.11.22	HP-UX 11i v1.6	Intel® Itanium® architecture
B.11.23	HP-UX 11i v2	Intel® Itanium® architecture
B.11.23	HP-UX 11i v2 September 2004 and later	PA-RISC and Intel® Itanium® architecture

Related Information

You can find additional information about HP-UX 11i Security Containment at http://www.docs.hp.com, in the internet and security solutions collection under HP-UX 11i Security Containment.

Other documents in this collection include:

HP-UX 11i Security Containment Release Notes

HP-UX Standard Mode Security Enhancements Release Notes

HP-UX Role-Based Access Control Release Notes

HP Encourages Your Comments

HP encourages your comments concerning this document. We are truly committed to providing documentation that meets your needs.

Please send comments to netinfo_feedback@cup.hp.com.

Please include document title; manufacturing part number; and any comment, error found, or suggestion for improvement you have concerning this document. Also, please include what we did right so we can incorporate it into other documents.

1 HP-UX 11i Security Containment Introduction

This chapter contains overview information about the features of HP-UX 11i Security Containment. It addresses the following topics:

Conceptual Overview

HP-UX 11i Security Containment uses three core technologies: compartments, fine-grained privileges, and role-based access control. Together, these three components provide a highly secure operating environment without requiring existing applications to be modified. In addition, HP-UX 11i Security Containment makes several newly enhanced trusted mode security features available on standard mode HP-UX systems. These features are called HP-UX Standard Mode Security Extensions (HP-UX SMSE).

With HP-UX 11i Security Containment, the HP-UX 11i v2 operating system provides a highly secure, easy-to-maintain, and backwards-compatible environment for business applications. HP-UX 11i Security Containment implements several important security concepts. The following sections describe these concepts as implemented by security containment:

- Authorization
- Account Policy Management
- Privileges
- Isolation
- Auditing

Authorization

Authorization is the concept of limiting the actions a user is allowed to perform on a system, often based on the user's business needs. A traditional UNIX system offers only two levels of authorization:

regular user Limited access to system resources

superuser Unlimited access to system resources

HP-UX Role-Based Access Control (HP-UX RBAC) creates many different levels of authorization, based on roles. You can configure roles based on business need, for a user or group of users to perform specific actions on the system. Then you assign users to the roles you configured.

Account Policy Management

Account policy management is the concept of maintaining user and system security attributes used for authorization. Some user and system attributes include the time of day a user is allowed to log on, how long a user can remain inactive before being automatically logged out, and how long a user's password remains valid.

Account policy management is implemented using HP-UX Standard Mode Security Extensions features of HP-UX 11i Security Containment.

Privileges

Privileges are similar to authorization, except that instead of limiting the actions a user can perform on a system, privileges limit the actions a program can perform on a system. On a traditional UNIX system, a program can run as though owned by the invoking user or by the file owner (for example, a `setuid` program). Access to certain system resources require the program to be set to the superuser using the `setuid` command. This allows the program great latitude in reading and modifying system resources.

Privileges break up the latitude of the superuser into many different levels. The fine-grained privileges feature of HP-UX 11i Security Containment implements the concept of privileges.

Isolation

Compartments are a method of isolating components of a system from one another. Conceptually, processes belong to a compartment, and resources are associated with an access list that specifies how processes in different compartments can access them. 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.

When configured properly, they can be an effective method to safeguard your HP-UX system and the data that resides on it.

Auditing

Auditing is the concept of tracking significant events on a system. You can record and analyze security events to help detect attempted security breaches and to understand successful breaches so that you can prevent them in the future.

Prior to the release of HP-UX 11i Security containment, auditing was available only on trusted mode HP-UX systems. With HP-UX 11i Security Containment, you can use enhanced auditing on standard mode HP-UX 11i v2 systems. You can configure HP-UX RBAC to audit access control request to the audit system.

Defined Terms

The following terms are used throughout this manual.

HP-UX RBAC

HP-UX Role-Based Access Control. Refer to Chapter 3, “HP-UX Role-Based Access Control,” on page 33 for information about HP-UX RBAC.

HP-UX SMSE

HP-UX Standard Mode Security Extensions. This set of features includes the user database and standard mode auditing.

NOTE When you run `swlist`, the HP-UX SMSE product name appears as `TrustedMigration`.

Refer to Chapter 6, “Standard Mode Security Extensions,” on page 97 for information about HP-UX SMSE.

Trusted Mode

Trusted Mode is a legacy method of securing the HP-UX operating system. Refer to *Managing Systems and Workgroups: A Guide for HP-UX Systems Administrators* for HP-UX 11i v 2 for information about trusted mode.

Legacy applications

In this document, a legacy application is an application created without awareness of fine-grained privileges or compartments. All applications released before HP-UX 11i Security Containment are legacy applications.

Features and Benefits

HP-UX 11i Security Containment Version B.11.23.01 contains a number of features to help you secure your HP-UX standard mode system.

Features

HP-UX 11i Security Containment Version B.11.23.01 includes the following components:

- Compartments

  Compartments isolate unrelated resources on a system, to prevent catastrophic damage to the system if one compartment is penetrated.

  When configured in a compartment, an application has restricted access to resources (processes, binaries, data files, and communication channels used) outside its compartment. This restriction is enforced by the HP-UX kernel and cannot be overridden unless specifically configured to do so. If the application is compromised, it will not be able to damage other parts of the system because it is isolated by the compartment configuration.

- Fine-Grained Privileges

  Traditional UNIX operating systems grant “all or nothing” administrative privileges based on the effective `UID` of the process that is running. If the process is running with the effective `UID=0`, it is granted all privileges. With fine-grained privileges, processes are granted only the privileges needed for the task and, optionally, only for the time needed to complete the task. Applications that are privilege-aware can elevate their privilege to the required level for the operation and lower it after the operation completes.

- HP-UX Role-Based Access Control (HP-UX RBAC)

  Typical UNIX system administration commands must be run by a superuser (root user). Similar to kernel level system call access, access is usually “all or nothing” based on the user's effective UID. HP-UX Role-Based Access Control (HP-UX RBAC) enables you to group common or related tasks into a role. For example, a common role might be User and Group Administration. Once the role is created, users are assigned a role or set of roles that enables them to run the commands defined by those roles.

  When you implement HP-UX RBAC, you enable non-root users to perform tasks previously requiring root privileges, without granting those users complete root privileges.

  For more information about HP-UX RBAC, refer to the *HP-UX Role-Based Access Control B.11.23.02 Release Notes*.

- HP-UX Standard Mode Security Extensions

  In addition to the new Security Containment features, HP-UX 11i v2 has been enhanced to support the following security features, previously available only in trusted mode:

  — Audit

    The HP-UX auditing system records security-related events for analysis. Administrators use auditing to detect and analyze security breaches. Auditing is now available on standard mode HP-UX systems; it was previously available only on trusted mode systems.

  — User Database

    Previously, all Standard Mode 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.

You can use the user database to enforce the following security measures:

— Lock a user account after a specified number of authentication failures

— Display the last successful and unsuccessful login

— Maintain a password history

— Expire inactive user accounts

— Prevent users from logging in with a null password

— Restrict users to logging in only during specified time periods

Benefits

Using HP-UX 11i Security Containment to secure your system offers the following benefits:

- Integrated security

  You can use HP-UX Standard Mode Security Extensions in combination with the new security containment features to enhance the security of your HP-UX systems.

- Fewer users who need full superuser access to systems

  Using HP-UX RBAC, you can give users specific administrator-level privileges on a system without giving those users full superuser access. These users can perform only specific administrative tasks on the system, as defined by their roles. This provides strong internal system security.

- Isolation of system resources

  Using compartments, you can isolate applications and resources on a single system. Even if the security of one application is compromised, other resources on the system remain secure.

- Interoperable with existing HP-UX 11i security products

  You can integrate HP-UX 11i Security Containment with your existing HP-UX security solution. HP-UX 11i Security Containment works with all other HP-UX 11i v2 security products and features.

- No need to modify existing applications

  HP-UX 11i Security Containment can be configured to be transparent at the application layer. You do not need to modify your existing applications to use HP-UX 11i Security Containment.

- Interoperability with HP Serviceguard

  HP Serviceguard is comparable with the HP-UX 11i Security Containment default configuration. Because Serviceguard requires communication and control between many processes and nodes, be sure to follow all constraints described in this document if you change the default containment configuration.

  For more information about configuring HP-UX 11i Security Containment to ensure proper cluster operation for appropriate enforcement of security policies, refer to “Fine-Grained Privileges in HP Serviceguard Clusters” on page 73 and “Compartments in HP Serviceguard Clusters” on page 95.

2 Installation

This chapter contains the information you need to install and remove HP-UX 11i Security Containment, HP-UX Role-Based Access Control, and Standard Mode Security Extensions. This chapter addresses the following topics:

Prerequisites and System Requirements

You must meet the following system requirements to install HP-UX 11i Security Containment.

Hardware

HP 9000 systems

HP Integrity systems

Software

HP-UX 11i Version 2 September 2004 release or later

Disk Space

66 Mbytes

Installing HP-UX 11i Security Containment

The following procedures describe how to download and install the HP-UX 11i Security Containment product from the `SecurityExt` bundle. This bundle includes the following software:

- Compartments
- Fine-grained privileges
- HP-UX RBAC
- HP-UX SMSE
- Audit

Subsequent sections of this chapter describe how to install the HP-UX RBAC, HP-UX SMSE, and audit features separately, from different software bundles. Refer to "Installing HP-UX Role-Based Access Control" on page 25 and "Installing HP-UX SMSE" on page 27.

IMPORTANT The HP-UX 11i Security Containment feature includes HP-UX RBAC B.11.23.02 as one of its components. If you install the HP-UX 11i Security Containment feature on a system that has HP-UX RBAC B.11.23.02 on it as an independent software unit, you must reconfigure HP-UX RBAC before you can use it with the fine-grained privileges and compartments components of HP-UX 11i Security Containment. Use the following command to reconfigure HP-UX RBAC:

```
# swconfig -x reconfigure=true RBAC
```

To download the HP-UX 11i Security Containment bundle from Software Depot, follow these steps:

Step 1. Go to HP Software Depot at http://www.hp.com/go/softwaredepot.

Step 2. Search for HP-UX 11i Security Containment. Read the product information Web page for the latest updates and release information.

Step 3. Click `Receive for Free>>`.

Step 4. Choose the correct version of HP-UX 11i Security Containment for your system.

Step 5. Enter your registration information. Read and accept the Terms and Conditions and the Software License Agreement.

Step 6. Click `Download`. Save the HP-UX 11i Security Containment bundle, `SecurityExt`, as a local file on your system. For example:

```
/tmp/<security_containment_bundle>.depot
```

NOTE The name of the HP-UX 11i Security Containment bundle may change. Check Software Depot for the correct bundle name.

Step 7. Verify that the depot file is saved on your system by using the following command:

```
# swlist -d @ /tmp/<security_containment_bundle>.depot
```

To install HP-UX 11i Security Containment, follow these steps:

Step 1. Be sure your system meets all requirements, as described in “Prerequisites and System Requirements” on page 21.

Step 2. Download the HP-UX 11i Security Containment bundle from Software Depot as described in the previous procedure.

Step 3. Log in to your system as the root user.

Step 4. Install HP-UX 11i Security Containment by using the following command:

```
# swinstall -x autoreboot=true -s /tmp/<security_containment_bundle>.depot
SecurityExt
```

Step 5. Go on to “Verifying the HP-UX 11i Security Containment Installation” on page 24.

Verifying the HP-UX 11i Security Containment Installation

Verify the installation of HP-UX 11i Security Containment with the following steps:

Step 1. Run the swverify command to ensure that the bundle installed correctly:

```
# swverify SecurityExt
```

If the installation is successful, many files are displayed and a success message appears after the verification is complete.

Step 2. Run the swlist command to verify that all parts of HP-UX 11i Security Containment are configured correctly on your system:

```
# swlist -a state -l fileset SecurityExt
```

If the product is configured correctly, each fileset is displayed as `configured`.

Installing HP-UX Role-Based Access Control

The following procedure describes how to install only HP-UX RBAC from the HP-UX 11i Security Containment bundle. To download and install HP-UX RBAC as a separate product, refer to the *HP-UX RBAC Version B.11.23.02 Release Notes* on http://docs.hp.com. To download and install the full HP-UX 11i Security Containment feature set, refer to "Installing HP-UX 11i Security Containment" on page 22.

NOTE If you have installed the full HP-UX 11i Security Containment feature set, you already have HP-UX RBAC installed.

To install HP-UX RBAC, follow these steps:

Step 1. Be sure your system meets all requirements, as described in "Prerequisites and System Requirements" on page 21.

Step 2. Download the HP-UX 11i Security Containment bundle from Software Depot, as described in "Installing HP-UX 11i Security Containment" on page 22.

Step 3. Log in to your system as the root user.

Step 4. Install HP-UX RBAC by using the following command:

```
# swinstall -s /tmp/<security_containment_bundle>.depot RBAC
```

Step 5. Go on to "Verifying the HP-UX Role-Based Access Control Installation" on page 26.

NOTE If you install HP-UX RBAC version B.11.23.02 on a system with version B.11.23.01 already installed, and you have modified the database files, the new version **does not** overwrite the database files.

Verifying the HP-UX Role-Based Access Control Installation

Verify the installation of HP-UX RBAC with the following steps:

Step 1. Run the swverify command to ensure that the bundle installed correctly:

```
# swverify RBAC
```

If the installation is successful, many files are displayed and a success message appears after the verification is complete.

Step 2. Run the swlist command to verify that all parts of HP-UX RBAC are configured correctly on your system:

```
# swlist -a state -l fileset RBAC
```

If the product is configured correctly, each fileset is displayed as `configured`.

Installing HP-UX SMSE

The following procedure describes how to install only HP-UX SMSE from the HP-UX 11i Security Containment bundle. To download and install HP-UX SMSE as a separate product, refer to the *HP-UX SMSE Version B.11.23.01 Release Notes* on http://docs.hp.com. To install the full HP-UX 11i Security Containment feature set, refer to "Installing HP-UX 11i Security Containment" on page 22.

NOTE If you have installed the full HP-UX 11i Security Containment feature set, you already have HP-UX SMSE installed.

To install HP-UX Standard Mode Security Extensions, follow these steps:

Step 1. Be sure your system meets all requirements, as described in "Prerequisites and System Requirements" on page 21.

Step 2. Download the HP-UX 11i Security Containment bundle from Software Depot, as described in "Installing HP-UX 11i Security Containment" on page 22.

Step 3. Log on to your system as the root user.

Step 4. Install HP-UX Standard Mode Security Extensions by using the following command:

```
# swinstall -x autoreboot=true -s /tmp/<security_containment_bundle>.depot
TrustedMigration PHCO_32144 PHCO_32163 PHCO_32451
```

Step 5. Go on to "Verifying the HP-UX Standard Mode Security Extensions Installation" on page 28.

Verifying the HP-UX Standard Mode Security Extensions Installation

Verify the installation of HP-UX SMSE with the following steps:

Step 1. Run the swverify command to ensure that the bundle installed correctly:

```
# swverify TrustedMigration
```

If the installation is successful, many files are displayed and a success message appears after the verification is complete.

Step 2. Run the swlist command to verify that all parts of HP-UX SMSE are configured correctly on your system:

```
# swlist -a state -l fileset TrustedMigration
```

If the product is configured correctly, each fileset is displayed as `configured`.

Uninstalling HP-UX 11i Security Containment

This section describes how to remove the HP-UX 11i Security Containment product from your system.

CAUTION HP recommends that you leave the `SecurityExt` bundle on your system. Removing the entire bundle will remove many patches from your system. Instead, remove only the software products as described in the following procedure.

NOTE You must remove HP-UX 11i Security Containment before you remove HP-UX RBAC or HP-UX SMSE, or you must remove all components at the same time.

To remove HP-UX 11i Security Containment, follow these steps:

Step 1. Log in to your system as the root user.

Step 2. Remove HP-UX 11i Security Containment and all associated software by using the following command:

```
# swremove -x autoreboot=true TrustedMigration RBAC Containment
```

Step 3. Use the `swlist` command to verify that HP-UX 11i Security Containment and all associated components were removed from the system.

The `swlist` command will not report HP-UX 11i Security Containment if it was successfully removed from the system.

Uninstalling HP-UX RBAC

To remove HP-UX RBAC from your system, follow these steps:

Step 1. Log in to your system as the root user.

Step 2. Remove HP-UX RBAC by using the following command:

```
# swremove RBAC
```

Step 3. Use the `swlist` command to verify that HP-UX RBAC was removed from the system. The `swlist` command will not report HP-UX RBAC if it was removed from the system.

NOTE You must remove HP-UX 11i Security Containment before you remove HP-UX RBAC or HP-UX SMSE, or you must remove all components at the same time.

Uninstalling HP-UX Standard Mode Security Extensions

To remove HP-UX Standard Mode Security Extensions from your system, follow these steps:

Step 1. Log in to your system as the root user.

Step 2. Remove HP-UX Standard Mode Security Extensions using the following command:

```
# swremove TrustedMigration
```

Step 3. Use the `swlist` command to verify that HP-UX Standard Mode Security Extensions was removed from the system. The `swlist` command will not report HP-UX Standard Mode Security Extensions if it was removed from the system.

NOTE You must remove HP-UX 11i Security Containment before you remove HP-UX RBAC or HP-UX SMSE, or you must remove all components at the same time.

3 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

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 Sarbanes-Oxley—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 Versus Other RBAC Solutions

HP-UX RBAC offers several advantages over other role-based access control solutions available today, including:

- 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 (C2) audit system, to produce a single, unified audit trail
- Pluggable architecture for customizing access control decisions

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 3-1, a user can have a set of authorizations, each of which allows access to a resource.

Table 3-1 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`	•			
`hpux.network.nfs.stop`	•			
`hpux.network.nfs.config`	•			
`hpux.fs.backup`	•	•		
`hpux.fs.restore`	•	•		

NOTE Table 3-1 shows only the operation element of the authorizations—not the object element of the authorizations.

Simplifying Access Control with Roles

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 3-2, which lists authorizations assigned to roles, to Table 3-1, which lists the authorizations assigned to each user. By comparing these two tables, you can see how roles simplify authorization assignment.

Table 3-2 Example of Authorizations Per Role

Operation Component of Authorization	**Role**			
	UserAdmin	*NetworkAdmin*	*BackupOper*	*Admin*
`hpux.user.add`	•			•
`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 3-2 shows only the operation element of the authorizations—not the object element of the authorization.

NOTE HP-UX RBAC B.11.23.02 and higher versions also allow UNIX groups to be assigned to roles. Refer to "Assigning Roles to Groups" on page 47 for more information.

HP-UX RBAC Components

The 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 (ACL).

Access Control Policy Switch (ACPS)
: Determines whether a subject is authorized to perform an operation on an object.

Access Control Policy Module
: Evaluates HP-UX RBAC databases files and applies mapping policies to service access control requests.

management commands
: Edits and validates HP-UX RBAC database files.

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 3-1 on page 41, from its location in the HP-UX RBAC architecture, the ACPS provides a layer of abstraction between the access control policy modules and the applications that make access control decisions.

The ACPS has the following interfaces, described in detail in each of 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 the following manpages for more information on the ACPS and its interfaces:

- *acps*(3)
- *acps.conf*(4)
- *acps_api*(3)
- *acps_spi*(3)

HP-UX RBAC Configuration Files

Table 3-3 lists and briefly describes the HP-UX RBAC files.

Table 3-3 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.
`/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.

HP-UX RBAC Commands

Table 3-4 lists and briefly describes the HP-UX RBAC commands.

Table 3-4 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.

HP-UX RBAC Manpages

Table 3-5 lists and briefly describes the HP-UX RBAC manpages.

Table 3-5 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.
acps_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.
authadm(1m)	Describes `authadm` functionality and syntax.
cmdprivadm(1m)	Describes `cmdprivadm` functionality and syntax.
rbacdbchk(1m)	Describes `rbacdbchk` functionality and syntax.

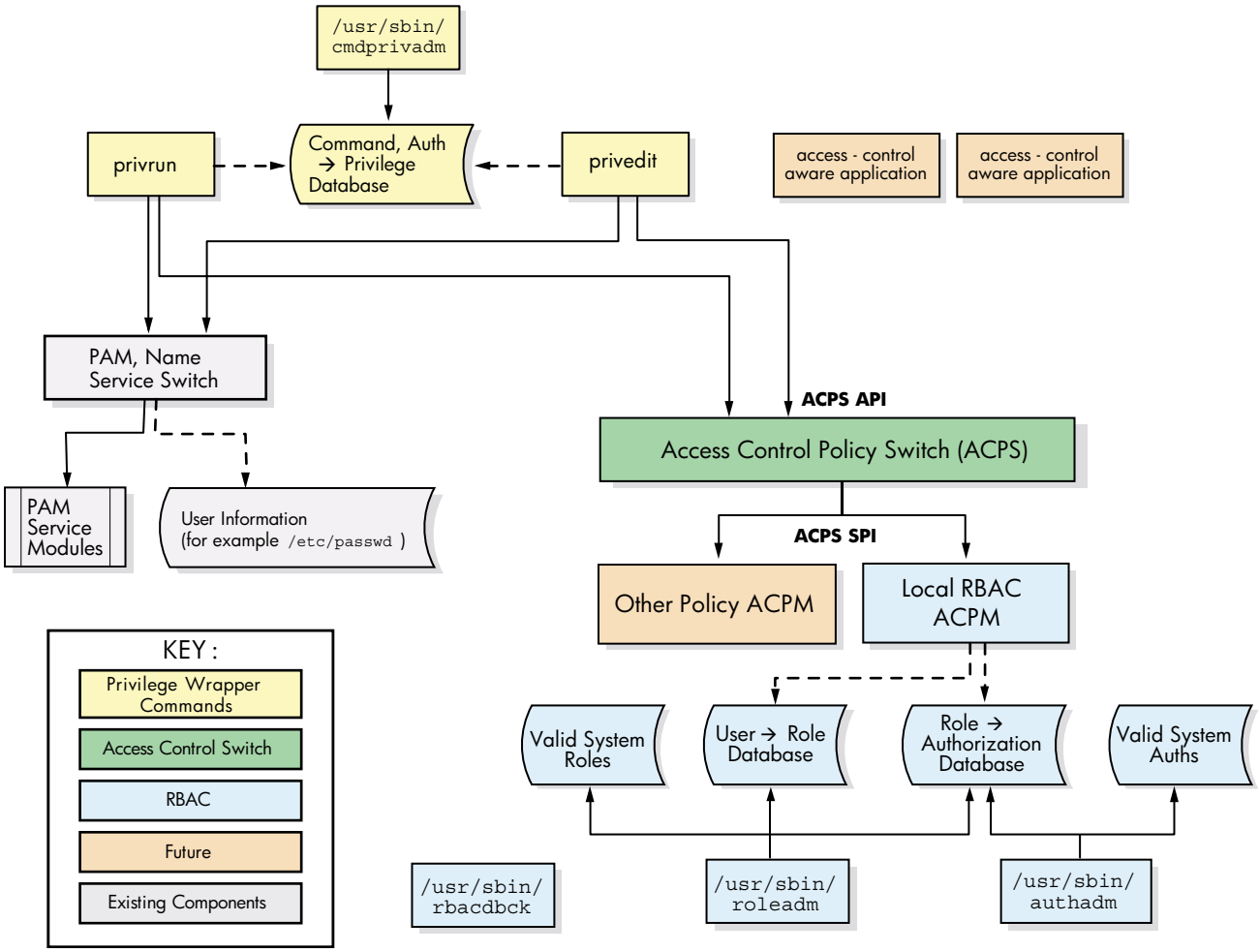
HP-UX RBAC Architecture

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 3-1 illustrates the HP-UX RBAC architecture.

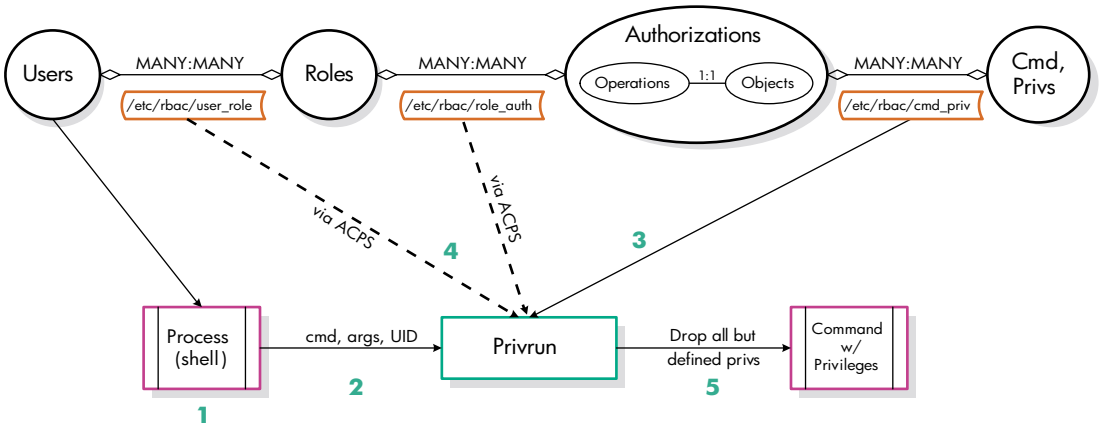
Figure 3-1 HP-UX RBAC Architecture



HP-UX RBAC Example Usage and Operation

Figure 3-2 and the subsequent footnotes illustrate a sample invocation of `privrun` and the configuration files that `privrun` uses to determine whether a user is allowed to invoke a command.

Figure 3-2 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 via 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.

Planning the HP-UX RBAC Deployment

Follow these planning steps before deploying HP-UX RBAC:

Step 1. Plan roles for users.

Step 2. Plan authorizations for the roles.

Step 3. Plan the authorization-to-command mappings.

Step 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.

Step 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. Use 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:

Step 1. List the system commands commonly used by each role.

Step 2. Compare the target commands from step 1 against the supplied sample `/etc/rbac/cmd_priv` database.

Step 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 your 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, using 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.

Step 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 your 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.

Refer to “Step 3: Configuring Additional Command Authorizations and Privileges” on page 49 for the procedure for configuring additional commands.

HP-UX RBAC Limitations and Restrictions

The 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. Refer to “HP-UX RBAC in Serviceguard Clusters” on page 58 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 (refer to Chapter 5, “Compartments,” on page 75). Remember the following items 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 (refer to *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.

Configuring HP-UX RBAC

Configuring HP-UX RBAC is a three-step process:

Step 1. Configuring roles.

Step 2. Configuring authorizations.

Step 3. Configuring 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.

Refer to the *Authorization* section in each of the HP-UX RBAC administrative commands manpages for more information about their authorizations.

This "Configuring HP-UX RBAC" section uses the example planning results and users in Table 3-6 to demonstrate the HP-UX RBAC administrative commands and configuration process.

Table 3-6 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`

Step 1: Configuring Roles

Configuring roles for users is a two-step process:

Step 1. Create roles.

Step 2. Assigning roles to users or groups.

Creating Roles

Use the `roleadm` command to create roles and assign them to users or UNIX 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
        | revoke user [role]
        | list [user=username][role=rolename][sys]
```

The 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 Refer to the *roleadm*(1m) manpage for more information.

The 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 UNIX 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.

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.

Assigning Roles to Groups

HP-UX RBAC also enables you to assign roles to UNIX 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
```

Step 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. The 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 3-6:

```
# 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.*, *)
```

Step 3: Configuring Additional Command Authorizations and Privileges

Define any additional commands that are not provided in the default configuration. You must have already created the authorizations needed to run the commands and assigned them 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 `cmdprivadm` has fewer sub-operations: only addition and removal.

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 Refer to *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.

Configuring HP-UX RBAC with Fine-Grained Privileges

NOTE HP-UX RBAC Version B.11.23.01 does not support the Fine-Grained Privileges component of the HP-UX 11i Security Containment feature.

Applications communicate with the system's resources using system calls, as this allows the operating system access to the system's resources. Certain system calls require special, elevated privileges for the application to access the operating system and system hardware. Before the release of the HP-UX 11i Security Containment feature—and specifically the Fine-Grained Privileges component of the HP-UX 11i Security Containment feature—`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—provide the means for 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 component of the HP-UX 11i Security Containment feature to run applications with additional privileges—but without `UID=0`.

If the Fine-Grained Privileges component is installed and enabled on the system, 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 Refer to *privileges*(5) for more information on the Fine-Grained Privileges component of the HP-UX 11i Security Containment feature.

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 "Using the privrun Command to Run Applications with Privileges" on page 57, 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. Refer to 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.

NOTE Use only the `cmdprivadm` command to configure fine-grained privileges 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.

Configuring HP-UX RBAC with Compartments

NOTE HP-UX RBAC version B.11.23.01 does not support the Compartments component of the HP-UX 11i Security Containment feature.

HP-UX RBAC can also use the Compartments component of the HP-UX 11i Security Containment feature to configure applications to run in a particular compartment. With the Compartments component 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.

Configuring HP-UX RBAC to Generate Audit Trails

On traditional root-based systems, where multiple administrators on the same system share the same root password, individual accountability is virtually impossible to achieve. Consequently, proper analysis of a security-significant event is difficult—sometimes impossible. However, recently introduced legislation—including the Health Insurance Portability and Accountability Act (HIPAA) and Sarbanes-Oxley—has helped to highlight the importance of understanding who did what and when. Because HP-UX RBAC provides the ability for commands to run with elevated privileges, it is important that you configure HP-UX RBAC to generate the appropriate audit trails.

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)

NOTE Refer to “Auditing” on page 104 for more information about auditing.

Auditing Based on HP-UX RBAC Criteria and the /etc/aud_filter File

NOTE HP-UX RBAC Version B.11.23.01 does not support auditing based on the 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 to generate audit records for. 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 `/etc/rbac/aud_filter` using an editor like vi 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 interface with `/etc/rbac/aud_filter`.

Procedure for Auditing HP-UX RBAC Criteria

The following steps describe how to configure an audit process to audit HP-UX RBAC criteria on your system:

Step 1. Configure the system to audit Passed or Failed events for the Administrator events by using the following command:

```
# audevent -PFe administrator
```

Step 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
```

Step 3. Execute an HP-UX RBAC command, for example:

```
# /usr/sbin/authadm add newauth
```

Step 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
```

Step 5. (Optional) Disable auditing on the system by using the following command:

```
# audsys -f
```

NOTE Refer to the *audit*(5), *audevent*(1m), *audsys*(1m), and *audisp*(1m) manpages to learn more about auditing HP-UX systems.

Using HP-UX RBAC

This section explains how to run the `privrun` and `privedit` commands to operate HP-UX RBAC.

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, he or she needs 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 Refer to the *privrun*(1m) and *rbac*(5) manpages for more about using the `privrun` command.

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. Refer to the latest *Managing Serviceguard* manual 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.

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. Refer to "Step 3: Configuring Additional Command Authorizations and Privileges" on page 49 and the *privedit*(1m) manpage for more information about flags and using the privedit command.

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

Refer to "HP-UX RBAC Access Control Policy Switch" on page 38, and *acps.conf*(4), *acps*(3), and *rbac*(5) for more information about the ACPS.

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.

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.

privrun -v Information

The second method for detecting issues 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
```

4 Fine-Grained Privileges

This chapter describes the fine-grained privileges feature of HP-UX 11i Security Containment. This chapter addresses the following topics:

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 *sam* (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 are replaced by the security containment model, including 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.

Refer to *privileges* (5) for more information.

Fine-Grained Privileges Components

The fine-grained privileges feature of HP-UX 11i Security Containment includes files, commands, and manpages. You can use these components to configure and administer fine-grained privileges.

Commands

Table 4-1 briefly describes the fine-grained privileges commands.

Table 4-1 Fine-Grained Privileges Commands

Commands	Description
`setfilexsec`	Sets various security attributes of binary files. The attributes currently include retained privileges, permitted privileges, compartment, and privilege awareness flag.
`getfilexsec`	Displays security attributes associated with binary executable files. The attributes include retained privileges, permitted privileges, compartment, and privilege awareness flag.
`getprocxsec`	Displays security attributes of processes. The attributes currently include effective privileges, retained privileges, permitted privileges, and compartment.

Manpages

Table 4-2 briefly describes the fine-grained privileges manpages.

Table 4-2 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.

Available Privileges

Table 4-3 describes each of the available privileges with the fine-grained privileges feature.

Table 4-3 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 `chown` system calls.
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 allow these operations.
PRIV_CMPTWRITE	Allows a process to write to a file or directory, bypassing compartment rules that otherwise would not allow this operation.
PRIV_COMMALLOWED	Allows a process to override compartment rules in the IPC and networking subsystems.
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-specific administrative operations, such as tape or disk formatting.
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_LIMIT	Allows a process to set resource and priority limits beyond the maximum limit values.
PRIV_LOCKRDONLY	Allows a process to set the locks of files with read-only permissions.

Table 4-3 Available Privileges (Continued)

Privilege	Description
PRIV_MKNOD	Allows a process to create character or block special files using mknod(2).
PRIV_MLOCK	Allows a process to access the `plock` system call.
PRIV_MOUNT	Allows a process to mount and unmount a file system.
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 a 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_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 RTP 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(2).
PRIV_SERIALIZE	Allows a process to force a target process to run serially with other processes configured with the PRIV_SERIALIZE privilege.
PRIV_SPUCTL	Allows a process to do certain administrative operations in the Instant Capacity product.
PRIV_SYSATTR	Allows a process to manage system attributes, including the setting of tunables, modifying the host name, domain name, and user quotas.

Table 4-3 Available Privileges (Continued)

Privilege	Description
PRIV_SYSNFS	Allows a process to perform NFS operations like exporting a file system, the getfh(2) 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.

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. Complete this registration process 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, refer to "Configuring HP-UX RBAC with Fine-Grained Privileges" on page 51.

TIP HP recommends you use HP-UX RBAC to configure applications that require variable privileges to run, depending on who is running the application.

To configure applications to use fine-grained privileges, use the `setfilexsec` command as follows:

```
# setfilexsec [options] filename
```

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.

Privilege Model

When you execute an application (binary file), it becomes a **process**. Processes have privilege sets associated with them; these privilege sets are generated when you execute the process. A process running from the same binary file can have different privileges at different invocations. Each process has three sets of privileges associated with it. These are the following:

- Permitted Privileges

  The maximum set of privileges a process can raise. A process can drop any privilege from this set, but cannot add any privileges to this set.

- Effective Privileges

  The set of privileges that is currently active for a process. A privilege-aware process can modify its effective privileges so that only necessary privileges are active at any given time. A 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 Privileges

  The set of privileges given to a new program by the current process when that executes a program via the execve() system call. A process can remove privileges from this set, but cannot add privileges to this set.

The retained privilege set is always a subset of the permitted privileges set.

Compound Privileges

Compound privileges are a shorthand way of specifying a set of simple privileges that can be granted to a process as a group.

The following are compound privileges:

- BASIC

  Basic privileges available to all processes.

- BASICROOT

  Privileges that provide powers usually associated with `UID=0`. These privileges together replace the power of root.

- POLICY

  Policy override privileges and policy configuration privileges. Policy override privileges override compartment rules. Policy configuration privileges control the configuration of fine-grained privileges.

For a complete list of the privileges in each of the sets described above, refer to *privileges* (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.

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.

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 Security Containment features 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.

Troubleshooting Fine-Grained Privileges

If something is not working on your system and you suspect the problem is occurring because of fine-grained privileges, you can check your 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.

Refer to *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: Run the following command to determine what privileges a process has:

```
# getprocxsec [options] [pid]
```

The following options are available with the `getprocxsec` command:

`-p`	Displays permitted privileges for the process.
`-e`	Displays effective privileges for the process.
`-r`	Displays retained privileges for the process.
`pid`	The process ID of the process for which privileges are displayed.

If the process does not have the correct privileges, configure the binary file that created this process with the correct privileges. Refer to “Configuring Applications with Fine-Grained Privileges” on page 70 for more information.

5 Compartments

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

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 your HP-UX system and the data that resides on it.

The compartments feature of the HP-UX Security Containment software enables you to isolate processes, or subjects, from each other and also from resources, or objects.

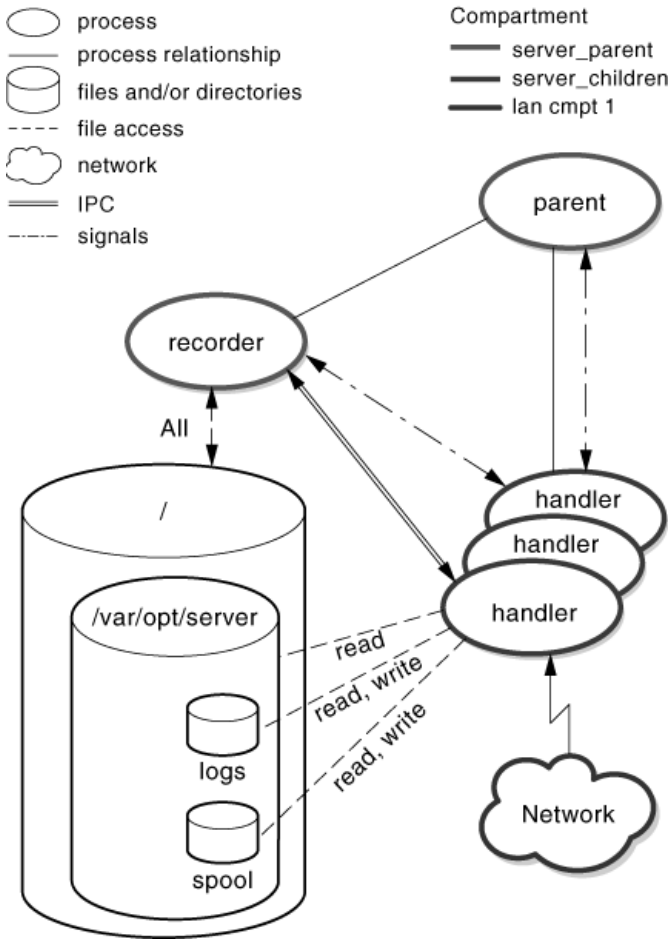
Conceptually, each process belongs to a compartment, and resources are handled in one of two ways. The resource can be labeled with the compartment of the creating process, for transient resources such as communication endpoints and shared memory. Alternately, resources can be associated with an access 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 your 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.

Compartment Architecture

Compartments isolate a process and its child processes within a system. Figure 5-1 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 5-1 Compartment Architecture



In Figure 5-1, 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 the handler processes is configured to communicate with the network.
- The recorder can access the file system.
- The handlers 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 via 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.

Default Compartment Configuration

When you enable the compartments feature, 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. The INIT compartment 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.

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 your security policy include requirements or problems that can be solved by using compartments?
- Does your 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 your system to specific compartments.

Consider the following recommendations when planning your compartment configuration:

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

  You can use the `#include` directive to create compartment configuration files anywhere on your 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 your 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.

Activating Compartments

To activate compartment rules on your system, follow these steps:

Step 1. Plan your compartment rules. See “Planning the Compartment Structure” on page 80 for more information.

TIP	HP recommends you plan your compartment rules configuration carefully. After you have edited your 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.

Step 2. Create compartment rules. See “Compartment Rules and Syntax” on page 86 for instructions on completing this step and for a complete description of compartment rules syntax.

Step 3. (Optional) Preview your 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 your system.

Step 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 your starting point if an editing problem occurs.

Step 5. Enable the compartments feature by entering the following command:

```
# cmpt_tune -e
```

Step 6. Reboot your system. This step is **mandatory**.

TIP	Keep your backup files; this makes it easier to revert to a prior configuration.

Modifying 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 your system until the next reboot.

Refer to “Changing Compartment Names” on page 82 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.

To modify your compartment configuration, follow these steps:

Changing Compartment Rules

Step 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 your starting point if an editing problem occurs.

Step 2. Examine your current compartment rules by the following command:

```
# getrules
```

Step 3. Create or modify compartment rules. See “Compartment Rules and Syntax” on page 86 for instructions on completing this step and for a complete description of compartment rules syntax.

Step 4. (Optional) Preview your 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 your system.

Step 5. (Optional) Make backup copies of the compartment configuration files.

Step 6. Run the `setrules` command to load the configured rules:

```
# setrules
```

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 refer to the new compartment.

CAUTION Do not change the name of the `INIT` compartment or otherwise modify the compartment definition. If you modify the `INIT` compartment definition, the compartments feature will not work properly.

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:

Compartment Configuration Files

Table 5-1 briefly describes the files you use with compartment components.

Table 5-1 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/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.

Compartment Commands

Table 5-2 contains the commands you use to manage compartments.

Table 5-2 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.

Compartment Manpages

Table 5-3 contains the manpages associated with compartments.

Table 5-3 Compartment Manpages

Manpage	Description
compartments (4)	Describes compartment rule syntax.
compartments (5)	Provides an overview of compartment functionality and describes the use of compartment rules.
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.
getrules (1M)	Describes `getrules` functionality and syntax.
setrules (1M)	Describes `setrules` functionality and syntax.
vhardlinks (1M)	Describes `vhardlinks` functionality and syntax.

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. Your rules file must have a `.rules` extension.

Refer to *compartments* (5) for additional information.

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> compartment <new_compartment_name> { <rules> }
```

For example:

```
sealed compartment server_children {

/* Deny all access to any file system objects ...*/

permission none /

}
```

`sealed` (Optional) A process in this compartment cannot gain privileges or change compartments by calling `execve()`.

`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.

NOTE The `INIT` compartment name is not case sensitive. `INIT`, `init`, and `Init` are all treated as the same compartment by the system. Do not use `INIT` or any variation for a new compartment name.

Compartment specifications are preprocessed with `cpp(1)` 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.

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.

The syntax for file system rules is as follows:

```
(permission|perm) <permission_list> <file_object>
```

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
```

`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.
- 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.

`file_object` The full path name of the file or directory.

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 `/`.

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>
```

For example:

```
/* allow the children to access UNIX domain*/
/* sockets created by the parent compartment*/

grant uxsock server_children
```

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.

`compartment_name` — The name of the other compartment where processes in this compartment can communicate with.

The second type of IPC rule governs process access. The syntax for this type of rule is as follows:

```
(send|receive) signal <compartment_name>
```

For example:

```
/* allow the parent to send signals to children*/
send signal server_children
```

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 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.

Network Rules

Network rules govern access to network interfaces. Network rules also govern communication between processes that use INET domain communication (TCP/IP sockets and streams). The default behavior is to deny access to the network.

Network endpoints are treated as objects labeled with the compartment of the process that creates them. However, a network endpoint can be created by one process, then passed to another process, which can run in a different compartment. Access checks are performed on the compartment containing the endpoint when the endpoint was created, not the current compartment. Additionally, the endpoint passes its compartment configuration to accepting endpoints when it receives new connections.

INET domain endpoints are frequently used for interprocess communications. Be sure to configure your compartments accordingly.

The syntax for a network rule is as follows:

```
(grant|deny)(server|client|bidir) (tcp|udp|raw [<protonum>])
[port <port_num>] [peer[port<port>]] <compartment_name>
```

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
```

Access — Grants or denies the compartment access to the network traffic in the specified compartment. The options are:

- grant
- deny

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 The compartment name associated with the peer endpoint or interface this rule applies to.

For more information about network rules, refer to *compartments* (4).

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_name[,interface_name][…]>
}
```

For example:

```
compartment iface0 {
/* Define the compartment for the network interface lan0 */

interface lan0

}

compartment other_ifaces {

/* Define the compartment for two of the other network interfaces */

interface lan1,lan5
```

`interface` Specifies that this is an interface definition.

<interface_name[,interface_name][...]>

A comma-separated list of interface names.

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[…]]>
```

For example:

```
/* Disallow all privileges except mount. */

disallowed privileges all,!mount

/* Disallow mount only. */

disallowed privileges none,mount
```

`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

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`.

Example Rules File

An example rules file is shipped with HP-UX 11i Security Containment, located in `/etc/cmpt/examples/sendmail.example`.

Configuring Applications in Compartments

You can configure an application to run in a particular compartment. Use 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
```

Alternately, you can use HP-UX RBAC to configure an application to run in a compartment. Refer to “Configuring HP-UX RBAC with Compartments” on page 53.

Troubleshooting Compartments

If something is not working on your system and you suspect the problem is occurring because of your compartment structure, you can check your compartment rules as follows.

Problem 1: Access is not being controlled according to the compartment rules I configured.

Solution: Your rules may not be set in the kernel. To check whether your rules are set in the kernel, follow these steps:

Step 1. Execute the following command:

```
# getrules
```

The `getrules` command displays the valid compartment rules in the kernel.

Step 2. Execute the following command:

```
# setrules -p
```

The `setrules` command with the `-p` option displays all rules configured on the system, including rules that have not been loaded into the kernel.

Step 3. Compare the output of step 1 to the output of step 2. If they are the same, all rules are loaded into the kernel.

If the output of step 1 is different from the output of step 2, go on to step 4.

Step 4. Execute the following command:

```
# setrules
```

The configured rules are loaded into the kernel.

Problem 2: A network interface on my compartment-enabled system is not accessible.

Solution: All network interfaces must be configured in a compartment. To check whether your network interface is configured in a compartment, follow these steps:

Step 1. Execute the following command:

```
# getrules
```

The `getrules` command displays the valid compartment rules in the kernel. Check the output for rules configuring the network interface.

If there are rules configuring the network interface in a compartment, go on to step 2 to check the rules syntax for errors.

If there are no rules for the network interface, go on to step 2.

Step 2. Execute the following command:

```
# setrules -p
```

The `setrules` command with the `-p` option displays all rules configured on the system, including rules that have not been loaded into the kernel.

If no rules are configured on the system, configure appropriate network interface rules. Refer to "Network Rules" on page 89 for network rules syntax.

The `setrules -p` command also checks for syntax errors. If there is a syntax error in your network interface rules, modify your rules as described in "Modifying Compartment Configuration" on page 82.

Step 3. Compare the output of step 1 to the output of step 2. If they are the same, all rules are loaded into the kernel.

If the output of step 2 displays rules for the network interface that were not present in the output of step 1, go on to step 4.

Step 4. Execute the following command:

```
# setrules
```

The configured rules are loaded into the kernel.

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:

Step 1. Execute the following command:

```
# vhardlinks
```

If the output shows an inconsistency, go on to step 2.

Step 2. Modify the rules to remove the inconsistency. Follow the procedure described in "Modifying Compartment Configuration" on page 82.

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
```

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. Refer to the latest editions of *Managing Serviceguard* and *Using Serviceguard Extension for RAC* for daemons required in Serviceguard and Serviceguard extensions for RAC cluster.

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 security containment features to 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`. Refer to *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. Oracle Real Application Cluster (RAC) processes must have access to the `libnmapi2` library, and must communicate with `cmsmgd`. Refer to *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.

6 Standard Mode Security Extensions

This chapter describes the Standard Mode Security Extensions features of HP-UX 11i Security Containment. This chapter addresses the following topics:

- “Auditing” on page 104

Overview

HP-UX Standard Mode Security Extensions (HP-UX SMSE) is a group of features that combine to enhance both user and operating system security for HP-UX 11i v2. Starting with the HP-UX 11i version 2 0505 update, 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.

The following new feature is included in HP-UX SMSE:

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

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.

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`.

Configuring Systemwide Attributes

To configure systemwide attributes, follow these steps:

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

Step 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.

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.

Configuration Files

Table 6-1 briefly describes the files you use with the user database.

Table 6-1 User Database Configuration Files

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

Commands

Table 6-2 briefly describes the commands you can use to modify and administer entries in the user database.

Table 6-2 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.

Attributes

The following security attributes are available for individual users:

Table 6-3 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.

Table 6-3 User Attributes (Continued)

Attribute	Description
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).

Manpages

Table 6-4 briefly describes the manpages you use with the user database.

Table 6-4 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.

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:

Step 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.

Step 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
```

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" on page 102 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.

Auditing

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.

HP-UX has two types of audit systems. On a trusted mode system, you enable auditing by using SAM or audit commands. On a standard mode system, auditing is a feature of the Standard Mode Security Extensions in HP-UX 11i Security Containment. The following sections describe auditing on a standard mode system.

Auditing Components

The auditing feature of HP-UX 11i Security Containment contains configuration files, commands, and manpages. These are listed in the following sections.

Commands

Table 6-5 contains a brief description of each auditing command.

Table 6-5 Audit Commands

Command	Description
`audevent`	Changes or displays event or system call status.
`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 by specifying the `AUDIT_FLAG=1` option.

Manpages

Table 6-6 contains a brief description of each manpage associated with the auditing feature.

Table 6-6 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 (5)	Gives introductory information about HP-UX auditing.

Auditing Your System

Use the following three procedures to plan, enable, and monitor auditing on your system.

Step 1: Planning Your Auditing Implementation

To plan your auditing implementation, follow these steps:

1. Determine which users to audit. By default, all users are selected for auditing.
2. Determine which events to audit. Use the audevent command to display a list of events and system calls that are currently selected for auditing.
3. Decide where you want to place the audit log files (audit trails) on your system. For more information on configuring your audit log files, refer to “Audit Log Files” on page 109.
4. Create a strategy to archive and back up audit files. Audit files often take up a lot of disk space and can overflow if you do not carefully plan file management.

For additional information about auditing system performance and administration that can help you plan your auditing implementation, refer to “Performance Considerations” on page 106 and “Guidelines for Administering Your Auditing System” on page 106.

Step 2: Enabling Auditing

To enable auditing on your system, follow these steps:

1. Configure the users you want to audit using the `userdbset` command. For more information on configuring auditing for users, refer to “Auditing Users” on page 107.
2. Configure the events you want to edit using the `audevent` command. For example, to configure the `admin`, `login`, and `moddac` events for auditing, enter the following command:

   ```
   # audevent -P -F -e admin -e login -e moddac
   ```

   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, refer to “Auditing Events” on page 107.
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 -e admin -e login -e moddac
   ```
4. Start the auditing system and define the log files using the `audsys` command. For example:

   ```
   # audsys -n -c primary_audit_file -s 1000
   ```
5. Set up your 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 your primary audit log file.

   b. Set `PRI_SWITCH` to the maximum size of your primary audit log file (in KB), at which audit logging switches to the auxiliary log file.

   c. Set `SEC_AUDFILE` to the name of your auxiliary log file.

   d. Set `SEC_SWITCH` to the maximum size of your secondary audit log file (in KB).

For more information about setting up primary and auxiliary audit log files, refer to “Audit Log Files” on page 109.

6. Set the AUDIT flag to 1 in the /etc/rc.config.d/auditing file to enable the auditing system to retain the current event configuration when the system is rebooted.

Step 3: Monitoring Audit Files

To view, monitor, and administer your audit files, follow these steps:

1. View the audit log files with the audisp command:

   ```
   # audisp audit_file
   ```

   Refer to “Viewing Audit Logs” on page 110 for details on using the audisp command.

2. Monitor the sizes of the log files with the audomon command:

   ```
   # audomon -p 20 -t 1 -w 90
   ```

   The audomon command also monitors the capacity of the file system on which the audit file is located. The audomon command takes the following arguments:

-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 for audit log file switch points on the console. 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%

   Refer to *audomon* (1M) for more information.

3. Set the audit log file monitor arguments in the /etc/rc.config.d/auditing file. Set the same values you used in step 2.
4. (Optional) Stop system auditing using the following command:

   ```
   # audsys -f
   ```

5. (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.

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.

Guidelines for Administering Your Auditing System

Use the following guidelines when administering your system:

- Check the audit logs according to your security policy. 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.

Auditing Users

By default, when system auditing is on, the audit status for all users is on. New users added to the system are automatically audited.

You can monitor what users are doing on HP-UX systems using the auditing. To change which users are audited, choose one of the following options:

- Audit all users.

  By default, audit status for all users is set to on 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.

  To turn off auditing for all users, follow these steps:

  1. Check to see which users are already being audited. To check, follow these steps:
     a. Check the `AUDIT_FLAG` setting in the `/etc/default/security` file.
     b. Check the `AUDIT_FLAG` setting stored in the user database using the following command:

        # **userdbget -a AUDIT_FLAG**
  2. Set `AUDIT_FLAG=0` in the `/etc/default/security` file.

- Audit specific users. To configure auditing for specific users, follow these steps:

  1. Deselect auditing for all users by setting the `AUDIT_FLAG=0` in the `/etc/default/security` file.
  2. Configure auditing for a specific user using the following command

     # **/usr/sbin/userdbset -u *user-name* 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.

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. Your security architect must determine which events to audit based on your business needs and any applicable government regulations.

NOTE HP recommends that you audit the following three events at a minimum:

- `admin`

- `login`
- `modaccess`

Configure the events you want to audit before you turn on the auditing system. When an event type is selected, its associated system calls are automatically enabled. To configure events for auditing, use the audevent command. The syntax for the audevent command is as follows:

```
# audevent [options]
```

The following options are commonly used with the audevent command:

Table 6-7 audevent command options

audevent options	Description
`-P`	Logs successful event operations
`-F`	Logs unsuccessful event operations
`-e [event]`	Specifies an event to log
`-l`	Displays a complete list of event types and associated system calls
`-S` or `-s`	Change event or system call audit status
no option	display the current status of the selected events or system calls

For example, to configure admin, login, and modaccess for auditing, enter the following command:

```
# audevent -P -F -e admin -e login -e moddac
```

Both `Audit Success` and `Audit Failure` are set as event types for monitoring successful and failed events or system calls. This is the minimum event type selection recommended for running a system.

A record is written when an event type is selected for auditing, and the user initiating the event has been selected for auditing. The `login` event is an exception. Once selected, the `login` event will be recorded whether or not the user logging in has been selected for auditing.

Streamlining Audit Log Data

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.

You can turn off self-auditing programs by turning off auditing on the system.

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

Most self-auditing programs generate audit data under a single event category. For example, the `audsys` command generate 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.

Audit Log Files

All auditing data is written to an audit log file. The primary log file is where audit records are collected. When this file approaches a predefined capacity (its Audit File Switch (AFS) size), or when the file system on which it resides approaches a predefined capacity (its File Space Switch (FSS) size), the auditing subsystem issues a warning. When either the AFS or the FSS of the primary log file is reached, the auditing subsystem attempts to switch to the auxiliary log file for recording audit data. If no auxiliary log file is specified, the primary log file continues to grow.

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

NOTE 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 your audit log files. You can assess the size of your file systems using the `bdf` command. HP recommends you configure your log files to **at least** the following parameters:

- ❏ The file system must have more than 5000 KB available for the primary audit log file.
- ❏ It must have more than 20% of its total file space available.

TIP HP recommends that the primary and auxiliary audit log files reside on separate file systems.

The growth of audit log files is closely monitored by the audit overflow monitor daemon, `audomon`, to insure that no audit data is lost.

Configuring Audit Log Files

Use the `audsys` command to specify the primary audit log file and the (optional) auxiliary audit log file to collect auditing data. For example:

```
# audsys -c primary_audit_file -s 5000 -x auxiliary_audit_file -z 2500
```

This example specifies a primary audit file 5000K in size, and an auxiliary audit file 2500K in size. Refer to *audsys* (1M) for more information about using the `audsys` command to configure audit log files.

NOTE If you specify the name of an existing file as your auxiliary audit log file, the contents of the file will be overwritten.

CAUTION If the file system containing the primary log file is full and no auxiliary log file is specified, 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 the *audsys* (1M) manpage.

Viewing Audit Logs

Auditing accumulates a lot of data. Use the `audisp` command to selects the data you want to view:

```
# /usr/sbin/audisp audit_file
```

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` Displays start time.

`-s` Displays end time.

`-u` *`user-name`* Displays information for a specific user.

`-l` *`terminal-name`* Displays information for a specific terminal.

`-e` *`process-name`* Displays all events for a specific process.

`>` *`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 your 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 zero if the call does not complete successfully.
- Auditing the superuser while changing the event or system call 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.

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_file***

- Direct the log output to `/tmp/mylogoutput`:

  # **/usr/sbin/audisp *audit_file* > /tmp/mylogoutput**

- View successful events only:

  # **/usr/sbin/audisp -p *audit_file***

- View activities owned by user joe:

  # **/usr/sbin/audisp -u joe *audit_file***

- View activities on terminal, `ttypa`:

  # **/usr/sbin/audisp -l ttypa *audit_file***

- View login events only:

  # **/usr/sbin/audisp -e login *audit_file***