Managing Auditing in Oracle® Solaris 11.2

Part No: E37127
July 2014

ORACLE

Contents

Using This Documentation

Managing Auditing in Oracle® Solaris 11.2 documents the auditing feature of Oracle Solaris.

- **Overview** – Describes how to administer auditing on an Oracle Solaris system or on a network of systems.
- **Audience** – System administrators who must implement security on the enterprise.
- **Required knowledge** – Familiarity with security concepts and terminology.

Product Documentation Library

Late-breaking information and known issues for this product are included in the documentation library at http://www.oracle.com/pls/topic/lookup?ctx=E36784.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Feedback

Provide feedback about this documentation at http://www.oracle.com/goto/docfeedback.

♦ ♦ ♦ CHAPTER 1

About Auditing in Oracle Solaris

The auditing subsystem of Oracle Solaris keeps a record of how the system is being used. The audit service includes tools to assist with the analysis of the auditing data.

This chapter introduces how auditing works in Oracle Solaris:

For planning suggestions, see Chapter 2, "Planning for Auditing". For procedures to configure auditing at your site, see the following chapters:

- Chapter 3, "Managing the Audit Service"
- Chapter 4, "Monitoring System Activities"
- Chapter 5, "Working With Audit Data"
- Chapter 6, "Analyzing and Resolving Audit Service Issues"

For reference information, see Chapter 7, "Auditing Reference".

What's New in the Audit Service in Oracle Solaris

This section highlights information for existing customers about important new features in the Oracle Solaris audit service.

- Audit records from an Oracle Solaris system can plug in to Oracle Audit Vault and Database Firewall, which can then be used to obtain information about audited events on Oracle Solaris systems.

- The audit configuration files, `audit_class`, `audit_event`, and `audit_warn`, have two package attributes set. The `preserve=renamenew` attribute enables the files to be modified, and the modifications are retained across package updates and package fixes. The `overlay=allow` attribute enables the files to be replaced by files in packages that a customer creates.

What Is Auditing?

Auditing is the collecting of data about the use of system resources. The audit data provides a record of security-related system events. This data can then be used to assign responsibility for actions that take place on a host.

Successful auditing starts with two security features: identification and authentication. At each login, after a user supplies a user name and PAM (pluggable authentication module) authentication succeeds, a unique and immutable *audit user ID* is generated and associated with the user, and a unique audit session ID is generated and associated with the user's process. The audit session ID is inherited by every process that is started during that login session. When a user switches to another user, all user actions are tracked with the same audit user ID. For more details about switching identity, see the `su`(1M) man page. Note that by default, certain actions such as booting and shutting down the system are always audited.

The audit service enables the following operations possible:

- Monitoring security-relevant events that take place on the host
- Recording the events in a network-wide audit trail
- Detecting misuse or unauthorized activity
- Reviewing patterns of access and the access histories of individuals and objects
- Discovering attempts to bypass the protection mechanisms
- Discovering extended use of privilege that occurs when a user changes identity

---

Note - To maintain security, not all audited events are viewable such as changed passwords. For more details, see “Audit Records and Audit Tokens” on page 15.

Audit Terminology and Concepts

The following terms are used to describe the audit service. Some definitions include pointers to more complete descriptions.

audit class — A grouping of audit events. Audit classes provide a way to select a group of events to be audited.

	For more information, see “Audit Classes and Preselection” on page 14, and the `audit_flags`(5), `audit_class`(4), and `audit_event`(4) man pages.
audit file system	A repository of audit files in binary format. For more information, see “Audit Logs” on page 16 and the `audit.log`(4) man page.
audit event	A security-related system action that is auditable. For ease of selection, events are grouped into audit classes. For more information, see “Audit Events” on page 13 and the `audit_event`(4) man page.
audit flag	An audit class that is supplied as an argument to a command or keyword. A flag can be prefixed by a plus sign or minus sign to indicate that the class is audited for success (`+`) or failure (`-`). A preceding caret (`^`) indicates that a success is not to be audited (`^+`) or a failure is not to be audited (`^-`). For more information, see the `audit_flags`(5) man page and “Audit Class Syntax” on page 115.
audit plugin	A module that transfers the audit records in the queue to a specified location. The `audit_binfile` plugin creates binary audit files. Binary files comprise the audit trail, which is stored on audit file systems. The `audit_remote` plugin sends binary audit records to a remote repository. The `audit_syslog` plugin summarizes selected audit records in the `syslog` logs. For more information, see “Audit Plugin Modules” on page 15 and the module man pages, `audit_binfile`(5), `audit_remote`(5), and `audit_syslog`(5).
audit policy	A set of auditing options that you can enable or disable at your site. You can specify whether to record certain kinds of audit data, and whether to suspend auditable actions when the audit queue is full. For more information, see “Understanding Audit Policy” on page 33 and the `auditconfig`(1M) man page.
audit record	Audit data that is collected in the audit queue. An audit record describes a single audit event. Each audit record is composed of audit tokens. For more information, see “Audit Records and Audit Tokens” on page 15 and the `audit.log`(4) man page.

audit token	A field of an audit record or event. Each audit token describes an attribute of an audit event, such as a user, a group, a program, or other object. For more information, see "Audit Token Formats" on page 122 and the `audit.log`(4) man page.
audit trail	A collection of one or more audit files that store the audit data from all audited systems that use the default plugin, `audit_binfile`. For more information, see "Audit Trail" on page 120.
local auditing	The collecting of audit records that are generated on the local system. The records can be generated in the global zone or in non-global zones, or both. For more information, see "Audit Plugin Modules" on page 15.
post-selection	The choice of which audit events to examine in the audit trail. The default active plugin, `audit_binfile`, creates the audit trail. A post-selection tool, the `auditreduce` command, selects records from the audit trail. For more information, see the `auditreduce`(1M) and `praudit`(1M) man pages.
preselection	The choice of which audit classes to monitor. The audit events of preselected audit classes are collected in the audit queue. Audit classes that are not preselected are not audited, so their events do not appear in the queue. For more information, see "Audit Classes and Preselection" on page 14 and the `audit_flags`(5) and `auditconfig`(1M) man pages.
public object	A file that is owned by the `root` user and readable by the world. For example, files in the `/etc` directory and the `/usr/bin` directory are public objects. Public objects are not audited for read-only events. For example, even if the `file_read` (`fr`) audit class is preselected, the reading of public objects is not audited. You can override the default by changing the `public` audit policy option.
remote auditing	The audit remote server (ARS) that receives and stores audit records from a system that is being audited and is configured with an active `audit_remote` plugin. To distinguish an audited system from an ARS, the audited system can be referred to as the "locally audited system."

For more information, see the `-setremote` options on the `auditconfig`(1M) man page and "Audit Remote Server" on page 117.

Audit Events

Audit events represent auditable actions on a system. Audit events are listed in the `/etc/security/audit_event` file. Each audit event is connected to a system call or user command, and is assigned to one or more audit classes. For a description of the format of the `audit_event` file, see the `audit_event`(4) man page.

For example, the `AUE_EXECVE` audit event audits the `execve` system call. The command `auditrecord -e execve` displays this entry:

```
# auditrecord -e execve
execve
system call execve              See execve(2)
event ID    23                  AUE_EXECVE
class       ps,ex               (0x0000000040100000)
header
path
[attribute]                 omitted on error
[exec_arguments]            output if argv policy is set
[exec_environment]          output if arge policy is set
subject
[use_of_privilege]
return
```

When you preselect either the audit class `ps` or the audit class `ex`, then every `execve` system call is recorded in the audit queue.

Auditing handles *attributable* and *non-attributable* events. Audit policy divides events into *synchronous* and *asynchronous* events, as follows:

- **Attributable events –** Events that can be attributed to a user. The `execve` system call can be attributed to a user, so the call is considered an attributable event. All attributable events are synchronous events.
- **Non-attributable events –** Events that occur at the kernel-interrupt level or before a user is authenticated. The `na` audit class handles audit events that are non-attributable. For example, booting the system is a non-attributable event. Most non-attributable events are asynchronous events. However, non-attributable events that have associated processes, such as a failed login, are synchronous events.
- **Synchronous events –** Events that are associated with a process in the system. Synchronous events are the majority of system events.
- **Asynchronous events –** Events that are not associated with any process, so no process is available to be blocked and later started. Initial system boot and PROM enter and exit events are examples of asynchronous events.

In addition to the audit events that are defined by the audit service, third-party applications can generate audit events. Audit event numbers from 32768 to 65535 are available for third-party applications. Vendors need to contact their Oracle Solaris representative to reserve event numbers and obtain access to the audit interfaces.

Audit Classes and Preselection

Each audit event belongs to an *audit class*. Audit classes are convenient containers for large numbers of audit events. When you *preselect* a class to be audited, all the events in that class are recorded in the audit queue. For example, when you preselect the `ps` audit class, `execve`, `fork`, and other system calls are recorded.

You can preselect for events on a system and for events initiated by a particular user.

- **System-wide preselection –** Specify the system-wide defaults for auditing by using the `-setflags` and `-setnaflags` options to the `auditconfig` command.

Note - If the `perzone` policy is set, default audit classes can be specified in every zone. For `perzone` auditing, the defaults are zone-wide, not system-wide.

- **User-specific preselection –** Specify differences from the system-wide auditing defaults for individual users by configuring the audit flags for the user. The `useradd`, `roleadd`, `usermod`, and `rolemod` commands place the `audit_flags` security attribute in the `user_attr` database. The `profiles` command places audit flags for rights profiles in the `prof_attr` database.

  The audit preselection mask determines which classes of events are audited for a user. For a description of the user preselection mask, see “Process Audit Characteristics” on page 119. For the configured audit flags that are used, see “Order of Search for Assigned Rights” in “Securing Users and Processes in Oracle Solaris 11.2 ”.

Audit classes are defined in the `/etc/security/audit_class` file. Each entry contains the audit mask for the class, the name for the class, and a descriptive name for the class. For example, the `lo` and `ps` class definitions appear in the `audit_class` file as follows:

```
0x0000000000001000:lo:login or logout
0x0000000000100000:ps:process start/stop
```

The audit classes include the two global classes: `all` and `no`. The audit classes are described in the `audit_class`(4) man page. For the list of classes, read the `/etc/security/audit_class` file.

The mapping of audit events to classes is configurable. You can remove events from a class, add events to a class, and create a new class to contain selected events. For the procedure, see

"How to Change an Audit Event's Class Membership" on page 55. To view the events that are mapped to a class, use the `auditrecord -c` *class* command.

Audit Records and Audit Tokens

Each *audit record* records the occurrence of a single audited event. The record includes information such as who did the action, which files were affected, what action was attempted, and where and when the action occurred. The following example shows a `login` audit record with three tokens, `header`, `subject`, and `return`:

```
header,69,2,login - local,,example_system,2010-10-10 10:10:10.020 -07:00
subject,jdoe,jdoe,staff,jdoe,staff,1210,4076076536,69 2 example_system
return,success,0
```

The type of information that is saved for each audit event is defined by a set of *audit tokens*. Each time an audit record is created for an event, the record contains some or all of the tokens that are defined for the event. The nature of the event determines which tokens are recorded. In the preceding example, each line begins with the name of the audit token. The content of the audit token follows the token name. Together, the `header`, `subject`, and `return` audit tokens comprise the `login - local` audit record. To display the tokens that comprise an audit record, use the `auditrecord -e` *event* command.

Note - Files with the `sensitive` system attribute do not have their contents or content changes included in the audit record. The attribute ensures that no sensitive information in specific files, such as passwords, PINs, keys, and so on, is accessible to anyone. For more details, refer to the `pfedit`(1M) man page.

For a detailed description of the structure of each audit token with an example of `praudit` output, see "Audit Token Formats" on page 122. For a description of the binary stream of audit tokens, see the `audit.log`(4) man page.

Audit Plugin Modules

The audit plugin modules direct the audit records from the audit queue to a file or repository. At least one plugin must be active. By default, the `audit_binfile` plugin is active. You configure plugins with the `auditconfig -setplugin` *plugin-name* command.

The audit service provides the following plugins:

- `audit_binfile` plugin – Handles delivery of the audit queue to the binary audit files. For more information, see the `audit.log`(4) man page.

- `audit_remote` plugin – Handles secure delivery of binary audit records from the audit queue to a configured remote server. The `audit_remote` plugin uses the `libgss` library to authenticate the server. The transmission is protected for privacy and integrity.
- `audit_syslog` plugin – Handles delivery of selected records from the audit queue to the `syslog` logs.

For information about how to configure a plugin, see the `auditconfig`(1M) man page. For examples of plugin configuration, see the tasks in "Configuring Audit Logs" on page 73.
For information about the plugins, see the `audit_binfile`(5), `audit_remote`(5), and `audit_syslog`(5) man pages.

Audit Logs

Audit records are collected in audit logs. The audit service provides three output modes for audit records.

- Logs that are called *audit files* store audit records in binary format. The set of audit files from a system or site provides a complete audit record. The complete audit record is called the *audit trail*. These logs are created by the `audit_binfile` plugin, and can be reviewed by the `praudit` and `auditreduce` post-selection commands.
- The `audit_remote` plugin streams audit records to a remote repository. The repository is responsible for maintaining an audit trail and supplying post-selection tools.
- The `syslog` utility collects and stores text summaries of the audit record. A `syslog` record is not complete. The following example shows a `syslog` entry for a `login` audit record:

```
Oct 10  10:10:20 example_system auditd: [ID 6472 audit.notice] \
login - login ok session 4076172534 by root as root:other
```

A site can configure auditing to collect audit records in all formats. You can configure the systems at your site to use binary mode locally, to send binary files to a remote repository, and to use `syslog` mode. The following table compares binary audit records with `syslog` audit records.

TABLE 1-1 Comparison of Binary, Remote, and `syslog` Audit Records

Feature	Binary and Remote Records	`syslog` Records
Protocol	Binary – Writes to the file system	Uses UDP for remote logging
	Remote – Streams to a remote repository	
Data type	Binary	Text
Record length	No limit	Up to 1024 characters per audit record

Feature	Binary and Remote Records	`syslog` Records
Location	Binary – Stored in a `zpool` on the system Remote – Remote repository	Stored in a location that is specified in the `syslog.conf` file
How to configure	Binary – Set the `p_dir` attribute on the `audit_binfile` plugin Remote – Set the `p_hosts` attribute on the `audit_remote` plugin and make the plugin active	Make the `audit_syslog` plugin active and configure the `syslog.conf` file
How to read	Binary – Typically, in batch mode, browser output in XML Remote – Repository dictates the procedure	In real time or searched by scripts that you have created for `syslog` Plain text output
Completeness	Guaranteed to be complete and to appear in the correct order	Not guaranteed to be complete
Time stamp	Coordinated Universal Time (UTC)	Time on the system that is being audited

For more information about plugins and audit logs, refer to the following:

- `audit_binfile`(5) man page
- `audit_syslog`(5) man page
- `audit.log`(4) man page
- “How to Assign Audit Space for the Audit Trail” on page 77
- “How to Configure `syslog` Audit Logs” on page 86

About Binary Records

Binary records provide the greatest security and coverage. Binary output meets the requirements of security certifications, such as the Common Criteria (http://www.commoncriteriaportal.org/) audit requirements.

The `audit_binfile` plugin writes the records to a file system that you protect from snooping. On a single system, all binary records are collected and displayed in order. The UTC time stamp on binary logs enables accurate comparison when systems on one audit trail are distributed across time zones. The `praudit -x` command enables you to view the records in a browser in XML. You can also use scripts to parse the XML output.

The `audit_remote` plugin writes the records to a remote repository. The repository handles storage and post-selection.

About `syslog` Audit Records

In contrast, the `syslog` records might provide greater convenience and flexibility. For example, you can collect the `syslog` data from a variety of sources. Also, when you monitor `audit.notice` events in the `syslog.conf` file, the `syslog` utility logs an audit record summary with the current time stamp. You can use the same management and analysis tools that you have developed for `syslog` messages from a variety of sources, including workstations, servers, firewalls, and routers. The records can be viewed in real time, and can be stored on a remote system.

By using `syslog.conf` to store audit records remotely, you protect log data from alteration or deletion by an attacker. However, consider the following drawbacks to the `syslog` mode.

- The records are susceptible to network attacks such as denial of service and spoofed source addresses.
- The UDP protocol can drop packets or can deliver packets out of order.
- The 1024 character limit for `syslog` entries can cause some audit records to be truncated in the log.
- On a single system, not all audit records are collected, and might not be displayed in order.
- Each audit record is stamped with the local system's date and time. Thus, you cannot rely on the time stamp to construct an audit trail for several systems.

Storing and Managing the Audit Trail

When the `audit_binfile` plugin is active, an *audit file system* holds audit files in binary format. A typical installation uses the `/var/audit` file system and can use additional file systems. The contents of all audit file systems comprise the *audit trail*. Audit records are stored in these file systems in the following order:

- **Primary audit file system–** The `/var/audit` file system, the default file system for audit files for a system
- **Secondary audit file systems –** File systems where the audit files for a system are placed at administrator discretion

The file systems are specified as arguments to the `p_dir` attribute of the `audit_binfile` plugin. A file system is not used until a file system that is earlier in the list is full. For an example with a list of file system entries, see “How to Create ZFS File Systems for Audit Files” on page 74.

Placing the audit files in the default audit root directory assists the audit reviewer when reviewing the audit trail. The `auditreduce` command uses the audit root directory to find all files in the audit trail. The default audit root directory is `/var/audit`.

You can use the following options with the `auditreduce` command:

- The `-M` option to the `auditreduce` command can be used to specify the audit files from a specific machine.
- The `-S` option can be used to specify a different audit file system.

For examples of the use of the `auditreduce` command, see "How to Merge Audit Files From the Audit Trail" on page 97. For more information, see the `auditreduce`(1M) man page.

The audit service provides commands to combine and filter files from the audit trail. The `auditreduce` command can merge audit files from the audit trail. The command can also filter files to locate particular events. The `praudit` command reads the binary files. Options to the `praudit` command provide output that is suitable for scripting and for browser display.

Ensuring Reliable Time Stamps

When you merge audit logs from several systems, the date and time on those systems must be accurate. Similarly, when you send audit logs to a remote system, the recording system and the repository system must have accurate clocks. The Network Time Protocol (NTP) keeps system clocks accurate and coordinated. For more information, see Chapter 3, "Time-Related Services," in "Introduction to Oracle Solaris 11.2 Network Services " and the `xntpd` (1M) man page.

Managing a Remote Repository

After the `audit_remote` plugin is configured, a remote repository receives the audit records. The ARS provides a receiver for audit records. The audit records stream to the ARS over a protected connection and can be stored similarly to how they are stored locally. To configure an ARS, see "How to Configure a Remote Repository for Audit Files" on page 82. For a description of the ARS, see "Audit Remote Server" on page 117 and the `ars`(5) man page.

How Is Auditing Related to Security?

Auditing helps to detect potential security breaches by revealing suspicious or abnormal patterns of system usage. Auditing also provides a means to trace suspect actions back to a particular user, thus serving as a deterrent. Users who know that their activities are being audited are less likely to attempt malicious activities.

To protect a computer system, especially a system on a network, requires mechanisms that control activities before system processes or user processes begin. Security requires tools that

monitor activities as the activities occur. Security also requires reports of activities after the activities have happened.

Set audit parameters before users log in or system processes begin, because most audit activity involves monitoring current events and reporting the events that meet the specified parameters. How the audit service monitors and reports these events is discussed in detail in Chapter 2, “Planning for Auditing” and Chapter 3, “Managing the Audit Service”.

Auditing cannot prevent hackers from unauthorized entry. However, the audit service can report, for example, that a specific user performed specific actions at a specific time and date. The audit report can identify the user by entry path and user name. Such information can be reported immediately to your terminal and to a file for later analysis. Thus, the audit service provides data that helps you determine the following:

- How system security was compromised
- What loopholes need to be closed to ensure the desired level of security

How Does Auditing Work?

Auditing generates audit records when specified events occur. Most commonly, events that generate audit records include the following:

- System startup and system shutdown
- Login and logout
- Process creation or process destruction, or thread creation or thread destruction
- Opening, closing, creating, destroying, or renaming of objects
- Use of rights
- Identification actions and authentication actions
- Permission changes by a process or user
- Administrative actions, such as installing a package
- Site-specific applications

Audit records are generated from three sources:

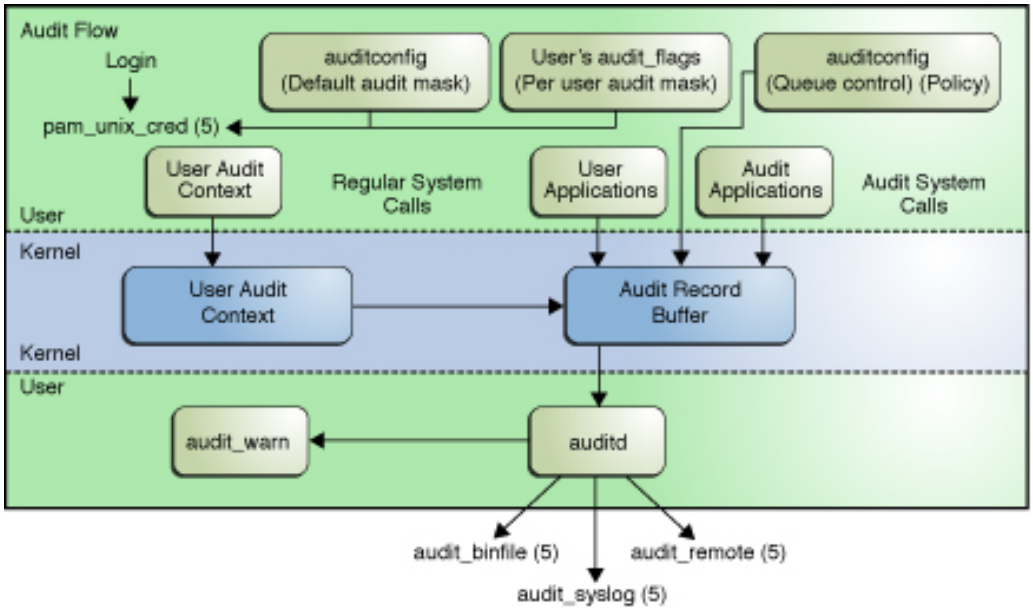
- By an application
- As a result of an asynchronous audit event
- As a result of a process system call

After the relevant event information has been captured, the information is formatted into an audit record. Contained in each audit record is information that identifies the event, what caused the event, the time of the event, and other relevant information. This record is then placed in an audit queue and sent to the active *plugins* for storage. At least one plugin must be active, although all plugins can be active. Plugins are described in “How Is Auditing Configured?” on page 21 and “Audit Plugin Modules” on page 15.

How Is Auditing Configured?

During system configuration, you *preselect* which classes of audit records to monitor. You can also fine-tune the degree of auditing that is done for individual users. The following figure shows details of the flow of auditing in Oracle Solaris.

FIGURE 1-1 Flow of Auditing



After audit data is collected in the kernel, plugins distribute the data to the appropriate locations.

- The `audit_binfile` plugin places binary audit records in the `/var/audit` file system. By default, the `audit_binfile` plugin is active. Post-selection tools enable you to examine interesting parts of the audit trail.

  Audit files can be stored in one or more ZFS pools. These pools can be on different systems and on different but linked networks. The collection of audit files that are linked together is considered an *audit trail*.
- The `audit_remote` plugin sends binary audit records across a protected link to a remote repository.
- The `audit_syslog` plugin sends text summaries of audit records to the `syslog` utility.

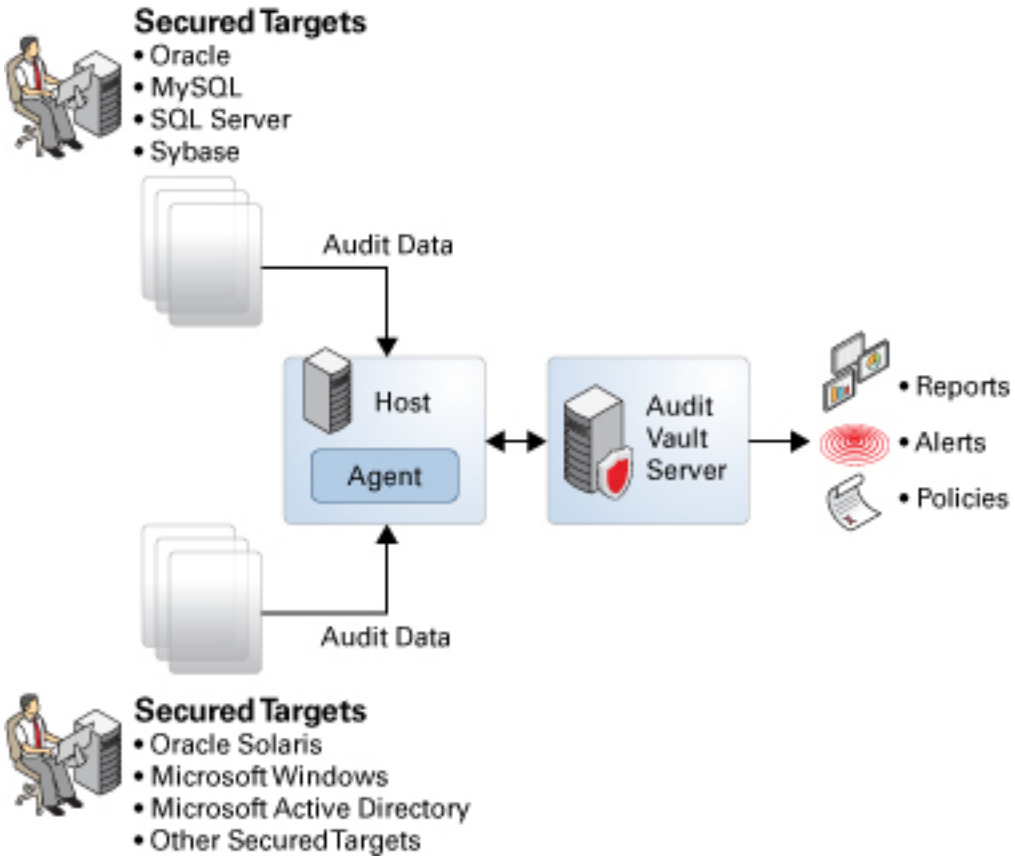
Systems that install non-global zones can audit all zones identically from the global zone. These systems can also be configured to collect different records in the non-global zones. For more information, see “Auditing and Oracle Solaris Zones” on page 114.

Using Oracle Audit Vault and Database Firewall for Storage and Analysis of Audit Records

Audit records from an Oracle Solaris system can plug in to Oracle Audit Vault and Database Firewall, release 12.1.0.0. Oracle Audit Vault and Database Firewall automates the consolidation and monitoring of audit data from Oracle and non-Oracle databases. Oracle Audit Vault and Database Firewall can then be used for analysis and reports of audited events on Oracle Solaris systems. For more information, see Oracle Audit Vault and Database Firewall (http://www.oracle.com/technetwork/products/audit-vault/overview/index.html).

The following figure shows how Oracle Audit Vault and Database Firewall collects Oracle Solaris audit records from designated secured targets. A secured target is any system that stores audit records or data.

FIGURE 1-2 Oracle Solaris and Audit Vault



A host is designated to run the AV agent that communicates with Oracle Audit Vault and Database Firewall. The agent enables Oracle Audit Vault and Database Firewall to receive and process audit data from secured targets. The agent reads the audit records from a designated audit trail on the secured target. These audit records are encoded in the native binary format. The agent converts the data to a format parseable by Oracle Audit Vault and Database Firewall. Oracle Audit Vault and Database Firewall receives the data and generates reports for administrators and security managers as required.

The agent can be installed on a secured target instead of on a separate host or system. Multiple hosts with agents can also be configured to connect to the Audit Vault server. However, when registering secured targets, indicate a specific host with which the AV server communicates to obtain audit data.

To configure Oracle Audit Vault and Database Firewall to accept audit records from both Oracle Solaris secured targets and non-Oracle Solaris secured targets, ensure that the agent is installed and activated on the designated host system. For more information, see the Oracle Audit Vault and Database Firewall documentation (http://www.oracle.com/technetwork/products/audit-vault/documentation/index.html).

Auditing on a System With Oracle Solaris Zones

A zone is a virtualized operating system environment that is created within a single instance of the Oracle Solaris OS. The audit service audits the entire system, including activities in zones. A system that has installed non-global zones can run a single audit service to audit all zones identically. Or, it can run one audit service per zone, including the global zone.

Sites that satisfy the following conditions can run a single audit service:

- The site requires a single-image audit trail.
- The non-global zones are used as application containers. The zones are part of one administrative domain. That is, no non-global zone has customized naming service files.

  If all the zones on a system are within one administrative domain, the `zonename` audit policy can be used to distinguish audit events that are configured in different zones.
- Administrators want low audit overhead. The global zone administrator audits all zones identically. Also, the global zone's audit daemon serves all zones on the system.

Sites that satisfy the following conditions can run one audit service per zone:

- The site does not require a single-image audit trail.
- The non-global zones have customized naming service files. These separate administrative domains typically function as servers.
- Individual zone administrators want to control auditing in the zones that they administer. In per-zone auditing, zone administrators can decide to enable or to disable auditing for the zone that they administer.

The advantages of per-zone auditing are a customized audit trail for each zone, and the ability to disable auditing on a zone-by-zone basis. These advantages can be offset by the administrative overhead. Each zone administrator must administer auditing. Each zone runs its own audit daemon, and has its own audit queue and audit logs. These audit logs must be managed.

♦ ♦ ♦ CHAPTER 2

Planning for Auditing

This chapter describes how to plan the customization of the audit service for your Oracle Solaris installation:

For an overview of auditing, see Chapter 1, "About Auditing in Oracle Solaris". For procedures to configure auditing at your site, see the following chapters:

- Chapter 3, "Managing the Audit Service"
- Chapter 4, "Monitoring System Activities"
- Chapter 5, "Working With Audit Data"
- Chapter 6, "Analyzing and Resolving Audit Service Issues"

For reference information, see Chapter 7, "Auditing Reference".

Concepts in Planning Auditing

You want to be selective about what kinds of activities are audited. At the same time, you want to collect useful audit information. You also need to carefully plan who to audit and what to audit. If you are using the default `audit_binfile` plugin, audit files can quickly grow to fill the available space, so you must allocate enough disk space.

Planning a Single System Audit Trail

Note - Implementing a single system audit trail applies only to the `audit_binfile` plugin.

Systems within a single administrative domain can create a single-system image audit trail.

To create a single-system image audit trail for a site, follow these requirements:

- Use the same naming service for all systems.

  For correct interpretation of the audit records, the `passwd`, `group`, and `hosts` files must be consistent.
- Configure the audit service identically on all systems. For information about displaying and modifying the service settings, see the `auditconfig`(1M) man page.
- Use the same `audit_warn`, `audit_event`, and `audit_class` files for all systems.

Refer to "How to Plan Who and What to Audit" on page 28 for additional considerations for enabling auditing on the systems.

Planning Auditing in Zones

If your system contains non-global zones, the zones can be audited as the global zone is audited, or the audit service for each non-global zone can be configured, enabled, and disabled separately. For example, you could audit only the non-global zones and not audit the global zone.

For a discussion of the trade-offs, see "Auditing on a System With Oracle Solaris Zones" on page 24.

The following options are available when implementing auditing in zones.

Implementing One Audit Service for All Zones

Auditing all zones identically can create a single-image audit trail. A single-image audit trail occurs when you are using the `audit_binfile` or the `audit_remote` plugin, and all zones on a system are part of one administrative domain. The audit records can then be easily compared because the records in every zone are preselected with identical settings.

This configuration treats all zones as part of one system. The global zone runs the only audit service on a system and collects audit records for every zone. You customize the `audit_class` and `audit_event` files only in the global zone, then copy these files to every non-global zone.

Use the following guidelines when configuring a single audit service for all the zones.

- Use the same naming service for every zone.

Note - If naming service files are customized in non-global zones, and `perzone` policy is not set, then careful use of the audit tools is required to select usable records. A user ID in one zone can refer to a different user from the same ID in a different zone.

- Enable the audit records to include the name of the zone.

  To put the zone name as part of the audit record, set the `zonename` policy in the global zone. The `auditreduce` command can then select audit events by zone from the audit trail. For an example, see the `auditreduce`(1M) man page.

To plan a single-image audit trail, refer to "How to Plan Who and What to Audit" on page 28. Start with the first step. The global zone administrator must also set aside storage, as described in "How to Plan Disk Space for Audit Records" on page 30.

Implementing One Audit Service Per Zone

Choose to configure per-zone auditing if different zones use different naming service databases, or if zone administrators want to control auditing in their zones.

Note - To audit non-global zones, the `perzone` policy must be set but the audit service does not have to be enabled in the global zone. Non-global zone auditing is configured and its audit service is enabled and disabled separately from the global zone.

- When you configure per-zone auditing, you set the `perzone` audit policy in the global zone. If per-zone auditing is set before a non-global zone is first booted, auditing begins at the zone's first boot. To set audit policy, see "How to Configure Per-Zone Auditing" on page 68.
- Each zone administrator configures auditing for the zone.

  A non-global zone administrator can set all policy options except `perzone` and `ahlt`.
- Each zone administrator can enable or disable auditing in the zone.
- To generate records that can be traced to their originating zone during review, set the `zonename` audit policy.

Note - In per zone auditing, if the `audit_binfile` plugin is active, each zone administrator must also set aside storage for every zone, as described in "How to Plan Disk Space for Audit Records" on page 30. For additional planning instructions, see "How to Plan Who and What to Audit" on page 28.

Planning Auditing

The following task map points to the major tasks that are required for planning disk space and which events to record.

TABLE 2-1 Planning Auditing Task Map

Task	For Instructions
Determine who and what to audit	"How to Plan Who and What to Audit" on page 28
Plan storage space for the audit trail	"How to Plan Disk Space for Audit Records" on page 30
Plan transmission of the audit trail to a remote server	"How to Prepare to Stream Audit Records to Remote Storage" on page 32

▼ How to Plan Who and What to Audit

Before You Begin If you are implementing non-global zones, review "Planning Auditing in Zones" on page 26 before using this procedure.

1. **Determine the audit policy.**

   By default, only the `cnt` policy is enabled.

   Use the `auditconfig -lspolicy` command to see a description of available policy options.

   - For the effects of the policy options, see "Understanding Audit Policy" on page 33.
   - For the effect of the `cnt` policy, see "Audit Policies for Asynchronous and Synchronous Events" on page 118.
   - To set audit policy, see "How to Change Audit Policy" on page 49.

2. **Determine whether you want to modify event-to-class mappings.**

   In almost all situations, the default mapping is sufficient. However, if you add new classes, change class definitions, or determine that a record of a specific system call is not useful, you might want to modify event-to-class mappings.

   For an example, see "How to Change an Audit Event's Class Membership" on page 55.

3. **Determine which audit classes to preselect.**

   The best time to add audit classes or to change the default classes is before users log in to the system.

The audit classes that you preselect with the `-setflags` and `-setnaflags` options to the `auditconfig` command apply to all users and processes. You can preselect a class for success, for failure, or for both.

For the list of audit classes, read the `/etc/security/audit_class` file.

4. **Determine user modifications to the system-wide preselections.**

   If you decide that some users should be audited differently from the system, you can modify the `audit_flags` security attribute for individual users or for a rights profile. The user preselection mask is modified for users whose audit flags are explicitly set or who are assigned a rights profile with explicit audit flags.

   For the procedure, see "How to Configure a User's Audit Characteristics" on page 45. For which audit flag values are in effect, see "Order of Search for Assigned Rights" in "Securing Users and Processes in Oracle Solaris 11.2 ".

5. **Decide how to manage the `audit_warn` email alias.**

   The `audit_warn` script is run whenever the audit system detects a situation that requires administrative attention. By default, the `audit_warn` script sends email to an `audit_warn` alias and sends a message to the console.

   To set up the alias, see "How to Configure the `audit_warn` Email Alias" on page 53.

6. **Decide in which format and where to collect audit records.**

   You have three choices.

   - By default, store binary audit records locally. The default storage directory is `/var/audit`. To further configure the `audit_binfile` plugin, see "How to Create ZFS File Systems for Audit Files" on page 74.
   - Stream binary audit records to a remote protected repository by using the `audit_remote` plugin. You must have a receiver for the records. For the requirements, see "Managing a Remote Repository" on page 19. For the procedure, see "How to Send Audit Files to a Remote Repository" on page 81.
   - Send audit record summaries to `syslog` by using the `audit_syslog` plugin. For the procedure, see "How to Configure `syslog` Audit Logs" on page 86.

     For a comparison of binary and `syslog` formats, see "Audit Logs" on page 16.

7. **Determine when to warn the administrator about shrinking disk space.**

Note - This step applies only to the `audit_binfile` plugin.

When disk space on an audit file system drops below the minimum free space percentage, or soft limit, the audit service switches to the next available audit directory. The service then sends a warning that the soft limit has been exceeded.

To see how to set a minimum free space percentage, see Example 4-7.

8. **Decide what action to take when all the audit directories are full.**

Note - This step applies only to the `audit_binfile` plugin.

In the default configuration, the `audit_binfile` plugin is active and the `cnt` policy is set. In this configuration, when the kernel audit queue is full, the system continues to work. The system counts the audit records that are dropped but does not record the events. For greater security, you can disable the `cnt` policy and enable the `ahlt` policy. The `ahlt` policy stops the system when an asynchronous event cannot be placed in the audit queue.

However, if the `audit_binfile` queue is full, and the queue for another active plugin is not full, then the kernel queue will continue to send records to the plugin that is not full. When the `audit_binfile` queue can again accept records, the audit service will resume sending records to it.

For a discussion of the `cnt` and `ahlt` policy options, see "Audit Policies for Asynchronous and Synchronous Events" on page 118. To see how to configure these policy options, see Example 3-10.

Note - The `cnt` or `ahlt` policy is not triggered if the queue for at least one plugin is accepting audit records.

Planning Disk Space for Audit Records

The `audit_binfile` plugin creates an audit trail. The audit trail requires dedicated file space. This space must be available and secure. The system uses the `/var/audit` file system for initial storage. You can configure additional audit file systems for audit files. The following procedure covers the issues that you must resolve when you plan for audit trail storage.

▼ How to Plan Disk Space for Audit Records

Before You Begin If you are implementing non-global zones, complete "Planning Auditing in Zones" on page 26 before using this procedure.

This procedure assumes that you are using the `audit_binfile` plugin.

1. **Determine how much auditing your site needs.**

   Balance your site's security needs against the availability of disk space for the audit trail.

   For guidance on how to reduce space requirements while still maintaining site security, as well as how to design audit storage, see "Controlling Auditing Costs" on page 35 and "Auditing Efficiently" on page 36.

   For practical steps, see "Volume of Audit Records Is Large" on page 106, "How to Compress Audit Files on a Dedicated File System" on page 63, and Example 5-4.

2. **Determine which systems are to be audited and configure their audit file systems.**

   Create a list of all the file systems that you plan to use. For configuration guidelines, see "Storing and Managing the Audit Trail" on page 18 and the `auditreduce`(1M) man page. To specify the audit file systems, see "How to Assign Audit Space for the Audit Trail" on page 77.

3. **Synchronize the clocks on all systems.**

   For more information, see "Ensuring Reliable Time Stamps" on page 19.

Preparing to Stream Audit Records to Remote Storage

The `audit_remote` plugin sends the binary audit trail to an ARS in the same format as the `audit_binfile` plugin writes to the local audit files. The `audit_remote` plugin uses the `libgss` library to authenticate the ARS, and a GSS-API mechanism to protect the transmission with privacy and integrity. For reference, see "What Is the Kerberos Service?" in "Managing Kerberos and Other Authentication Services in Oracle Solaris 11.2 " and "Kerberos Utilities" in "Managing Kerberos and Other Authentication Services in Oracle Solaris 11.2 ".

The only currently supported GSS-API mechanism is `kerberosv5`. For more information, see the `mech`(4) man page.

▼ How to Prepare to Stream Audit Records to Remote Storage

Note - If you have a Kerberos realm configured with an identified Audit Remote Server (ARS) and all audited systems within the realm, you can skip this procedure. The steps to configure the ARS and the audited systems are covered in "How to Configure a Remote Repository for Audit Files" on page 82 and "How to Send Audit Files to a Remote Repository" on page 81.

To verify whether a Kerberos realm is configured, issue the following command. The sample output indicates that Kerberos is not installed on the system.

```
# pkg info system/security/kerberos-5
pkg: info: no packages matching these patterns are installed on the system.
```

Before You Begin This procedure assumes that you are using the `audit_remote` plugin.

1. **Install the master KDC (Key Distribution Center) package.**

   You can use the system that will serve as the ARS, or you can use a nearby system. The ARS sends a significant amount of authentication traffic to the master KDC.

   ```
   # pkg install pkg:/system/security/kerberos-5
   ```

   On the master KDC, you use the Kerberos `kdcmgr` and `kadmin` commands to manage the realm. For more information, see the `kdcmgr`(1M) and `kadmin`(1M) man pages.

2. **On every audited system that will send audit records to the ARS, install the master KDC package.**

   ```
   # pkg install pkg:/system/security/kerberos-5
   ```

   This package includes the `kclient` command. On these systems, you run the `kclient` command to connect with the KDC. For more information, see the `kclient`(1M) man page.

3. **Synchronize the clocks in the KDC realm.**

   If the clock skew is too big between the audited systems and the ARS, the attempt at connection will fail. After a connection is established, the local time on the ARS determines the names of the stored audit files, as described in "Conventions for Binary Audit File Names" on page 120.

   For more information about the clocks, see "Ensuring Reliable Time Stamps" on page 19.

Understanding Audit Policy

Audit policy determines the characteristics of the audit records for the local system. You use the `auditconfig` command to set these policies. For more information, see the `auditconfig`(1M) man page.

Most audit policy options are disabled by default to minimize storage requirements and system processing demands. These options are properties of the audit service and determine the policies that are in effect at system boot. For more information, see the `auditconfig`(1M) man page.

Use the following table to determine whether the needs of your site justify the additional overhead that results from enabling one or more audit policy options.

TABLE 2-2 Effects of Audit Policy Options

Policy Name	Description	Policy Considerations
`ahlt`	This policy applies to asynchronous events only. When disabled, this policy allows the event to complete without an audit record being generated. When enabled, this policy stops the system when the audit queue is full. Administrative intervention is required to clean up the audit queue, make space available for audit records, and reboot. This policy can be enabled only in the global zone. The policy affects all zones.	The disabled option is preferable when system availability is more important than security. The enabled option is preferable in an environment where security is paramount. For a fuller discussion, see "Audit Policies for Asynchronous and Synchronous Events" on page 118.
`arge`	When disabled, this policy omits environment variables of an executed program from the `execve` audit record. When enabled, this policy adds the environment variables of an executed program to the `execve` audit record. The resulting audit records contain much more detail than when this policy is disabled.	The disabled option collects much less information than the enabled option. For a comparison, see "How to Audit All Commands by Users" on page 57. The enabled option is preferable when you are auditing a few users. The option is also useful when you are unsure about the environment variables that are being used in programs in the `ex` audit class.
`argv`	When disabled, this policy omits the arguments of an executed program from the `execve` audit record. When enabled, this policy adds the arguments of an executed program to the `execve` audit record. The resulting audit records contain much more detail than when this policy is disabled.	The disabled option collects much less information than the enabled option. For a comparison, see "How to Audit All Commands by Users" on page 57. The enabled option is preferable when you are auditing a few users. The option is also useful when you have reason to believe that unusual programs in the `ex` audit class are being run.
`cnt`	When disabled, this policy blocks a user or application from running. The blocking happens when audit records cannot be added to the audit trail because the audit queue is full.	The disabled option is preferable in an environment where security is paramount.

Policy Name	Description	Policy Considerations
	When enabled, this policy allows the event to complete without an audit record being generated. The policy maintains a count of audit records that are dropped.	The enabled option is preferable when system availability is more important than security. For a fuller discussion, see "Audit Policies for Asynchronous and Synchronous Events" on page 118.
`group`	When disabled, this policy does not add a groups list to audit records. When enabled, this policy adds a groups list to every audit record as a special token.	The disabled option usually satisfies requirements for site security. The enabled option is preferable when you need to audit the supplemental groups to which the subject belongs to.
`path`	When disabled, this policy records in an audit record at most one path that is used during a system call. When enabled, this policy records every path that is used in conjunction with an audit event to every audit record.	The disabled option places at most one path in an audit record. The enabled option enters each file name or path that is used during a system call in the audit record as a `path` token.
`perzone`	When disabled, this policy maintains a single audit configuration for a system. One audit service runs in the global zone. Audit events in specific zones can be located in the audit record if the `zonename` audit token was preselected. When enabled, this policy maintains a separate audit configuration, audit queue, and audit logs for each zone. An audit service runs in each zone. This policy can be enabled in the global zone only.	The disabled option is useful when you have no special reason to maintain a separate audit log, queue, and daemon for each zone. The enabled option is useful when you cannot monitor your system effectively by simply examining audit records with the `zonename` audit token.
`public`	When disabled, this policy does not add read-only events of public objects to the audit trail when the reading of files is preselected. Audit classes that contain read-only events include `fr`, `fa`, and `cl`. When enabled, this policy records every read-only audit event of public objects if an appropriate audit class is preselected.	The disabled option usually satisfies requirements for site security. The enabled option is rarely useful.
`seq`	When disabled, this policy does not add a sequence number to every audit record. When enabled, this policy adds a sequence number to every audit record. The `sequence` token holds the sequence number.	The disabled option is sufficient when auditing is running smoothly. The enabled option is preferable when the `cnt` policy is enabled. The `seq` policy enables you to determine when data was discarded. Alternatively, you can use the `auditstat` command to view dropped records.
`trail`	When disabled, this policy does not add a `trailer` token to audit records. When enabled, this policy adds a `trailer` token to every audit record.	The disabled option creates a smaller audit record. The enabled option clearly marks the end of each audit record with a `trailer` token. The `trailer` token is often used with the `sequence` token. The

Policy Name	Description	Policy Considerations
		`trailer` token aids in the recovery of damaged audit trails.
`zonename`	When disabled, this policy does not include a `zonename` token in audit records.	The disabled option is useful when you do not need to track audit behavior per zone.
	When enabled, this policy includes a `zonename` token in every audit record.	The enabled option is useful when you want to isolate and compare audit behavior across zones by post-selecting records according to zone.

Controlling Auditing Costs

Because auditing consumes system resources, you must control the degree of detail that is recorded. When you decide what to audit, consider the following costs of auditing:

- Cost of increased processing time
- Cost of analysis of audit data

If you are using the default plugin, `audit_binfile`, you must also consider the storage cost of audit data.

Cost of Increased Processing Time of Audit Data

The cost of increased processing time is the least significant of the costs of auditing. Auditing generally does not occur during computation-intensive tasks, such as image processing, complex calculations, and so forth. Also, if you are using the `audit_binfile` plugin, audit administrators can move the post-selection tasks from the audited system to systems that are dedicated to analyzing audit data. Finally, unless kernel events are preselected, the audit service has no measurable impact on system performance.

Cost of Analysis of Audit Data

The cost of analysis is roughly proportional to the amount of audit data that is collected. The cost of analysis includes the time that is required to merge and review audit records.

For records that are collected by the `audit_binfile` plugin, cost also includes the time that is required to archive the records and their supporting name service databases, and to keep the records in a safe place. Supporting databases include `groups`, `hosts`, and `passwd`.

The fewer records that you generate, the less time that is required to analyze the audit trail. The sections “Cost of Storage of Audit Data” on page 36 and “Auditing

Efficiently" on page 36 describe ways to audit efficiently. Efficient auditing reduces the amount of audit data while still providing enough coverage to achieve your site's security goals.

Cost of Storage of Audit Data

If you are using the `audit_binfile` plugin, storage cost is the most significant cost of auditing. The amount of audit data depends on the following:

- Number of users
- Number of systems
- Amount of use
- Degree of traceability and accountability that is required

Because these factors vary from site to site, no formula can predetermine the amount of disk space to set aside for audit data storage. Use the following information as a guide:

- Understand the audit classes

  Before you configure auditing, you should understand the types of events that the classes contain. You can change the audit event-class mappings to optimize audit record collection.
- Preselect audit classes judiciously to reduce the volume of records that are generated.

  Full auditing, that is, with the `all` class, fills disk space quickly. Even a simple task such as compiling a program could generate a large audit file. A program of modest size could generate thousands of audit records in less than a minute.

  For example, by omitting the `file_read` audit class, `fr`, you can significantly reduce audit volume. By choosing to audit for failed operations only, you can at times reduce audit volume. For example, by auditing for failed `file_read` operations, `-fr`, you can generate far fewer records than by auditing for all `file_read` events.
- If you are using the `audit_binfile` plugin, efficient audit file management is also important. For example, you can compress a ZFS file system that is dedicated to audit files.
- Develop a philosophy of auditing for your site.

  Base your philosophy on measures such as the amount of traceability that your site requires, and the types of users that you administer.

Auditing Efficiently

The following techniques can help you achieve your organization's security goals while auditing more efficiently.

- For as many audit classes as possible, preselect those classes only for users and roles, not system-wide.
- Randomly audit only a certain percentage of users at any one time.
- If the `audit_binfile` plugin is active, reduce the disk storage requirements for audit files by filtering, merging, and compressing the files. Develop procedures for archiving the files, for transferring the files to removable media, and for storing the files offline.
- Monitor the audit data in real time for unusual behaviors.
  - `audit_syslog` plugin – You can extend management and analysis tools that you have already developed to handle the audit records in `syslog` files.
  - `audit_binfile` plugin – You can set up procedures to monitor the audit trail for certain activities. You can write a script to trigger an automatic increase in the auditing of certain users or certain systems in response to detection of unusual events.

    For example, you could write a script that does the following:

    1. Monitors the creation of audit files on the audited systems.
    2. Processes the audit files with the `tail` command.

       The piping of the output from the `tail -0f` command through the `praudit` command can yield a stream of audit records as the records are generated. For more information, see the `tail`(1) man page.
    3. Analyzes this stream for unusual message types or other indicators, and delivers the analysis to the auditor.

       Alternatively, the script can be used to trigger automatic responses.
    4. Constantly monitors the audit file systems for the appearance of new `not_terminated` audit files.
    5. Terminates outstanding `tail` processes when their files are no longer being written to.

♦ ♦ ♦ CHAPTER 3

Managing the Audit Service

This chapter provides procedures to help you configure and manage auditing on an Oracle Solaris system. The chapter covers the following tasks:

In addition, the following chapters describe other audit management tasks:

For an overview of the audit service, see Chapter 1, "About Auditing in Oracle Solaris". For planning suggestions, see Chapter 2, "Planning for Auditing". For reference information, see Chapter 7, "Auditing Reference".

Default Configuration of the Audit Service

The audit service has a default configuration and is immediately operational on the global zone after you install Oracle Solaris 11.2. No additional action is required to enable or configure the service to become usable. With its default configuration, the audit service records the following operations:

- Login and logout operations
- Use of the `su` command
- Screen lock and screen unlock operations

Because the service's default configuration has no performance impact on the system, disabling the service on performance grounds is not required.

Provided that you have the appropriate audit-related rights, such as those in the Audit Review Rights profile, you can review the audit logs. The logs are stored in `/var/audit/`*hostname*. You view these files by using the `praudit` and `auditreduce` commands. For more information, see "Displaying Audit Trail Data" on page 89.

The subsequent sections in this chapter provide instructions for customizing the audit service configuration, if the default configuration is insufficient for your needs.

Displaying Audit Service Defaults

The audit service is regulated by the following parameters:

- Classes of attributable and non-attributable events
- Audit policy
- Audit plugins
- Queue controls

To display the audit service defaults, you typically use `auditconfig -get*` subcommand. This subcommand displays the current configuration of the parameter that is represented by the asterisk (*), such as `-getflags` `-getpolicy`, or `-getqctrl`. To display information about classes for non-attributable events, use the `auditconfig -getnaflags` subcommand.

For more information about the `auditconfig` command, see the `auditconfig`(1M) man page.

Note - To display the audit service configuration, you must become an administrator who is assigned the Audit Configuration or Audit Control rights profile. For more information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2 ".

The following examples show the appropriate command syntax to use to display the default audit configuration settings.

EXAMPLE 3-1 Displaying Default Class for Events

In this example, two subcommands are used to display the preselected classes for attributable and non-attributable events respectively. To see which events are assigned to a class, and therefore which events are being recorded, run the `auditrecord -c` *class* command.

```
# auditconfig -getflags
```

```
active user default audit flags = lo(0x1000,0x1000)
configured user default audit flags = lo(0x1000,0x1000)
```

`lo` is the flag for the `login/logout` audit class. The format of the mask output is (*success*,*failure*).

```
# auditconfig -getnaflags
active non-attributable audit flags = lo(0x1000,0x1000)
configured non-attributable audit flags = lo(0x1000,0x1000)
```

EXAMPLE 3-2 Displaying the Default Audit Policy

```
$ auditconfig -getpolicy
configured audit policies = cnt
active audit policies = cnt
```

The *active* policy is the current policy, but the policy value is not being stored by the audit service. The *configured* policy is stored by the audit service, so the policy is restored when you restart the audit service.

EXAMPLE 3-3 Displaying the Default Audit Plugins

```
$ auditconfig -getplugin
Plugin: audit_binfile
Attributes: p_dir=/var/audit;p_fsize=0;p_minfree=1;

Plugin: audit_syslog (inactive)
Attributes: p_flags=;

Plugin: audit_remote (inactive)
Attributes: p_hosts=;p_retries=3;p_timeout=5;
```

The `audit_binfile` plugin is active by default.

EXAMPLE 3-4 Displaying the Audit Queue Controls

```
$ auditconfig -getqctrl
no configured audit queue hiwater mark
no configured audit queue lowater mark
no configured audit queue buffer size
no configured audit queue delay
active audit queue hiwater mark (records) = 100
active audit queue lowater mark (records) = 10
active audit queue buffer size (bytes) = 8192
active audit queue delay (ticks) = 20
```

The *active* queue control is the queue control that is currently used by the kernel. The string `no configured` indicates that the system is using the default values.

Enabling and Disabling the Audit Service

The audit service is enabled by default. If the `perzone` audit policy is set, zone administrators must enable, refresh, or disable the audit service in each non-global zone as desired. If the `perzone` audit policy is not set, enabling, refreshing, or disabling the audit service from the global zone is effective for all non-global zones.

To disable or enable the audit service, you must become an administrator who is assigned the Audit Control rights profile. For more information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2 ".

To disable the audit service, use the following command:

```
# audit -t
```

To enable the audit service, use the following command:

```
# audit -s
```

To verify that the audit service is running, use the following command:

```
# auditconfig -getcond
audit condition = auditing
```

If the `perzone` audit policy is set, then you must perform this verification in the non-global zones where you enabled auditing.

For more information, see the `audit`(1M) and `auditd`(1M) man pages.

Configuring the Audit Service

Before you enable auditing on your network, you can modify the defaults to satisfy your site auditing requirements. Best practice is to customize your audit configuration as much as possible before the first users log in.

If you have implemented zones, you can choose to audit all zones from the global zone or to audit non-global zones individually. For an overview, see "Auditing and Oracle Solaris Zones" on page 114. For planning, see "Planning Auditing in Zones" on page 26. For procedures, see "Configuring the Audit Service in Zones" on page 65.

To configure the audit service, you typically use `auditconfig` subcommands. The configuration that is set with these subcommands applies to the whole system.

- `auditconfig -get*` displays the current configuration of the parameter that is represented by the asterisk (*), as shown in the examples of "Displaying Audit Service Defaults" on page 40.

- `auditconfig -set*` assigns a value to the parameter that is represented by the asterisk (*), such as `-setflags`, `-setpolicy`, or `-setqctrl`. To configure classes for non-attributable events, you use the `auditconfig setnaflags` subcommand.

You can also customize auditing to apply to users or profiles, rather than to the entire system. Audit class preselections for each user are specified by the `audit_flags` security attribute. These user-specific values, plus the preselected classes for the system, determine the user's audit mask, as described in "Process Audit Characteristics" on page 119.

By preselecting classes on a per user basis rather than on a per system basis, you can sometimes reduce the impact of auditing on system performance. Also, you might want to audit specific users slightly differently from the system.

To configure auditing that applies to users or profiles, you use the following commands:

- `usrattr` displays the `audit_flags` value that is set for users. By default, users are audited for the system-wide settings only.
- `usermod -K` sets flags that apply to users.
- `profile` sets flags that apply to profiles.

For a description of the `userattr` command, see the `userattr`(1) man page. For a description of the `audit_flags` keyword, see the `user_attr`(4) man page.

The following task map points to the procedures for configuring auditing. All tasks are optional.

TABLE 3-1 Configuring the Audit Service Task Map

Task	Description	For Instructions
Select which events are audited.	Preselects system-wide audit classes. If an event is attributable, then all users are audited for this event.	"How to Preselect Audit Classes" on page 44
Select which events are audited for specific users.	Sets user-specific differences from the system-wide audit classes.	"How to Configure a User's Audit Characteristics" on page 45
Specify audit policy.	Defines additional audit data that your site requires.	"How to Change Audit Policy" on page 49
Specify queue controls.	Modifies the default buffer size, audit records in the queue, and interval between writing audit records to the buffer.	"How to Change Audit Queue Controls" on page 51
Create the `audit_warn` email alias.	Defines who receives email warnings when the audit service needs attention.	"How to Configure the `audit_warn` Email Alias" on page 53
Configure audit logs.	Configures the location of audit records for each plugin.	"Configuring Audit Logs" on page 73
Add audit classes.	Reduces the number of audit records by creating a new audit class to hold critical events.	"How to Add an Audit Class" on page 54

Task	Description	For Instructions
Change event-to-class mappings.	Reduces the number of audit records by changing the event-class mapping.	"How to Change an Audit Event's Class Membership" on page 55

How to Preselect Audit Classes

Preselect audit classes that contain the events that you want to monitor. Events that are not in preselected classes are not recorded.

Before You Begin You must become an administrator who is assigned the Audit Configuration rights profile. For more information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2 ".

1. Determine the current preselected classes.

```
# auditconfig -getflags
...

# auditconfig -getnaflags
,,,
```

For an explanation of the output, see "Displaying Audit Service Defaults" on page 40.

2. Preselect the attributable classes.

```
# auditconfig -setflags lo,ps,fw
user default audit flags = ps,lo,fw(0x101002,0x101002)
```

This command audits the events in the `login/logout`, `process start/stop`, and `file write` classes for success and for failure.

Note - The `auditconfig -setflags` command *replaces* the current preselection, so you must specify all classes that you want to preselect.

---

3. Preselect the non-attributable classes.

The `na` class contains PROM, boot, and non-attributable mounts, among other events.

```
# auditconfig -setnaflags lo,na
non-attributable audit flags = lo,na(0x1400,0x1400)
```

`lo` and `na` are the only useful arguments to the `-setnaflags` option.

Note - The `auditconfig -setnaflags` command *replaces* the current preselection, so you must specify all classes that you want to preselect.

▼ How to Configure a User's Audit Characteristics

These user-specific audit charactersitics that you set with this procedure are combined with the preselected classes for the system. Together they determine the user's audit mask, as described in "Process Audit Characteristics" on page 119.

Before You Begin You must assume the `root` role. For more information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2 ".

1. (Optional) Display the audit classes that are currently selected for existing users.

a. Display the list of users.

```
# who
adoe    pts/1        Oct 10 10:20 (:0.0)
adoe    pts/2        Oct 10 10:20 (:0.0)
jdoe    pts/5        Oct 12 12:20 (:0.0)
jdoe    pts/6        Oct 12 12:20 (:0.0)
...
```

b. Display each user's `audit_flags` attribute value.

```
# userattr audit_flags adoe
# userattr audit_flags jdoe
```

2. Set the audit flags in the `user_attr` or in the `prof_attr` database.

For example, you can create a rights profile that defines the rights of a subset of your users. Users who are assigned that rights profile are audited identically.

- **To set audit flags for a user, use the `usermod` command.**

```
# usermod -K audit_flags=fw:no jdoe
```

The format of the `audit_flags` keyword is *always-audit*:*never-audit*.

always-audit Lists the audit classes that are audited for this user. Modifications to the system-wide classes are prefixed by a caret (^). Classes that are added to the system-wide classes are not prefixed by a caret.

never-audit Lists the audit classes that are never audited for the user, even if these audit events are audited system-wide. Modifications to the system-wide classes are prefixed by a caret (^).

To specify multiple audit classes, separate the classes with commas. For more information, see the `audit_flags`(5) man page.

- **To set audit flags for a rights profile, use the `profiles` command.**

```
# profiles -p "System Administrator"
profiles:System Administrator> set name="Audited System Administrator"
profiles:Audited System Administrator> set always_audit=fw,as
profiles:Audited System Administrator> end
profiles:Audited System Administrator> exit
```

When you assign the Audited System Administrator rights profile to a user or a role, that user or role is audited for those flags, subject to search order as described in "Order of Search for Assigned Rights" in "Securing Users and Processes in Oracle Solaris 11.2 ".

Example 3-5 Changing Which Events Are Audited for One User

In this example, the audit preselection mask for all users is the following:

```
# auditconfig -getflags
active user default audit flags = ss,lo(0x11000,0x11000)
configured user default audit flags = ss,lo(0x11000,0x11000)
```

No user except the administrator is logged in.

To lessen the impact of the `AUE_PFEXEC` audit event on system resources, the administrator does not audit this event at the system level. Rather, the administrator preselects the `pf` class for a user, `jdoe`. The `pf` class is created in Example 3-15.

```
# usermod -K audit_flags=pf:no jdoe
```

The `userattr` command shows the addition.

```
# userattr audit_flags jdoe
pf:no
```

When the user `jdoe` logs in, `jdoe`'s audit preselection mask is a combination of the `audit_flags` values with the system default values. `289` is the PID of `jdoe`'s login shell.

```
# auditconfig -getpinfo 289
audit id = jdoe(1234)
process preselection mask = ss,pf,lo(0x0100000008011000,0x0100000008011000)
terminal id (maj,min,host) = 242,511,example1(192.168.160.171)
audit session id = 103203403
```

Example 3-6 Modifying Audit Preselection Exception for One User

In this example, the audit preselection mask for all users is the following:

```
# auditconfig -getflags
active user default audit flags = ss,lo(0x11000,0x11000)
configured user default audit flags = ss,lo(0x11000,0x11000)
```

No users except the administrator are logged in.

The administrator decides not to collect failed `ss` events for the `jdoe` user.

```
# usermod -K audit_flags=^-ss:no jdoe
```

The `userattr` command shows the exception.

```
# userattr audit_flags jdoe
^-ss:no
```

When the user `jdoe` logs in, `jdoe`'s audit preselection mask is a combination of the `audit_flags` values with the system default values. `289` is the PID of `jdoe`'s login shell.

```
# auditconfig -getpinfo 289
audit id = jdoe(1234)
process preselection mask = +ss,lo(0x11000,0x1000)
terminal id (maj,min,host) = 242,511,example1(192.168.160.171)
audit session id = 103203403
```

Example 3-7 Auditing Selected Users, No System-Wide Auditing

In this example, the login and role activities of four selected users are audited on the system. No audit classes are preselected for the system.

First, the administrator removes all system-wide flags.

```
# auditconfig -setflags no
user default audit flags = no(0x0,0x0)
```

Then, the administrator preselects two audit classes for the four users. The `pf` class is created in Example 3-15.

```
# usermod -K audit_flags=lo,pf:no jdoe
# usermod -K audit_flags=lo,pf:no kdoe
# usermod -K audit_flags=lo,pf:no pdoe
# usermod -K audit_flags=lo,pf:no zdoe
```

Then, the administrator preselects the `pf` class for the `root` role.

```
# userattr audit_flags root
# rolemod -K audit_flags=lo,pf:no root
```

```
# userattr audit_flags root
lo,pf:no
```

To continue to record unwarranted intrusion, the administrator does not change the auditing of non-attributable logins.

```
# auditconfig -getnaflags
active non-attributable audit flags = lo(0x1000,0x1000)
configured non-attributable audit flags = lo(0x1000,0x1000)
```

Example 3-8 Removing a User's Audit Flags

In the following example, the administrator removes all user-specific audit flags. Existing processes of users who are currently logged in continue to be audited.

The administrator runs the `usermod` command with the `audit_flags` keyword set to no value.

```
# usermod -K audit_flags= jdoe
# usermod -K audit_flags= kdoe
# usermod -K audit_flags= ldoe
```

Then, the administrator verifies the removal.

```
# userattr audit_flags jdoe
# userattr audit_flags kdoe
# userattr audit_flags ldoe
```

Example 3-9 Creating a Rights Profile for a Group of Users

The administrator wants all administrative rights profiles at the site to explicitly audit the `pf` class. For every rights profile that is going to be assigned, the administrator creates a site-specific version in LDAP that includes audit flags.

First, the administrator clones an existing rights profile, then changes the name and adds audit flags.

```
# profiles -p "Network Wifi Management" -S ldap
profiles: Network Wifi Management> set name="Wifi Management"
profiles: Wifi Management> set desc="Audited wifi management"
profiles: Wifi Management> set audit_always=pf
profiles: Wifi Management> exit
```

After repeating this procedure for every rights profile that is going to be used, the administrator lists the information in the Wifi Management profile.

```
# profiles -p "Wifi Management" -S ldap info
name=Wifi Management
desc=Audited wifi management
auths=solaris.network.wifi.config
help=RtNetWifiMngmnt.html
```

```
always_audit=pf
```

How to Change Audit Policy

You might change default audit policy to record detailed information about audited commands, to add a zone name to every record, or to satisfy other site security requirements.

Before You Begin You must become an administrator who is assigned the Audit Configuration rights profile. For more information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2 ".

1. View the current audit policy.

```
$ auditconfig -getpolicy
...
```

For an explanation of the output, see "Displaying Audit Service Defaults" on page 40.

2. View the available policy options.

```
$ auditconfig -lspolicy
policy string    description:
ahlt             halt machine if it can not record an async event
all              all policies for the zone
arge             include exec environment args in audit recs
argv             include exec command line args in audit recs
cnt              when no more space, drop recs and keep a cnt
group            include supplementary groups in audit recs
none             no policies
path             allow multiple paths per event
perzone          use a separate queue and auditd per zone
public           audit public files
seq              include a sequence number in audit recs
trail            include trailer token in audit recs
windata_down     include downgraded window information in audit recs
windata_up       include upgraded window information in audit recs
zonename         include zonename token in audit recs
```

Note - The `perzone` and `ahlt` policy options can be set only in the global zone. For the trade-offs to using a particular policy option, see "Understanding Audit Policy" on page 33.

---

3. Enable or disable selected audit policy options.

```
# auditconfig [ -t ] -setpolicy [prefix]policy[,policy...]
```

`-t` Optional. Creates a temporary, or *active*, policy. You might set a temporary policy for debugging or testing purposes.

A temporary policy is in effect until the audit service is refreshed, or until the policy is modified by the `auditconfig -setpolicy` command.

prefix — A *prefix* value of + adds the list of policies to the current policy. A *prefix* value of - removes the list of policies from the current policy. Without a prefix, the audit policy is reset. This option enables you to retain current audit policies.

policy — Selects the policy to be enabled or to be disabled.

Example 3-10 Setting the `ahlt` Audit Policy Option

In this example, strict site security requires the `ahlt` policy.

```
# auditconfig -setpolicy -cnt
# auditconfig -setpolicy +ahlt
```

The plus sign (+) before the `ahlt` policy adds the policy to current policy settings. Without the plus sign, the `ahlt` policy replaces all current audit policies.

Example 3-11 Setting a Temporary Audit Policy

In this example, the `ahlt` audit policy is configured. For debugging, the administrator adds the `trail` audit policy to the active policy (`+trail`) temporarily (`-t`). The `trail` policy aids in the recovery of damaged audit trails.

```
$ auditconfig -setpolicy ahlt
$ auditconfig -getpolicy
configured audit policies = ahlt
active audit policies = ahlt
$ auditconfig -t -setpolicy +trail
configured audit policies = ahlt
active audit policies = ahlt,trail
```

The administrator disables the `trail` policy when the debugging is completed.

```
$ auditconfig -setpolicy -trail
$ auditconfig -getpolicy
configured audit policies = ahlt
active audit policies = ahlt
```

Refreshing the audit service by running the `audit -s` command also removes this temporary policy, plus any other temporary values in the audit service. For examples of other temporary values, see “How to Change Audit Queue Controls” on page 51.

Example 3-12 Setting the `perzone` Audit Policy

In this example, the `perzone` audit policy is added to the existing policy in the global zone. The `perzone` policy setting is stored as a permanent property, so `perzone` policy is in effect during the session and when the audit service is restarted. For the zones, the policy is available at the next zone boot.

```
$ auditconfig -getpolicy
configured audit policies = cnt
active audit policies = cnt
$ auditconfig -setpolicy +perzone
$ auditconfig -getpolicy
configured audit policies = perzone,cnt
active audit policies = perzone,cnt
```

Example 3-13 Collecting Audit Records for External Auditors

In this example, the administrator is collecting audit records to satisfy external auditors' requirements. The administrator decides to use an Audit Remote Server (ARS) to collect information about administrative activities. The administrator also collects actions that cannot be attributed to a user, such as booting.

The administrator sets up ARS. In addition to auditing the `cusa` class, the administrator adds policies to the audit configuration.

```
# auditconfig -setflags cusa
user default audit flags = ex,xa,ua,as,ss,ap,lo,ft(0x80475080,0x80475080)
# auditconfig -setpolicy ahlt,argv,argeauditconfig # auditconfig -getpolicy
configured audit policies = ahlt,arge,argv
active audit policies = ahlt,arge,argv
# auditconfig -setnaflags lo,na
non-attributable audit flags = lo,na(0x1400,0x1400)
```

When the administrator enables the `audit_remote` plugin and refreshes the audit service, the records are collected.

▼ How to Change Audit Queue Controls

The audit service provides default values for audit queue parameters. You can inspect, permanently change, and temporarily change these values with the `auditconfig` command.

Before You Begin You must become an administrator who is assigned the Audit Configuration rights profile. For more information, see “Using Your Assigned Administrative Rights” in “Securing Users and Processes in Oracle Solaris 11.2 ”.

1. **View the current values of the audit queue controls.**

```
$ auditconfig -getqctrl
...
```

For an explanation of the output, see "Displaying Audit Service Defaults" on page 40.

2. Modify selected audit queue controls.

For examples and a description of the audit queue controls, see the `auditconfig`(1M) man page.

- To modify some or all audit queue controls, use the `-setqctrl` option.

  ```
  # auditconfig [ -t ] -setqctrl hiwater lowater bufsz interval
  ```

  The high water (`hiwater`) and low water (`lowater`) values indicate the points at which processes are respectively suspended and resume. The points are measured in terms of the number of undelivered audit records. The buffer size (`bufsz`) refers to the buffer size of the queue. `Interval` indicates the delay, measured in number of ticks, between generation of audit output.

  For example, set the *interval* value to `10` without setting the other controls.

  ```
  # auditconfig -setqctrl 0 0 0 10
  ```

- To modify a specific audit queue control, specify its option. The `-setqdelay` option is the equivalent of `-setqctrl 0 0 0` *interval*, as in **`auditconfig -setqdelay 10`**.

  ```
  # auditconfig [ -t ] -setqhiwater value
  # auditconfig [ -t ] -setqlowater value
  # auditconfig [ -t ] -setqbufsz value
  # auditconfig [ -t ] -setqdelay value
  ```

Example 3-14 Resetting an Audit Queue Control to the Default

The administrator sets all audit queue controls, then changes the *lowater* value in the repository back to the default.

```
# auditconfig -setqctrl 200 5 10216 10
# auditconfig -setqctrl 200 0 10216 10
configured audit queue hiwater mark (records) = 200
no configured audit queue lowater mark
configured audit queue buffer size (bytes) = 10216
configured audit queue delay (ticks) = 10
active audit queue hiwater mark (records) = 200
active audit queue lowater mark (records) = 5
active audit queue buffer size (bytes) = 10216
active audit queue delay (ticks) = 10
```

Later, the administrator sets the *lowater* value to the default for the current session.

```
# auditconfig -setqlowater 10
```

```
# auditconfig -getqlowater
configured audit queue lowater mark (records) = 10
active audit queue lowater mark (records) = 10
```

How to Configure the audit_warn Email Alias

The /etc/security/audit_warn script generates mail to notify the administrator of audit incidents that might need attention. You can customize the script and you can send the mail to an account other than root.

If the perzone policy is set, the non-global zone administrator must configure the audit_warn email alias in the non-global zone.

Before You Begin You must become an administrator who is assigned the solaris.admin.edit/etc/security/audit_warn authorization. By default, only the root role has this authorization. For more information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2 ".

- **Configure the audit_warn email alias.**

  Choose one of the following options:

  - Replace the audit_warn email alias with another email account in the audit_warn script.

    Change the audit_warn email alias in the ADDRESS line of the script to another address:

    ```
    #ADDRESS=audit_warn           # standard alias for audit alerts
    ADDRESS=audadmin               # role alias for audit alerts
    ```

Note - For information about the effects of modifying an audit configuration file, see "Audit Configuration Files and Packaging" on page 115.

  - Redirect the audit_warn email to another mail account.

    Add the audit_warn email alias to the appropriate mail aliases file. You could add the alias to the local /etc/mail/aliases file or to the mail_aliases database in the name space. The /etc/mail/aliases entry would resemble the following example if the root and audadmin email accounts were added as members of the audit_warn email alias:

    ```
    audit_warn: root,audadmin
    ```

    Then, run the newaliases command to rebuild the random access database for the aliases file.

    ```
    # newaliases
    ```

```
/etc/mail/aliases: 14 aliases, longest 10 bytes, 156 bytes total
```

How to Add an Audit Class

When you create your own audit class, you can place into it just those audit events that you want to audit for your site. This strategy can reduce the number of records that are collected and reduce noise in your audit trail.

When you add the class on one system, copy the change to all systems that are being audited. Best practice is to create audit classes before the first users log in.

For information about the effects of modifying an audit configuration file, see "Audit Configuration Files and Packaging" on page 115.

Tip - In Oracle Solaris you can create your own package that contains files and replace the Oracle Solaris packages with your site-customized files. When you set the `preserve` attribute to `true` in your package, the `pkg` subcommands, such as verify, fix, revert, and so on, will run relative to your packages. For more information, see the `pkg`(1) and `pkg`(5) man pages.

Before You Begin Choose free bits for your unique entry. Verify which bits are available for customer use in the `/etc/security/audit_class` file.

You must become an administrator who is assigned the `solaris.admin.edit/etc/security/audit_class` authorization. By default, only the `root` role has this authorization. For more information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2".

1. **(Optional) Save a backup copy of the `audit_class` file.**

   ```
   # cp /etc/security/audit_class /etc/security/audit_class.orig
   ```

2. **Add new entries to the `audit_class` file.**

   Each entry has the following format:

   `0x`*64bitnumber*:*flag*:*description*

   For a description of the fields, see the `audit_class`(4) man page. For the list of existing classes, read the `/etc/security/audit_class` file.

Example 3-15 Creating a New Audit Class

This example creates a class to hold administrative commands that are executed in a role. The added entry to the `audit_class` file is as follows:

```
0x0100000000000000:pf:profile command
```

The entry creates the new `pf` audit class. Example 3-16 shows how to populate the new audit class.

Troubleshooting If you have customized the `audit_class` file, make sure that any audit flags that are assigned directly to users or rights profiles are consistent with the new audit classes. Errors occur when an `audit_flags` value is not a subset of the `audit_class` file.

How to Change an Audit Event's Class Membership

You might want to change an audit event's class membership to reduce the size of an existing audit class, or to place the event in a class of its own.

Caution - Never comment out events in the `audit_event` file. This file is used by the `praudit` command to read binary audit files. Archived audit files might contain events that are listed in the file.

When you reconfigure audit event-class mappings on one system, copy the change to all systems that are being audited. Best practice is to change event-class mappings before the first users log in.

Note - For information about the effects of modifying an audit configuration file, see "Audit Configuration Files and Packaging" on page 115.

Tip - In Oracle Solaris you can create your own package that contains files and replace the Oracle Solaris packages with your site-customized files. When you set the `preserve` attribute to `true` in your package, the `pkg` subcommands, such as verify, fix, revert, and so on, will run relative to your packages. For more information, see the `pkg`(1) and `pkg`(5) man pages.

Before You Begin You must become an administrator who is assigned the `solaris.admin.edit/etc/security/audit_event` authorization. By default, only the `root` role has this authorization. For more

information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2".

1. **(Optional) Save a backup copy of the `audit_event` file.**

```
# cp /etc/security/audit_event /etc/security/audit_event.orig
```

2. **Change the class to which particular events belong by changing the *class-list* of the events.**

Each entry has the following format:

number:*name*:*description*:*class-list*

number	The audit event ID.
name	The name of the audit event.
description	Typically, the system call or executable that triggers the creation of an audit record.
class-list	A comma-separated list of audit classes.

Example 3-16 Mapping Existing Audit Events to a New Class

This example maps an existing audit event to the new class that was created in Example 3-15. By default, the `AUE_PFEXEC` audit event is mapped to several audit classes. By creating the new class, the administrator can audit `AUE_PFEXEC` events without auditing the events in the other classes.

```
# grep pf /etc/security/audit_class
0x0100000000000000:pf:profile command
# grep AUE_PFEXEC /etc/security/audit_event
116:AUE_PFEXEC:execve(2) with pfexec enabled:ps,ex,ua,as,cusa
# pfedit /etc/security/audit_event
#116:AUE_PFEXEC:execve(2) with pfexec enabled:ps,ex,ua,as,cusa
116:AUE_PFEXEC:execve(2) with pfexec enabled:pf
# auditconfig -setflags lo,pf
user default audit flags = pf,lo(0x0100000000001000,0x0100000000001000)
```

Customizing What Is Audited

The following task map points to procedures to configure auditing that is specific to your needs.

TABLE 3-2 Customizing Auditing Task Map

Task	Description	For Instructions
Audit everything that a user does on the system.	Audit one or more users for every command.	"How to Audit All Commands by Users" on page 57
Change the audit events that are being recorded and have the change affect existing sessions.	Update a user's preselection mask.	"How to Update the Preselection Mask of Logged In Users" on page 61
Locate modifications to particular files.	Audit file modifications, then use the `auditreduce` command to find particular files.	"How to Find Audit Records of Changes to Specific Files" on page 59
Use less file system space for audit files.	Use ZFS quotas and compression.	"How to Compress Audit Files on a Dedicated File System" on page 63
Remove audit events from the `audit_event` file.	Correctly update the `audit_event` file.	"How to Prevent the Auditing of Specific Events" on page 62

▼ How to Audit All Commands by Users

As part of site security policy, some sites require audit records of all commands that are run by the `root` account and administrative roles. Some sites can require audit records of all commands by all users. Additionally, sites can require that the command arguments and environment be recorded.

Before You Begin To preselect audit classes and set audit policy, you must become an administrator who is assigned the Audit Configuration rights profile. To assign audit flags to users, roles, and rights profiles, you must assume the `root` role.

1. Display user level event information for `lo` and `ex` classes.

The `ex` class audits all calls to the `exec` and `execve` functions.

The `lo` class audits logins, logouts, and screen locks. The following output lists all the events in the `ex` and `lo` classes.

```
% auditconfig -lsevent | grep " lo "
AUE_login                       6152 lo login - local
AUE_logout                      6153 lo logout
AUE_telnet                      6154 lo login - telnet
AUE_rlogin                      6155 lo login - rlogin
AUE_rshd                        6158 lo rsh access
AUE_su                          6159 lo su
AUE_rexecd                      6162 lo rexecd
AUE_passwd                      6163 lo passwd
AUE_rexd                        6164 lo rexd
AUE_ftpd                        6165 lo ftp access
```

```
AUE_ftpd_logout                6171 lo ftp logout
AUE_ssh                        6172 lo login - ssh
AUE_role_login                 6173 lo role login
AUE_newgrp_login               6212 lo newgrp login
AUE_admin_authenticate         6213 lo admin login
AUE_screenlock                 6221 lo screenlock - lock
AUE_screenunlock               6222 lo screenlock - unlock
AUE_zlogin                     6227 lo login - zlogin
AUE_su_logout                  6228 lo su logout
AUE_role_logout                6229 lo role logout
AUE_smbd_session               6244 lo smbd(1m) session setup
AUE_smbd_logoff                6245 lo smbd(1m) session logoff
AUE_ClientConnect              9101 lo client connection to x server
AUE_ClientDisconnect           9102 lo client disconn. from x server

% auditconfig -lsevent | egrep " ex |,ex |ex,"
AUE_EXECVE                       23 ex,ps execve(2)
```

2. **Audit the `lo` and `ex` classes.**

   - **To audit these classes for administrative roles, modify the roles' security attributes.**

     In the following example, `root` is a role. The site has created three roles, `sysadm`, `auditadm`, and `netadm`. All roles are audited for the success and failure of events in the `ex` and `lo` classes.

     ```
     # rolemod -K audit_flags=lo,ex:no root

     # rolemod -K audit_flags=lo,ex:no sysadm

     # rolemod -K audit_flags=lo,ex:no auditadm

     # rolemod -K audit_flags=lo,ex:no netadm
     ```

   - **To audit these classes for all users, set the system-wide flags.**

     ```
     # auditconfig -setflags lo,ex
     ```

     The output appears similar to the following:

     ```
     header,129,2,AUE_EXECVE,,mach1,2010-10-14 12:17:12.616 -07:00
     path,/usr/bin/ls
     attribute,100555,root,bin,21,320271,18446744073709551615
     subject,jdoe,root,root,root,root,2486,50036632,82 0 mach1
     return,success,0
     ```

3. **Specify additional information to be recorded about command use.**

   - **To record the arguments to commands, add the `argv` policy.**

     ```
     # auditconfig -setpolicy +argv
     ```

The `exec_args` token records the command arguments:

```
header,151,2,AUE_EXECVE,,mach1,2010-10-14 12:26:17.373 -07:00
path,/usr/bin/ls
attribute,100555,root,bin,21,320271,18446744073709551615
exec_args
,2,ls,/etc/security
subject,jdoe,root,root,root,root,2494,50036632,82 0 mach1
return,success,0
```

- **To record the environment in which the command is run, add the `arge` policy.**

```
# auditconfig -setpolicy +arge
```

The `exec_env` token records the command environment:

```
header,1460,2,AUE_EXECVE,,mach1,2010-10-14 12:29:39.679 -07:00
path,/usr/bin/ls
attribute,100555,root,bin,21,320271,18446744073709551615
exec_args,2,ls,/etc/security
exec_env
,49,MANPATH=/usr/share/man,USER=jdoe,GDM_KEYBOARD_LAYOUT=us,EDITOR=gedit,
LANG=en_US.UTF-8,GDM_LANG=en_US.UTF-8,PS1=#,GDMSESSION=gnome,SESSIONTYPE=1,SHLVL=2,
HOME=/home/jdoe,LOGNAME=jdoe,G_FILENAME_ENCODING=@locale,UTF-8,
PRINTER=example-dbl,...,_=/usr/bin/ls
subject,jdoe,root,root,root,root,2502,50036632,82 0 mach1
return,success,0
```

How to Find Audit Records of Changes to Specific Files

If your goal is to log file writes against a limited number of files, such as `/etc/passwd` and the files in the `/etc/default` directory, you can use the `auditreduce` command to locate the files.

Before You Begin

The `root` role can perform every task in this procedure.

If administrative rights are distributed in your organization, note the following:

- An administrator with the Audit Configuration rights profile can run the `auditconfig` command.
- An administrator with the Audit Review rights profile can run the `auditreduce` command.
- Only the `root` role can assign audit flags.

For more information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2".

1. **Perform one of the following steps to audit file changes.**

- Audit the `fw` class.

  Adding the `fw`class to the audit flags of a user or role generates fewer records than adding this class to the system-wide audit preselection mask. Perform one of the following steps:

  - Add the `fw` class to specific roles.

    ```
    # rolemod -K audit_flags=fw:no root
    # rolemod -K audit_flags=fw:no sysadm
    # rolemod -K audit_flags=fw:no auditadm
    # rolemod -K audit_flags=fw:no netadm
    ```

  - Add the `fw` class to the system-wide flags.

    ```
    # auditconfig -getflags
    active user default audit flags = lo(0x1000,0x1000)
    configured user default audit flags = lo(0x1000,0x1000)

    # auditconfig -setflags lo,fw
    user default audit flags = lo,fw(0x1002,0x1002)
    ```

- Audit successful file-writes.

  Auditing successes generates fewer records than auditing failures and successes. Perform one of the following steps:

  - Add the `+fw` flag to specific roles.

    ```
    # rolemod -K audit_flags=+fw:no root
    # rolemod -K audit_flags=+fw:no sysadm
    # rolemod -K audit_flags=+fw:no auditadm
    # rolemod -K audit_flags=+fw:no netadm
    ```

  - Add the `+fw` flag to the system-wide flags.

    ```
    # auditconfig -getflags
    active user default audit flags = lo(0x1000,0x1000)
    configured user default audit flags = lo(0x1000,0x1000)

    # auditconfig -setflags lo,+fw
    user default audit flags = lo,+fw(0x1002,0x1000)
    ```

2. **Obtain the audit records for specific files with the `auditreduce` command.**

   ```
   # auditreduce -o file=/etc/passwd,/etc/default -O filechg
   ```

   The `auditreduce` command searches the audit trail for all instances of the `file` argument. The command creates a binary file with the suffix `filechg` which contains all records that include the path of the files of interest. See the `auditreduce`(1M) man page for the syntax of the `-o file=` *pathname* option.

3. **Read the `filechg` file with the `praudit` command.**

```
# praudit *filechg
```

▼ How to Update the Preselection Mask of Logged In Users

This procedure describes how to audit users who are already logged in for changes to the system-wide audit preselection mask. You can accomplish this task typically instructing the users to log out and to log back in. Alternatively, in a role that is assigned the Process Management rights profile, you can manually terminate active sessions with the `kill` command. The new sessions will inherit the new preselection mask.

However, terminating user sessions could be impractical. As an alternative, you can use the `auditconfig` command to dynamically change each logged-in user's preselection mask.

The following procedure assumes that you changed the system-wide audit preselection mask from `lo` to `lo,ex` by running the following command:

```
# auditconfig -setflags lo,ex
```

Before You Begin You must become an administrator who is assigned the Audit Configuration rights profile. To terminate user sessions, you must become an administrator who is assigned the Process Management rights profile. For more information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2 ".

1. **List the regular users who are logged in and their process IDs.**

```
# who -a
jdoe  - vt/2         Jan 25 07:56  4:10   1597   (:0)
jdoe  + pts/1        Jan 25 10:10   .     1706   (:0.0)
...
jdoe  + pts/2        Jan 25 11:36  3:41   1706   (:0.0)
```

2. **For later comparison, display each user's preselection mask.**

```
# auditconfig -getpinfo 1706
audit id = jdoe(1234)
process preselection mask = lo(0x1000,0x1000)
terminal id (maj,min,host) = 9426,65559,mach1(192.168.123.234)
audit session id = 103203403
```

3. **Modify the appropriate preselection mask by running one or more of the following commands:**

```
# auditconfig -setpmask 1706 lo,ex              /* for this process */
```

```
# auditconfig -setumask jdoe lo,ex              /* for this user */

# auditconfig -setsmask 103203403 lo,ex         /* for this session */
```

4. **Verify that the preselection mask for the user has changed.**

   For example, check a process that existed before you changed the mask.

```
# auditconfig -getpinfo 1706
audit id = jdoe(1234)
process preselection mask = ex,lo(0x40001000,0x40001000)
terminal id (maj,min,host) = 9426,65559,mach1(192.168.123.234)
audit session id = 103203403
```

▼ How to Prevent the Auditing of Specific Events

For maintenance purposes, sometimes a site wants to prevent events from being audited.

Before You Begin You must assume the `root` role. For more information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2 ".

1. **Change the class of the event to the `no` class.**

   Note - For information about the effects of modifying an audit configuration file, see "Audit Configuration Files and Packaging" on page 115.

   For example, events 26 and 27 belong to the `pm` class.

```
## audit_event file
...
25:AUE_VFORK:vfork(2):ps
26:AUE_SETGROUPS:setgroups(2):pm
27:AUE_SETPGRP:setpgrp(2):pm
28:AUE_SWAPON:swapon(2):no
...
```

   Change these events to the `no` class.

```
## audit_event file
...
25:AUE_VFORK:vfork(2):ps
26:AUE_SETGROUPS:setgroups(2):no
27:AUE_SETPGRP:setpgrp(2):no
28:AUE_SWAPON:swapon(2):no
...
```

   If the `pm` class is currently being audited, existing sessions will still audit events 26 and 27. To stop these events from being audited, you must update the users' preselection masks

by following the instructions in "How to Update the Preselection Mask of Logged In Users" on page 61.

Caution - Never comment out events in the `audit_event` file. This file is used by the `praudit` command to read binary audit files. Archived audit files might contain events that are listed in the file.

2. **Refresh the kernel events.**

```
# auditconfig -conf
Configured 283 kernel events.
```

How to Compress Audit Files on a Dedicated File System

Audit files can grow large. You can set an upper limit to the size of a file, as shown in Example 4-3. In this procedure, you use compression to reduce the size.

Before You Begin You must become an administrator who is assigned the ZFS File System Management and ZFS Storage Management rights profiles. The latter profile enables you to create storage pools. For more information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2".

1. **Dedicate a ZFS file system for audit files.**

   For the procedure, see "How to Create ZFS File Systems for Audit Files" on page 74.

2. **Compress the ZFS storage pool by using one of the following options.**

   With both options, the audit file system is compressed. After the audit service is refreshed, the compression ratio is displayed.

   In the following examples, the ZFS pool `auditp/auditf` is the dataset.

   - **Use the default compression algorithm.**

```
# zfs set compression=on auditp/auditf
# audit -s
# zfs get compressratio auditp/auditf
NAME           PROPERTY       VALUE  SOURCE
auditp/auditf  compressratio  4.54x  -
```

   - **Use a higher compression algorithm.**

```
# zfs set compression=gzip-9 auditp/auditf
```

```
# zfs get compression auditp/auditf
NAME           PROPERTY     VALUE     SOURCE
auditp/auditf  compression  gzip-9    local
```

The `gzip-9` compression algorithm results in files that occupy one-third less space than the default compression algorithm, `lzjb`. For more information, see Chapter 5, "Managing Oracle Solaris ZFS File Systems," in "Managing ZFS File Systems in Oracle Solaris 11.2 ".

3. **Refresh the audit service.**

```
# audit -s
```

4. **(Optional) Verify the new compression setting.**

For example, if you used the higher compression algorithm, the information would be similar to the following:

```
# zfs get compressration auditp/auditf
NAME           PROPERTY       VALUE  SOURCE
auditp/auditf  compressratio  16.89x  -
```

How to Audit FTP and SFTP File Transfers

The FTP service creates logs of its file transfers. The SFTP service, which runs under the `ssh` protocol, can be audited by preselecting the `ft` audit class. Logins to both services can be audited.

Note - For information about how to log commands and file transfers of the FTP service, see the `proftpd`(8) man page.

For the available logging options, read ProFTPD Logging (http://www.proftpd.org/docs/howto/Logging.html).

- **Perform one of the following depending on whether you want to audit SFTP or FTP.**

  - To log `sftp` access and file transfers, edit the `ft` class.

    The `ft` class includes the following SFTP transactions:

```
% auditrecord -c ft
file transfer: chmod ...
file transfer: chown ...
file transfer: get ...
```

```
file transfer: mkdir ...
file transfer: put ...
file transfer: remove ...
file transfer: rename ...
file transfer: rmdir ...
file transfer: session start ...
file transfer: session end ...
file transfer: symlink ...
file transfer: utimes
```

- To record access to the FTP server, audit the `lo` class.

  As the following sample output indicates, logging in to and out of the `proftpd` daemon generates audit records.

```
% auditrecord -c lo | more
...
FTP server login
program     proftpd             See in.ftpd(1M)
event ID    6165                AUE_ftpd
class       lo                  (0x0000000000001000)
header
subject
[text]                      error message
return

FTP server logout
program     proftpd             See in.ftpd(1M)
event ID    6171                AUE_ftpd_logout
class       lo                  (0x0000000000001000)
header
subject
return
...
```

Configuring the Audit Service in Zones

The audit service audits the entire system, including audit events in zones. A system that has installed non-global zones can audit all zones identically, or can configure auditing per zone. For more information, see “Planning Auditing in Zones” on page 26.

When you audit the non-global zones exactly as the global zone is audited, the non-global zone administrators might not have access to the audit records. Also, the global zone administrator can modify the audit preselection masks of users in non-global zones.

When you audit the non-global zones individually, the audit records are visible to the non-global zone and to the global zone from the non-global zone root.

How to Configure All Zones Identically for Auditing

This procedure enables audits every zone identically. This method requires the least computer overhead and administrative resources.

Before You Begin You must assume the `root` role. For more information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2 ".

1. **Configure the global zone for auditing.**

   Complete the tasks in "Configuring the Audit Service" on page 42, with the following exceptions:

   - Do not enable `perzone` audit policy.
   - Set the `zonename` policy. This policy adds the name of the zone to every audit record.

     ```
     # auditconfig -setpolicy +zonename
     ```

2. **If you modified audit configuration files, copy them from the global zone to every non-global zone.**

   If you modified the `audit_class` or `audit_event` file, copy it in one of two ways:

   - You can loopback mount the files.
   - You can copy the files.

   The non-global zone must be running.

   - **Mount the changed `audit_class` and `audit_event` files as a loopback file system (`lofs`).**

     a. **From the global zone, halt the non-global zone.**

        ```
        # zoneadm -z non-global-zone halt
        ```

     b. **Create a read-only loopback mount for every audit configuration file that you modified in the global zone.**

        ```
        # zonecfg -z non-global-zone
        zone: add fs
        zone/fs: set special=/etc/security/audit-file
        ```

```
zone/fs: set dir=/etc/security/audit-file
zone/fs: set type=lofs
zone/fs: add options [ro,nodevices,nosetuid]
zone/fs: commit
zone/fs: end
zone: exit
#
```

c. To make the changes effective, boot the non-global zone.

```
# zoneadm -z non-global-zone boot
```

Later, if you modify an audit configuration file in the global zone, you reboot each zone to refresh the loopback-mounted files in the non-global zones.

- **Copy the files.**

a. From the global zone, list the `/etc/security` directory in each non-global zone.

```
# ls /zone/zonename/root/etc/security/
```

b. Copy the changed `audit_class` and `audit_event` files to each zone's `/etc/security` directory.

```
# cp /etc/security/audit-file /zone/zonename/root/etc/security/audit-file
```

Later, if you change one of these files in the global zone, you must copy the changed file to the non-global zones.

The non-global zones are audited when the audit service is restarted in the global zone or when the zones are rebooted.

Example 3-17 Mounting Audit Configuration Files as Loopback Mounts in a Zone

In this example, the system administrator has modified the `audit_class`, `audit_event`, and `audit_warn` files.

The `audit_warn` file is read in the global zone only, so does not have to be mounted into the non-global zones.

On this system, `machine1`, the administrator has created two non-global zones, `machine1–webserver` and `machine1–appserver`. The administrator has finished modifying the audit configuration files. If the administrator later modifies the files, the zone must be rebooted to re-read the loopback mounts.

```
# zoneadm -z machine1-webserver halt
# zoneadm -z machine1-appserver halt
# zonecfg -z machine1-webserver
```

```
webserver: add fs
webserver/fs: set special=/etc/security/audit_class
webserver/fs: set dir=/etc/security/audit_class
webserver/fs: set type=lofs
webserver/fs: add options [ro,nodevices,nosetuid]
webserver/fs: commit
webserver/fs: end
webserver: add fs
webserver/fs: set special=/etc/security/audit_event
webserver/fs: set dir=/etc/security/audit_event
webserver/fs: set type=lofs
webserver/fs: add options [ro,nodevices,nosetuid]
webserver/fs: commit
webserver/fs: end
webserver: exit
#

# zonecfg -z machine1-appserver
appserver: add fs
appserver/fs: set special=/etc/security/audit_class
appserver/fs: set dir=/etc/security/audit_class
appserver/fs: set type=lofs
appserver/fs: add options [ro,nodevices,nosetuid]
appserver/fs: commit
appserver/fs: end
appserver: exit
```

When the non-global zones are rebooted, the `audit_class` and `audit_event` files are read-only in the zones.

How to Configure Per-Zone Auditing

This procedure enables separate zone administrators to control the audit service in their zone. For the complete list of policy options, see the `auditconfig`(1M) man page.

Before You Begin To configure auditing, you must become an administrator who is assigned the Audit Configuration rights profile. To enable the audit service, you must become an administrator who is assigned the Audit Control rights profile . For more information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2 ".

1. **In the global zone, configure auditing.**

   a. **Complete the tasks in "Configuring the Audit Service" on page 42.**

   b. **Add the `perzone` audit policy. For the command, see Example 3-12.**

Note - You are not required to enable the audit service in the global zone.

2. **In each non-global zone that you plan to audit, configure the audit files.**

   a. **Complete the tasks in "Configuring the Audit Service" on page 42.**

   b. **Do not configure system-wide audit settings.**

      Specifically, do not add the `perzone` or `ahlt` policy to the non-global zone.

3. **Enable auditing in your zone.**

```
myzone# audit -s
```

Example 3-18 Disabling Auditing in a Non-Global Zone

This example works if the `perzone` audit policy is set. The zone administrator of the `noaudit` zone disables auditing for that zone.

```
noauditzone # auditconfig -getcond
audit condition = auditing
noauditzone # audit -t
noauditzone # auditconfig -getcond
audit condition = noaudit
```

Example: Configuring Oracle Solaris Auditing

This section provides an example of how you configure and implement Oracle Solaris auditing. It begins with the configuration of different attributes of the service according to specific needs and requirements. After configuration is completed, the audit service is started to effect the configuration settings. Each time that you need to revise an existing audit configuration to accommodate new requirements, follow the same sequence of actions in this example:

1. Configure the audit parameters.
2. Refresh the audit service.
3. Verify the new audit configuration.

- First, the administrator adds a temporary policy.

```
# auditconfig -t -setpolicy +zonename
# auditconfig -getpolicy
configured audit policies = ahlt,arge,argv,perzone
```

```
active audit policies = ahlt,arge,argv,perzone,zonename
```

- Then, the administrator specifies queue controls.

```
# auditconfig -setqctrl 200 20 0 0
# auditconfig -getqctrl
configured audit queue hiwater mark (records) = 200
configured audit queue lowater mark (records) = 20
configured audit queue buffer size (bytes) = 8192
configured audit queue delay (ticks) = 20
active audit queue hiwater mark (records) = 200
active audit queue lowater mark (records) = 20
active audit queue buffer size (bytes) = 8192
active audit queue delay (ticks) = 20
```

- Then, the administrator specifies plugin attributes.
  - For the `audit_binfile` plugin, the administrator removes the `qsize` value.

```
# auditconfig -getplugin audit_binfile
Plugin: audit_binfile
Attributes: p_dir=/audit/sys1.1,/var/audit;
p_minfree=2;p_fsize=4G;
Queue size: 200
# auditconfig -setplugin audit_binfile  "" 0
# auditconfig -getplugin audit_binfile
Plugin: audit_binfile
Attributes: p_dir=/audit/sys1.1,/var/audit
p_minfree=2;p_fsize=4G;
```

  - For the `audit_syslog` plugin, the administrator specifies that successful login and logout events and failed executables be sent to `syslog`. The `qsize` for this plugin is set to 150.

```
# auditconfig -setplugin audit_syslog active p_flags=+lo,-ex 150
# auditconfig -getplugin audit_syslog
auditconfig -getplugin audit_syslog
Plugin: audit_syslog
Attributes: p_flags=+lo,-ex;
Queue size: 150
```

  - The administrator does not configure or use the `audit_remote` plugin.
- Then, the administrator refreshes the audit service and verifies the configuration.
  - The temporary `zonename` policy is no longer set.

```
# audit -s
# auditconfig -getpolicy
configured audit policies = ahlt,arge,argv,perzone
active audit policies = ahlt,arge,argv,perzone
```

  - The queue controls remain the same.

```
# auditconfig -getqctrl
configured audit queue hiwater mark (records) = 200
configured audit queue lowater mark (records) = 20
configured audit queue buffer size (bytes) = 8192
configured audit queue delay (ticks) = 20
active audit queue hiwater mark (records) = 200
active audit queue lowater mark (records) = 20
active audit queue buffer size (bytes) = 8192
active audit queue delay (ticks) = 20
```

- The `audit_binfile` plugin does not have a specified queue size. The `audit_syslog` plugin has a specified queue size.

```
# auditconfig -getplugin
Plugin: audit_binfile
Attributes: p_dir=/var/audit;p_fsize=4G;p_minfree=2;

Plugin: audit_syslog
Attributes: p_flags=+lo,-ex;
Queue size: 50
...
```

♦ ♦ ♦ CHAPTER 4

Monitoring System Activities

This chapter provides procedures to help you configure audit logs that enable to you to monitor activities in the system. In addition, the following chapters describe other audit management tasks:

- Chapter 3, “Managing the Audit Service”
- Chapter 5, “Working With Audit Data”
- Chapter 6, “Analyzing and Resolving Audit Service Issues”

For an overview of the audit service, see Chapter 1, “About Auditing in Oracle Solaris”. For planning suggestions, see Chapter 2, “Planning for Auditing”. For reference information, see Chapter 7, “Auditing Reference”.

Configuring Audit Logs

Two audit plugins, `audit_binfile` and `audit_syslog`, can create local audit logs. The following tasks explain how to configure these logs.

Configuring Audit Logs

The following task map points to the procedures for configuring audit logs for the various plugins. Configuring logs for the `audit_binfile` plugin is optional. The logs for other plugins must be configured by an administrator.

TABLE 4-1 Configuring Audit Logs Task Map

Task	Description	For Instructions
Add local storage for the `audit_binfile` plugin	Creates additional disk space for the audit files and protects them with file permissions	“How to Create ZFS File Systems for Audit Files” on page 74
Assign storage for the `audit_binfile` plugin	Identifies directories for binary audit records	“How to Assign Audit Space for the Audit Trail” on page 77

Task	Description	For Instructions
Configure streaming audit records to a remote system	Enables you to send audit records to a remote repository through a protected mechanism	"How to Send Audit Files to a Remote Repository" on page 81
Configure remote storage for audit files	Enables you to receive audit records on a remote system	"How to Configure a Remote Repository for Audit Files" on page 82
Configure storage for the `audit_syslog` plugin.	Enables you to stream audit events in text format to `syslog`.	"How to Configure `syslog` Audit Logs" on page 86

▼ How to Create ZFS File Systems for Audit Files

The following procedure shows how to create a ZFS pool for audit files, as well as the corresponding file systems and mount point. By default, the `/var/audit` file system holds audit files for the `audit_binfile` plugin.

Before You Begin You must become an administrator who is assigned the ZFS File System Management and ZFS Storage Management rights profiles. The latter profile enables you to create storage pools. For more information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2".

1. **Determine the amount of disk space that is required.**

   Assign at least 200 MB of disk space per host. However, how much auditing you require dictates the disk space requirements. Your disk space requirements might be far greater than this figure.

   Note - The default class preselection creates files in `/var/audit` that grow by about 80 bytes for every recorded instance of an event in the `lo` class, such as a login, logout, or role assumption.

   ---

2. **Create a mirrored ZFS storage pool.**

   The `zpool create` command creates a storage pool, that is, a container for the ZFS file systems. For more information, see Chapter 1, "Oracle Solaris ZFS File System (Introduction)," in "Managing ZFS File Systems in Oracle Solaris 11.2 ".

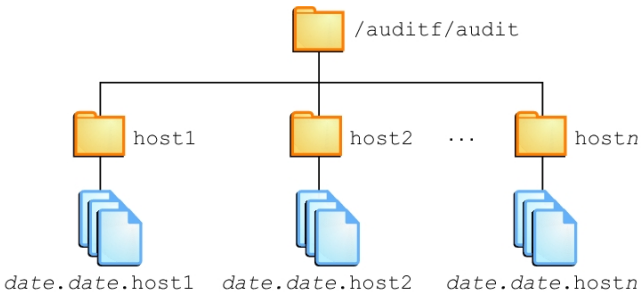
   ```
   # zpool create audit-pool mirror disk1 disk2
   ```

   For example, create the `auditp` pool from two disks, `c3t1d0` and `c3t2d0`, and mirror them.

   ```
   # zpool create auditp mirror c3t1d0 c3t2d0
   ```

3. **Create a ZFS file system and mount point for the audit files.**

You create the file system and mount point with one command. At creation, the file system is mounted. For example, the following illustration shows audit trail storage that is stored by host name.



Note - If you plan to encrypt the file system, you must encrypt the file system at creation. For an example, see Example 4-1.

Encryption requires management. For example, a passphrase is required at mount time. For more information, see "Encrypting ZFS File Systems" in "Managing ZFS File Systems in Oracle Solaris 11.2 ".

```
# zfs create -o mountpoint=/mountpoint audit-pool/mountpoint
```

For example, create the `/audit` mount point for the `auditf` file system.

```
# zfs create -o mountpoint=/audit auditp/auditf
```

4. **Create a ZFS file system for the audit files.**

```
# zfs create -p auditp/auditf/system
```

For example, create an unencrypted ZFS file system for the `sys1` system.

```
# zfs create -p auditp/auditf/sys1
```

5. **(Optional) Create additional file systems for audit files.**

One reason to create additional file systems is to prevent audit overflow. You can set a ZFS quota per file system, as shown in Step 8. The `audit_warn` email alias notifies you when each quota is reached. To free space, you can move the closed audit files to a remote server.

```
# zfs create -p auditp/auditf/sys1.1

# zfs create -p auditp/auditf/sys1.2
```

6. **Protect the parent audit file system.**

   The following ZFS properties are set to `off` for all file systems in the pool:

   ```
   # zfs set devices=off auditp/auditf

   # zfs set exec=off auditp/auditf

   # zfs set setuid=off auditp/auditf
   ```

7. **Compress the audit files in the pool.**

   Typically, compression is set in ZFS at the file system level. However, because all the file systems in this pool contain audit files, compression is set at the top-level dataset for the pool.

   ```
   # zfs set compression=on auditp
   ```

   See also "Interactions Between ZFS Compression, Deduplication, and Encryption Properties" in "Managing ZFS File Systems in Oracle Solaris 11.2 ".

8. **Set quotas.**

   You can set quotas at the parent file system, the descendant file systems, or both. If you set a quota on the parent audit file system, quotas on the descendant file systems impose an additional limit.

   a. **Set a quota on the parent audit file system.**

      In the following example, when both disks in the `auditp` pool reach the quota, the `audit_warn` script notifies the audit administrator.

      ```
      # zfs set quota=510G auditp/auditf
      ```

   b. **Set a quota on the descendant audit file systems.**

      In the following example, when the quota for the `auditp/auditf/`*system* file system is reached, the `audit_warn` script notifies the audit administrator.

      ```
      # zfs set quota=170G auditp/auditf/sys1

      # zfs set quota=170G auditp/auditf/sys1.1

      # zfs set quota=165G auditp/auditf/sys1.2
      ```

9. **For a large pool, limit the size of the audit files.**

   By default, an audit file can grow to the size of the pool. For manageability, limit the size of the audit files. See Example 4-3.

Example 4-1 Creating an Encrypted File System for Audit Files

To comply with site security requirements, the administrator performs the following steps:

1. Creates, if necessary, a new ZFS pool to store the encrypted audit logs.
2. Generates an encryption key.
3. Creates the audit file system with encryption turned on to store the audit logs, as well as sets the mount point.
4. Configures auditing to use the encrypted directory.
5. Refreshes the audit service to apply the new configuration settings.

```
# zpool create auditp mirror disk1 disk2
```

```
# pktool genkey keystore=file outkey=/filename keytype=aes keylen=256

# zfs create -o encryption=aes-256-ccm \
-o keysource=raw,file:///filename \
-o compression=on -o mountpoint=/audit auditp/auditf

# auditconfig -setplugin audit_binfile p_dir=/audit/

# audit -s
```

You must back up and protect the file where the key is stored, such as *filename* in the example.

When the administrator creates additional file systems under the `auditf` file system, these descendant file systems are also encrypted.

Example 4-2 Setting a Quota on the `/var/audit` Directory

In this example, the administrator sets a quota on the default audit file system. When this quota is reached, the `audit_warn` script warns the audit administrator.

```
# zfs set quota=252G rpool/var/audit
```

▼ How to Assign Audit Space for the Audit Trail

In this procedure, you use attributes to the `audit_binfile` plugin to assign additional disk space to the audit trail.

Before You Begin You must become an administrator who is assigned the Audit Configuration rights profile. For more information, see “Using Your Assigned Administrative Rights” in “Securing Users and Processes in Oracle Solaris 11.2 ”.

1. Determine the attributes to the `audit_binfile` plugin.

Read the `OBJECT ATTRIBUTES` section of the `audit_binfile`(5) man page.

```
# man audit_binfile

...
OBJECT ATTRIBUTES
The p_dir attribute specifies where the audit files will be created.
The directories are listed in the order in which they are to be used.

The p_minfree attribute defines the percentage of free space that the
audit system requires before the audit daemon invokes the audit_warn
script.

The p_fsize attribute defines the maximum size that an audit
file can become before it is automatically closed and a new
audit file is opened. ... The format of the p_fsize value can
be specified as an exact value in bytes or in a human-readable
form with a suffix of  B,  K, M, G, T, P, E, Z (for bytes,
kilobytes, megabytes, gigabytes, terabytes,  petabytes, exabytes,
or zettabytes, respectively). Suffixes of KB, MB, GB, TB, PB, EB,
and ZB are also accepted.
```

2. To add directories to the audit trail, specify the `p_dir` attribute.

The default file system is `/var/audit`.

```
# auditconfig -setplugin audit_binfile p_dir=/audit/sys1.1,/var/audit
```

The preceding command sets the `/audit/sys1.1` file system as the primary directory for audit files and the default `/var/audit` file system as the secondary directory. In this scenario, `/var/audit` is the directory of last resort. For this configuration to succeed, the `/audit/sys1.1` file system must exist.

A similar file system is created in "How to Create ZFS File Systems for Audit Files" on page 74.

3. Refresh the audit service.

The `auditconfig -setplugin` command sets the *configured* value. This value is a property of the audit service, so it is restored when the service is refreshed or restarted. The configured value becomes *active* when the audit service is refreshed or restarted. For information about configured and active values, see the `auditconfig`(1M) man page.

```
# audit -s
```

Example 4-3 Limiting File Size for the `audit_binfile` Plugin

In the following example, the size of a binary audit file is set to a specific size. The size is specified in megabytes.

```
# auditconfig -setplugin audit_binfile p_fsize=4M

# auditconfig -getplugin audit_binfile
Plugin: audit_binfile
```

```
Attributes: p_dir=/var/audit;p_fsize=4M;p_minfree=1;
```

By default, an audit file can grow without limit. To create smaller audit files, the administrator specifies a file size limit of 4 MB. The audit service creates a new file when the size limit is reached. The file size limit goes into effect after the administrator refreshes the audit service.

```
# audit -s
```

Example 4-4 Specifying Time for Log Rotation

In the following example, a time limit is set for an audit file. The time limit is specified in terms of hours, days, weeks, months, or years.

```
# auditconfig -setplugin audit_binfile "p_age=1w"

# auditconfig -getplugin audit_binfile
Plugin: audit_binfile
Attributes: p_dir=/var/audit;p_age=1w;
Queue size: 200
```

By default, an audit file has no time limit. The file remains open indefinitely until an external operation causes a file rotation. The administrator sets the file's time limit to one week, beyond which a new audit file is opened. To implement the new time limit, the administrator refreshes the audit service.

```
# audit -s
```

Example 4-5 Specifying Several Changes to an Audit Plugin

In the following example, the administrator on a system with high throughput and a large ZFS pool changes the queue size, the binary file size, and the soft limit warning for the `audit_binfile` plugin. The administrator allows audit files to grow to 4 GB, is warned when 2 percent of the ZFS pool remains, and doubles the allowed queue size. The default queue size is the high water mark for the kernel audit queue, `100`, as in `active audit queue hiwater mark (records) = 100`. The audit file is also set to have a time limit of 2 weeks.

```
# auditconfig -getplugin audit_binfile
Plugin: audit_binfile
Attributes: p_dir=/var/audit;p_fsize=2G;p_minfree=1;

# auditconfig -setplugin audit_binfile \
      "p_minfree=2;p_fsize=4G;p_age=2w" 200

# auditconfig -getplugin audit_binfile
Plugin: audit_binfile
Attributes: p_dir=/var/audit;p_fsize=4G;p_minfree=2;p_age=2w;
Queue size: 200
```

The changed specifications go into effect after the administrator refreshes the audit service.

```
# audit -s
```

Example 4-6 Removing Queue Size for an Audit Plugin

In the following example, the queue size for the `audit_binfile` plugin is removed.

```
# auditconfig -getplugin audit_binfile
Plugin: audit_binfile
Attributes: p_dir=/var/audit;p_fsize=4G;p_minfree=2;
Queue size: 200

# auditconfig -setplugin audit_binfile "" 0

# auditconfig -getplugin audit_binfile
Plugin: audit_binfile
Attributes: p_dir=/var/audit;p_fsize=4G;p_minfree=2;
```

The empty quotation marks (`""`) retain the current attribute values. The final `0` sets the queue size for the plugin to the default.

The change in `qsize` specification for the plugin goes into effect after the administrator refreshes the audit service.

```
# audit -s
```

Example 4-7 Setting a Soft Limit for Warnings

In this example, the minimum free-space level for all audit file systems is set so that a warning is issued when two percent of the file system is still available.

```
# auditconfig -setplugin audit_binfile p_minfree=2
```

The default percentage is one (`1`). For a large ZFS pool, choose a reasonably low percentage. For example, 10 percent of a 16 TB pool is around 16 GB, which would warn the audit administrator when plenty of disk space remains. A value of `2` sends the `audit_warn` message when about two GB of disk space remains.

The `audit_warn` email alias receives the warning. To set up the alias, see “How to Configure the `audit_warn` Email Alias” on page 53.

For a large pool, the administrator also limits the file size to 3 GB.

```
# auditconfig -setplugin audit_binfile p_fsize=3G
```

The `p_minfree` and `p_fsize` specifications for the plugin go into effect after the administrator refreshes the audit service.

```
# audit -s
```

How to Send Audit Files to a Remote Repository

In this procedure, you use attributes of the `audit_remote` plugin to send the audit trail to a remote audit repository. To configure a remote repository on an Oracle Solaris system, see "How to Configure a Remote Repository for Audit Files" on page 82.

Before You Begin You must have a receiver of audit files at your remote repository. You must become an administrator who is assigned the Audit Configuration rights profile. For more information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2 ".

1. **Determine the attributes of the `audit_remote` plugin.**

   Read the `OBJECT ATTRIBUTES` section of the `audit_remote`(5) man page.

   ```
   # man audit_remote

   ...
   OBJECT ATTRIBUTES
   The p_hosts attribute specifies the remote servers.
   You can also specify the port number and the GSS-API
   mechanism.

   The p_retries attribute specifies the number of retries for
   connecting and sending data. The default is 3.

   The p_timeout attribute specifies the number of seconds
   in which a connection times out.
   ```

   The default port is the `solaris_audit` IANA-assigned port, `16162/tcp`. The default mechanism is `kerberos_v5`. The timeout default is `5` seconds. You can also specify a queue size for the plugin.

2. **To specify the remote receiving system, use the `p_hosts` attribute.**

   In this example, the receiving system uses a different port.

   ```
   # auditconfig -setplugin audit_remote \
        p_hosts=ars.example.com:16088:kerberos_v5
   ```

3. **Specify other attributes of the plugin that you want to change.**

   For example, the following command specifies values for all optional attributes:

   ```
   # auditconfig -setplugin audit_remote "p_retries=;p_timeout=3" 300
   ```

4. **Verify the values, then activate the plugin.**

   For example, the following commands specify and verify the values of the plugin:

   ```
   # auditconfig -getplugin audit_remote
   Plugin: audit_remote (inactive)
   ```

```
Attributes: p_hosts=ars.example.com:16088:kerberos_v5;p_retries=5;p_timeout=3;
Queue size: 300

# auditconfig -setplugin audit_remote active
```

5. **Refresh the audit service.**

   The audit service reads the audit plugin change upon refresh.

   ```
   # audit -s
   ```

Example 4-8 Tuning the Audit Queue Buffer Size

In this example, the audit queue is full behind the `audit_remote` plugin. This audited system is configured to audit many classes and is transmitting across a high-traffic, slow network. The administrator enlarges the plugin's buffer size to enable the audit queue to grow and not exceed the buffer's limit before records are removed from the queue.

```
audsys1 # auditconfig -setplugin audit_remote "" 1000

audsys1 # audit -s
```

How to Configure a Remote Repository for Audit Files

In this procedure, you configure a remote system, the Audit Remote Server (ARS), to receive and store audit records from one or more audited systems. Then, you activate the audit daemon on the remote server.

The configuration is twofold. First, you configure the underlying security mechanisms to securely transport the audit data, that is, you configure the KDC. Second, you configure the audit service on both the audited system and the ARS. This procedure illustrates a scenario with one audited client and one ARS, where the ARS and the KDC are on the same server. More complex scenarios can be configured similarly. The first four steps describe the configuration of the KDC, while the final step describes the configuration of the audit service.

Before You Begin Ensure that you have completed the following You must assume the `root` role.

- You have assumed the `root` role.
- You have installed the Kerberos packages, as described in “How to Prepare to Stream Audit Records to Remote Storage” on page 32.
- You are working with an administrator who has configured the audited system, as described in “How to Send Audit Files to a Remote Repository” on page 81.

1. **If your site has not yet configured a KDC, configure one.**

You need a KDC on a system that both the audited system and the ARS can use, a host principal for each system, and an `audit` service principal. The following example illustrates a KDC configuration strategy:

```
arstore # kdcmgr -a audr/admin -r EXAMPLE.COM create master
```

This command uses the administrative principal `audr/admin` to create a master KDC in the `EXAMPLE.COM` realm, enables the master KDC, and starts the Kerberos service.

2. **Verify that the KDC is available.**

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

```
# kdcmgr status

KDC Status Information
------------------------------------------
svc:/network/security/krb5kdc:default (Kerberos key distribution center)
State: online since Wed Feb 29 01:59:27 2012
See: man -M /usr/share/man -s 1M krb5kdc
See: /var/svc/log/network-security-krb5kdc:default.log
Impact: None.

KDC Master Status Information
------------------------------------------
svc:/network/security/kadmin:default (Kerberos administration daemon)
State: online since Wed Feb 29 01:59:28 2012
See: man -M /usr/share/man -s 1M kadmind
See: /var/svc/log/network-security-kadmin:default.log
Impact: None.

Transaction Log Information
------------------------------------------

Kerberos update log (/var/krb5/principal.ulog)
Update log dump :
Log version # : 1
Log state : Stable
Entry block size : 2048
Number of entries : 13
First serial # : 1
Last serial # : 13
First time stamp : Wed Feb 29 01:59:27 2012
Last time stamp : Mon Mar 5 19:29:28 2012


Kerberos Related File Information
------------------------------------------
(Displays any missing files)
```

3. **Add the `audit` service principal to the KDC keytab file.**

You can add the principal by typing the `kadmin.local` command on the KDC system. Or, you can remotely add the principal by using the `kadmin` command and providing a password. In this example, the `arstore` system is running the KDC.

```
# kadmin -p audr/admin

kadmin: addprinc -randkey audit/arstore.example.com@EXAMPLE.COM

kadmin: ktadd audit/arstore.example.com@EXAMPLE.COM
```

4. **On each audited system, add keys.**

   The receiver and the sender must have keys.

```
enigma # kclient

.. Enter the Kerberos realm:
EXAMPLE.COM

.. KDC hostname for the above realm:
arstore.example.com

.. Will this client need service keys ? [y/n]:
y
```

5. **Configure the audit service on the ARS.**

   - **To create a group that accepts audit records from any audited system in the Kerberos realm, name a connection group.**

     ```
     # auditconfig -setremote group create Bank_A
     ```

     `Bank_A` is a connection group. Because the `hosts` attribute is not defined, this group accepts all connections, which means that it is a *wildcard* group. Any audited system in this Kerberos realm whose `audit_remote` plugin is correctly configured can reach this ARS.

   - **To limit connections to this group, specify the audited systems that can use this repository.**

     ```
     # auditconfig -setremote group Bank_A "hosts=enigma.example.com"
     ```

     Connection group `Bank_A` now accepts only connections from the `enigma` system. A connection from any other host is refused.

   - **To prevent an audit file in this group from growing too large, set a maximum size.**

     ```
     # auditconfig -setremote group Bank_A "binfile_fsize=4GB"
     ```

```
# auditconfig -getremote
Audit Remote Server
Attributes: listen_address=;login_grace_time=30;max_startups=10;listen_port=0;
Connection group: Bank_A (inactive)
Attributes: binfile_dir=/var/audit;binfile_fsize=4GB;binfile_minfree=1;
hosts=enigma.example.com;
```

6. Configure the audit service on the audited system.

To specify the ARS, use the `p_hosts` attribute.

```
enigma # auditconfig -setplugin audit_remote \
     active p_hosts=arstore.example.com

enigma # auditconfig -getplugin audit_remote
Plugin: audit_remote
Attributes: p_retries=3;p_timeout=5;p_hosts=arstore.example.com;
```

7. Refresh the audit service.

The audit service reads the audit plugin change upon refresh.

```
# audit -s
```

The KDC now manages the connection between the audited system `enigma` and the ARS.

Example 4-9 Streaming Audit Records to Different File Locations on the Same ARS

This example extends the example in the procedure. The administrator separates audit records by host on the ARS by creating two connection groups.

Audit files from `audsys1` stream to the `Bank_A` connection group on this ARS.

```
arstore # auditconfig -setremote group create Bank_A

arstore # auditconfig -setremote group active Bank_A "hosts=audsys1" \
   "hosts=audsys1;binfile_dir=/var/audit/audsys1;binfile_fsize=4M;"
```

Audit files from `audsys2` stream to the `Bank_B` connection group.

```
arstore # auditconfig -setremote group create Bank_B

arstore # auditconfig -setremote group active Bank_B \
"hosts=audsys2;binfile_dir=/var/audit/audsys2;binfile_fsize=4M;"
```

For easier maintenance, the administrator sets other attribute values identically.

```
arstore # auditconfig -getremote
Audit Remote Server
Attributes: listen_address=;login_grace_time=30;max_startups=10;listen_port=0;

Connection group: Bank_A
Attributes: binfile_dir=/var/audit/audsys1;binfile_fsize=4M;binfile_minfree=1;
hosts=audsys1
```

```
Connection group: Bank_B
Attributes: binfile_dir=/var/audit/audsys2;binfile_fsize=4M;binfile_minfree=1;
hosts=audsys2
```

Example 4-10 Placing the ARS on a Different System From the KDC

In this example, the administrator places the ARS on a different system from the KDC. First, the administrator creates and configures the master KDC.

```
kserv # kdcmgr -a audr/admin -r EXAMPLE.COM create master

kserv # kadmin.local -p audr/admin

kadmin: addprinc -randkey \
audit/arstore.example.com@EXAMPLE.COM

kadmin: ktadd -t /tmp/krb5.keytab.audit \
     audit/arstore.example.com@EXAMPLE.COM
```

After securely transmitting the `/tmp/krb5.keytab.audit` file to the ARS, `arstore`, the administrator moves the file to the correct location.

```
arstore # chown root:root krb5.keytab.audit

arstore # chmod 600 krb5.keytab.audit

arstore # mv krb5.keytab.audit /etc/krb5/krb5.keytab
```

Rather than rewrite the file, the administrator also has the option to use the `ktutil` command on the ARS to merge the KDC `krb5.keytab.audit` file with existing keys in the `arstore` 's `/etc/krb5/krb5.keytab` file.

Finally, the administrator generates keys on the audited system.

```
enigma # kclient

.. Enter the Kerberos realm: EXAMPLE.COM

.. KDC hostname for the above realm: kserv.example.com

.. Will this client need service keys ? [y/n]: y
```

▼ How to Configure `syslog` Audit Logs

You can instruct the audit service to copy some or all of the audit records in the audit queue to the `syslog` utility. If you record both binary audit data and text summaries, the binary data provide a complete audit record, while the summaries filter the data for real-time review.

Before You Begin To configure the audit_syslog plugin, you must become an administrator who is assigned the Audit Configuration rights profile. To configure the syslog utility and create the auditlog file, you must assume the root role.

1. **Select audit classes to be sent to the audit_syslog plugin, and make the plugin active.**

   Note - p_flags audit classes must be preselected as either system defaults or in the audit flags of a a user or a rights profile. Records are not collected for a class that is not preselected.

   ```
   # auditconfig -setplugin audit_syslog \
        active p_flags=lo,+as,-ss
   ```

2. **Configure the syslog utility.**

   a. **Add an audit.notice entry to the syslog.conf file.**

      The entry includes the location of the log file.

      ```
      # cat /etc/syslog.conf

      …
      audit.notice       /var/adm/auditlog
      ```

   b. **Create the log file.**

      ```
      # touch /var/adm/auditlog
      ```

   c. **Set the log file's permissions to 640.**

      ```
      # chmod 640 /var/adm/auditlog
      ```

   d. **Check which system-log service instance is running on the system.**

      ```
      # svcs system-log

      STATE        STIME      FMRI
      online       Nov_27     svc:/system/system-log:default
      disabled     Nov 27     svc:/system/system-log:rsyslog
      ```

   e. **Refresh the configuration information for the active syslog service instance.**

      ```
      # svcadm refresh system/system-log:default
      ```

3. **Refresh the audit service.**

   The audit service reads the changes to the audit plugin upon refresh.

```
# audit -s
```

4. **Regularly archive the `syslog` log files.**

   The audit service can generate extensive output. To manage the logs, see the logadm(1M) man page.

Example 4-11 Specifying Audit Classes for `syslog` Output

In the following example, the `syslog` utility collects a subset of the preselected audit classes. The `pf` class is created in Example 3-15.

```
# auditconfig -setnaflags lo,na

# auditconfig -setflags lo,ss

# usermod -K audit_flags=pf:no jdoe

# auditconfig -setplugin audit_syslog \
    active p_flags=lo,+na,-ss,+pf
```

The arguments to the `auditconfig` command instruct the system to collect all login/logout, non-attributable, and change of system state audit records. The `audit_syslog` plugin entry instructs the `syslog` utility to collect all logins, successful non-attributable events, and failed changes of system state.

For the `jdoe` user, the binary utility collects successful and failed calls to the `pfexec` command. The `syslog` utility collects successful calls to the `pfexec` command.

Example 4-12 Putting `syslog` Audit Records on a Remote System

You can change the `audit.notice` entry in the `syslog.conf` file to point to a remote system. In this example, the name of the local system is `sys1.1`. The remote system is `remote1`.

```
sys1.1 # cat /etc/syslog.conf

…
audit.notice       @remote1
```

The `audit.notice` entry in the `syslog.conf` file on the `remote1` system points to the log file.

```
remote1 # cat /etc/syslog.conf

…
audit.notice       /var/adm/auditlog
```

♦ ♦ ♦ CHAPTER 5

Working With Audit Data

This chapter provides procedures to help you with audit data that are generated from different local systems. This chapter covers the following topics:

- "Displaying Audit Trail Data" on page 89
- "Managing Audit Records on Local Systems" on page 97

In addition, the following chapters describe other audit management tasks:

- Chapter 3, "Managing the Audit Service"
- Chapter 4, "Monitoring System Activities"
- Chapter 6, "Analyzing and Resolving Audit Service Issues"

For an overview of the audit service, see Chapter 1, "About Auditing in Oracle Solaris". For planning suggestions, see Chapter 2, "Planning for Auditing". For reference information, see Chapter 7, "Auditing Reference".

Displaying Audit Trail Data

The default plugin, `audit_binfile`, creates an audit trail. The trail can contain large amounts of data. The following sections describe how to work with this data.

Displaying Audit Record Definitions

To display audit record definitions, use the `auditrecord` command. The definitions provide the audit event number, audit class, selection mask, and record format of an audit event.

```
% auditrecord -options
```

The screen output generated by the command depends on the option that you use, as shown in the following partial list.

- The `-p` option displays the audit record definitions of a program.

- The -c option displays the audit record definitions of an audit class.
- The -a option lists all audit event definitions.

You can also the print displayed output to a file.

For more information, see the auditrecord(1M) man page.

EXAMPLE 5-1 Displaying the Audit Record Definitions of a Program

In this example, the definition of all audit records that are generated by the login program are displayed. Login programs include rlogin, telnet, newgrp, and the Secure Shell feature of Oracle Solaris.

```
% auditrecord -p login
...
login: logout
program     various              See login(1)
event ID    6153                 AUE_logout
class       lo                  (0x0000000000001000)
...
newgrp
program     newgrp               See newgrp login
event ID    6212                 AUE_newgrp_login
class       lo                  (0x0000000000001000)
...
rlogin
program     /usr/sbin/login      See login(1) - rlogin
event ID    6155                 AUE_rlogin
class       lo                   (0x0000000000001000)
...
/usr/lib/ssh/sshd
program     /usr/lib/ssh/sshd    See login - ssh
event ID    6172                 AUE_ssh
class       lo                   (0x0000000000001000)
...
telnet login
program     /usr/sbin/login      See login(1) - telnet
event ID    6154                 AUE_telnet
class       lo                   (0x0000000000001000)
…
```

EXAMPLE 5-2 Displaying the Audit Record Definitions of an Audit Class

In this example, the definitions of all audit records in the pf class that was created in Example 3-15 is displayed.

```
% auditrecord -c pf
pfexec
system call pfexec               See execve(2) with pfexec enabled
event ID    116                  AUE_PFEXEC
```

```
class       pf                  (0x0100000000000000)
header
path                    pathname of the executable
path                    pathname of working directory
[privileges]            privileges if the limit or inheritable set are changed
[privileges]            privileges if the limit or inheritable set are changed
[process]               process if ruid, euid, rgid or egid is changed
exec_arguments
[exec_environment]      output if arge policy is set
subject
[use_of_privilege]
return
```

The `use_of_privilege` token is recorded whenever privilege is used. The `privileges` tokens are recorded if the limit or inheritable set is changed. The `process` token is recorded if an ID is changed. No policy option is required for these tokens to be included in the record.

EXAMPLE 5-3 Printing Audit Record Definitions to a File

In this example, the `-h` option is added to put all the audit record definitions to a file in HTML format. When you display the HTML file in a browser, use the browser's Find tool to find specific audit record definitions.

```
% auditrecord -ah > audit.events.html
```

Selecting Audit Events to Be Displayed

As an administrator who is assigned the Audit Review rights profile, you can filter audit records for examination by using the `auditreduce` command. This command can eliminate the less interesting records as it combines the input files.

auditreduce -*option argument* **[***optional-file***]**

where *argument* is the specific argument that an option requires.

The following is a partial list of *record selection* options and their corresponding arguments:

`-c` Selects an audit class where *argument* is an audit class, such as `ua`.

`-d` Selects all of the events on a particular date. The date format for *argument* is *yyymmdd*. Other date options such as `-b` and `-a` select events before and after a particular date respectively.

`-u` Selects all of the events attributable to a particular user. For this option, you specify a user name. Another user option, `-e` , selects all of the events attributable to an effective user ID.

`-g`	Selects all of the events attributable to a particular group. For this option, specify a group name.
`-c`	Selects all of the events in a preselected audit class. To use this option, specify an audit class name.
`-m`	Selects all of the instances of a particular audit event.
`-o`	Selects by object type. Use this option to select by file, group, file owner, FMRI, PID, and other object types.
optional-file	The name of an audit file.

The command also uses *file selection* options which are all in upper case as shown in the following examples. For the full list of options, see the `auditreduce`(1M) man page.

EXAMPLE 5-4 Combining and Reducing Audit Files

In this example, only the login and logout records in audit files that are over a month old are retained. The example assumes that the current date is Sept 27. If you need to retrieve the complete audit trail, you could recover the trail from backup media. The `-O` option directs the command's output to a file named `lo.summary`.

```
# cd /var/audit/audit_summary
# auditreduce -O lo.summary -b 20100827 -c lo; compress *lo.summary
```

EXAMPLE 5-5 Copying One User's Audit Records to a Summary File

In this example, the records in the audit trail that contain the name of a particular user are merged. The `-e` option finds the effective user. The `-u` option finds the login user. The `-O` option directs the output to the file `tamiko`.

```
# cd /var/audit/audit_summary
# auditreduce -e tamiko -O tamiko
```

You can further narrow the displayed information. In this next example, the following are filtered and printed to a file called `tamikolo`.

- Time of user login and logout, specified by the `-c` option.
- Date of Sept 7, 2013, specified by the `-d` option. The short form of the date is *yyyymmdd*.
- User name of tamiko, specified by the `-u` option.
- Name of machine, specified by the `-M` option.

```
# auditreduce -M tamiko -O tamikolo -d 20130907 -u tamiko -c lo
```

EXAMPLE 5-6 Merging Selected Records to a Single File

In this example, login and logout records for a particular day are selected from the audit trail. The records are merged into a target file. The target file is written in a file system other than the file system that contains the audit root directory.

```
# auditreduce -c lo -d 20130827 -O /var/audit/audit_summary/logins

# ls /var/audit/audit_summary/*logins
/var/audit/audit_summary/20130827183936.20130827232326.logins
```

Viewing the Contents of Binary Audit Files

As an administrator who is assigned the Audit Review rights profile, you can view the contents of binary audit files by using the `praudit` command.

```
# praudit options
```

The following is a partial list of options. You can combine any one of these options with the `-l` option to display each record on one line.

`-s` Displays audit records in a short format, one token per line.

`-r` Displays audit records in their raw format, one token per line.

`-x` Displays audit records in XML format, one token per line. This option is useful for further processing.

You can also use the `auditreduce` and `praudit` commands together by piping the `praudit` output from the `auditreduce` command.

EXAMPLE 5-7 Displaying Audit Records in a Short Format

In this example, log in and log out events that are extracted by the `auditreduce` command are displayed in short format.

```
# auditreduce -c lo | praudit -s

header,69,2,AUE_screenlock,,mach1,2010-10-14 08:02:56.348 -07:00
subject,jdoe,root,staff,jdoe,staff,856,50036632,82 0 mach1
return,success,0
sequence,1298
```

EXAMPLE 5-8 Displaying Audit Records in Raw Format

In this example, log in and log out events that are extracted by the `auditreduce` command are displayed in raw format.

```
# auditreduce -c lo | praudit -r

21,69,2,6222,0x0000,10.132.136.45,1287070091,698391050
36,26700,0,10,26700,10,856,50036632,82 0 10.132.136.45
39,0,0
47,1298
```

EXAMPLE 5-9 Putting Audit Records in XML Format

In this example, the audit records are converted to XML format.

```
# praudit -x 20100827183214.20100827215318.logins > 20100827.logins.xml
```

Similarly, you can display audit records filtered by the `auditreduce` command in XML format.

```
# auditreduce -c lo | praudit -x
<record version="2" event="screenlock - unlock" host="mach1"
iso8601="2010-10-14 08:28:11.698 -07:00">
<subject audit-uid="jdoe" uid="root" gid="staff" ruid="jdoe
rgid="staff" pid="856" sid="50036632" tid="82 0 mach1"/>
<return errval="success" retval="0"/>
<sequence seq-num="1298"/>
</record>
```

The contents of the file can be operated on by a script to extract the relevant information.

EXAMPLE 5-10 Making Audit Records in XML Format Readable in a Browser

You can reformat records in the XML file to become readable in any browser by using the `xsltproc` tool. This tool applies stylesheet definitions to the file contents. To put the reformatted contents in a separate file, you would type the following:

```
# auditreduce -c lo | praudit -x | xsltproc - > logins.html
```

In a browser, the contents of `logins.html` would be displayed in a format similar to the following:

```
                    Audit Trail Data

File: time: 2013-11-04 12:54:28.000 -08:00

Event: login - local
time: 2013-11-04 12:54:28.418 -08:00 vers: 2 mod: host: host
SUBJECT audit-uid: jdoe uid: jdoe gid: staff ruid: jdoe rgid: staff
      pid: 1534 sid: 3583012893 tid: 0 0 host
RETURN errval: success retval: 0

Event: connect to RAD
```

```
time: 2013-11-04 12:54:52.029 -08:00 vers: 2 mod: host: host
SUBJECT audit-uid: jdoe uid: jdoe gid: staff ruid: jdoe rgid: staff
      pid: 1835 sid: 3583012893 tid: 0 0 host
RETURN errval: success retval: 0

Event: role login
time: 2013-11-08 08:42:52.286 -08:00 vers: 2 mod: host: host
SUBJECT audit-uid: jdoe uid: root gid: root ruid: root rgid: root
      pid: 4265 sid: 3583012893 tid: 0 0 host
RETURN errval: success retval: 0

Event: role logout
time: 2013-11-08 08:43:37.125 -08:00 vers: 2 mod: host: host
SUBJECT audit-uid: jdoe uid: root gid: root ruid: root rgid: root
      pid: 4265 sid: 3583012893 tid: 0 0 host
RETURN errval: success retval: 0

Event: login - ssh
time: 2013-12-23 12:24:37.292 -08:00 vers: 2 mod: host: host
SUBJECT audit-uid: jsmith uid: jsmith gid: staff ruid: jsmith rgid: staff
      pid: 2002 sid: 39351741 tid: 14632 202240 host.example.com
RETURN errval: success retval: 0

Event: role login
time: 2013-12-23 12:25:07.345 -08:00 vers: 2 mod: fe host: host
SUBJECT audit-uid: jsmith uid: root gid: root ruid: root rgid: root
      pid: 2023 sid: 39351741 tid: 14632 202240 host.example.com
RETURN errval: failure retval: Permission denied

Event: su
time: 2013-12-23 17:19:24.031 -08:00 vers: 2 mod: na host: host
RETURN errval: success retval: 0

Event: su logout
time: 2013-12-23 17:19:24.362 -08:00 vers: 2 mod: na host: host
RETURN errval: success retval: 0

Event: login - ssh
time: 2013-12-23 17:27:21.306 -08:00 vers: 2 mod: host: host
SUBJECT audit-uid: jsmith uid: jsmith gid: staff ruid: jsmith rgid: staff
      pid: 2583 sid: 3401970889 tid: 13861 5632 host.example.com
RETURN errval: success retval: 0

Event: role login
time: 2013-12-23 17:27:28.361 -08:00 vers: 2 mod: host: host
SUBJECT audit-uid: jsmith uid: root gid: root ruid: root rgid: root
      pid: 2593 sid: 3401970889 tid: 13861 5632 host.example.com
RETURN errval: success retval: 0

Event: role logout
time: 2013-12-23 17:30:39.029 -08:00 vers: 2 mod: host: host
SUBJECT audit-uid: jsmith uid: root gid: root ruid: root rgid: root
      pid: 2593 sid: 3401970889 tid: 13861 5632 host.example.com
RETURN errval: success retval: 0
```

Other events

EXAMPLE 5-11 Displaying `pfedit` Records Only

You can use filters to extract and view only specific records from the audit trail. In this example, records that capture the use of the `pfedit` command are filtered. Suppose that the summary file is `20130827183936.20130827232326.logins`. Use of the `pfedit` command generates the `AUE_admin_edit` event. Therefore, to extract `pfedit` records, run the following command:

```
auditreduce -m AUE_admin_edit 20130827183936.20130827232326.logins | praudit
```

EXAMPLE 5-12 Printing the Entire Audit Trail

With a pipe to the print command, the output for the entire audit trail goes to the printer. For security reasons, the printer has limited access.

```
# auditreduce | praudit | lp -d example.protected.printer
```

EXAMPLE 5-13 Viewing a Specific Audit File

In this example, a summary login file is examined in a terminal window.

```
# cd /var/audit/audit_summary/logins

# praudit 20100827183936.20100827232326.logins | more
```

EXAMPLE 5-14 Processing `praudit` Output With a Script

You might want to process output from the `praudit` command as lines of text. For example, you might want to select records that the `auditreduce` command cannot select. You can use a simple shell script to process the output of the `praudit` command. The following sample script puts one audit record on one line, searches for a user-specified string, then returns the audit file to its original form.

```
#!/bin/sh
#
## This script takes an argument of a user-specified string.
#  The sed command prefixes the header tokens with Control-A
#  The first tr command puts the audit tokens for one record
#  onto one line while preserving the line breaks as Control-A
#
praudit | sed -e '1,2d' -e '$s/^file.*$//' -e 's/^header/^aheader/' \\
| tr '\\012\\001' '\\002\\012' \\
| grep "$1" \\
```

Finds the user-specified string

```
| tr '\\002' '\\012'
```

Restores the original newline breaks

Note that the ^a in the script is Control-A, not the two characters ^ and a. The prefix distinguishes the header token from the string header that might appear as text.

A message similar to the following indicates that you do not have enough privilege to use the praudit command:

```
praudit: Can't assign 20090408164827.20090408171614.sys1.1 to stdin.
```

Run the praudit command in a profile shell. You must become an administrator who is assigned the Audit Review rights profile. For more information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2 ".

Managing Audit Records on Local Systems

The following task map points to procedures for selecting, analyzing, and managing audit records.

TABLE 5-1 Managing Audit Records on Local Systems Task Map

Task	Description	For Instructions
Merge audit records.	Combines audit files from several machines into one audit trail.	"How to Merge Audit Files From the Audit Trail" on page 97
Clean up incorrectly named audit files.	Provides an end time stamp to audit files that were inadvertently left open by the audit service.	"How to Clean Up a not_terminated Audit File" on page 99
Prevent audit trail overflow.	Prevents the audit file systems from becoming full.	"Preventing Audit Trail Overflow" on page 100

▼ How to Merge Audit Files From the Audit Trail

By merging the audit files from all the audit directories, you can analyze the contents of the entire audit trail.

Note - Because the time stamps in the audit trail are in Coordinated Universal Time (UTC), the date and hour must be translated to the current time zone to be meaningful. Be aware of this point whenever you manipulate these files with standard file commands rather than with the auditreduce command.

Before You Begin You must become an administrator who is assigned the Audit Review rights profile. For more information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2".

1. **Create a file system for storing merged audit files.**

   To lessen the chance of reaching the limit of disk space, this file system should be in a *different zpool* from the file systems that you created in "How to Create ZFS File Systems for Audit Files" on page 74 to store the original files.

2. **Merge the audit records in the audit trail.**

   Go to the directory for storing merged audit files. From this directory, merge the audit records into a file with a named suffix. All directories in the audit trail on the local system are merged and placed in this directory.

   ```
   # cd audit-storage-directory
   # auditreduce -Uppercase-option -O suffix
   ```

   The uppercase options of the `auditreduce` command manipulate files in the audit trail. The uppercase options include the following:

`-A`	Selects all of the files in the audit trail.
`-C`	Selects complete files only.
`-M`	Selects files with a particular suffix. The suffix can be a machine name or a suffix that you have specified for a summary file.
`-O`	Creates an audit file with 14-character time stamps for both the start time and the end time, with the suffix *suffix* in the current directory.
`-R` *pathname*	Specifies to read audit files in *pathname*, an alternate audit root directory.
`-S` *server*	Specifies to read audit files from the specified server.

   For the full list of options, see the `auditreduce`(1M) man page.

Example 5-15 Copying Audit Files to a Summary File

In the following example, an administrator who is assigned the System Administrator rights profile copies all files from the audit trail into a merged file on a different file system. The `/var/audit/storage` file system is on a separate disk from the `/var/audit` file system, the audit root file system.

```
$ cd /var/audit/storage
$ auditreduce -A -O All
$ ls /var/audit/storage/*All
```

```
20100827183214.20100827215318.All
```

In the following example, only complete files are copied from the audit trail into a merged file. The complete path is specified as the value of the -0 option. The last component of the path, Complete, is used as the suffix.

```
$ auditreduce -C -O /var/audit/storage/Complete

$ ls /var/audit/storage/*Complete
20100827183214.20100827214217.Complete
```

In the following example, by adding the -D option, the original audit files are deleted.

```
$ auditreduce -C -O daily_sys1.1 -D sys1.1

$ ls *sys1.1
20100827183214.20100827214217.daily_sys1.1
```

How to Clean Up a not_terminated Audit File

When anomalous system interruptions occur, the audit service exits while its audit file is still open. Or, a file system becomes inaccessible and forces the system to switch to a new file system. In such instances, an audit file remains with the string not_terminated as the end time stamp, even though the file is no longer used for audit records. Use the auditreduce -O command to give the file the correct time stamp.

Before You Begin You must become an administrator who is assigned the Audit Review rights profile. For more information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2".

1. **List the files with the not_terminated string on your audit file system in order of creation.**

   ```
   # ls -R1t audit-directory */* | grep not_terminated
   ```

   -R Lists files in subdirectories.

   -t Lists files from most recent to oldest.

   -1 Lists the files in one column.

2. **Clean up the old not_terminated file.**

   Specify the name of the old file to the auditreduce -O command.

   ```
   # auditreduce -O system-name old-not-terminated-file
   ```

3. **Remove the old `not_terminated` file.**

   `# rm` *system-name old-not-terminated-file*

Example 5-16 Cleaning Up Closed `not_terminated` Audit Files

In the following example, `not_terminated` files are found, renamed, then the originals are removed.

```
ls -R1t */* | grep not_terminated
…/egret.1/20100908162220.not_terminated.egret
…/egret.1/20100827215359.not_terminated.egret

# cd */egret.1
# auditreduce -O egret 20100908162220.not_terminated.egret
# ls -1t
20100908162220.not_terminated.egret      Current audit file

20100827230920.20100830000909.egret      Cleaned-up audit file

20100827215359.not_terminated.egret      Input (old) audit file

# rm 20100827215359.not_terminated.egret
# ls -1t
20100908162220.not_terminated.egret      Current audit file

20100827230920.20100830000909.egret      Cleaned-up audit file
```

The start time stamp on the new file reflects the time of the first audit event in the `not_terminated` file. The end time stamp reflects the time of the last audit event in the file.

Preventing Audit Trail Overflow

If your security policy requires that all audit data be saved, prevent audit record loss by observing the following practices.

Note - You must assume the `root` role. For more information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2 ".

- Set a minimum free size on the `audit_binfile` plugin.

  Use the `p_minfree` attribute.

  The `audit_warn` email alias sends a warning when the disk space fills to the minimum free size. See Example 4-7.
- Set up a schedule to regularly archive audit files.

Archive audit files by backing up the files to offline media. You can also move the files to an archive file system.

If you are collecting text audit logs with the `syslog` utility, archive the text logs. For more information, see the `logadm`(1M) man page.

- Set up a schedule to delete the archived audit files from the audit file system.
- Save and store auxiliary information.

  Archive information that is necessary to interpret audit records along with the audit trail. Minimally, you save the `passwd`, `group`, and `hosts` files. You also might archive the `audit_event` and `audit_class` files.
- Keep records of which audit files have been archived.
- Store the archived media appropriately.
- Reduce the amount of file system capacity that is required by enabling ZFS compression.

  On a ZFS file system that is dedicated to audit files, compression shrinks the files considerably. For an example, see "How to Compress Audit Files on a Dedicated File System" on page 63.

  See also "Interactions Between ZFS Compression, Deduplication, and Encryption Properties" in "Managing ZFS File Systems in Oracle Solaris 11.2 ".
- Reduce the volume of audit data that you store by creating summary files.

  You can extract summary files from the audit trail by using options to the `auditreduce` command. The summary files contain only records for specified types of audit events. To extract summary files, see Example 5-4 and Example 5-6.

♦ ♦ ♦ CHAPTER 6

Analyzing and Resolving Audit Service Issues

This chapter provides procedures to help you troubleshoot audit-related issues. In addition, the following chapters describe other audit management tasks:

- Chapter 3, “Managing the Audit Service”
- Chapter 4, “Monitoring System Activities”
- Chapter 5, “Working With Audit Data”

For an overview of the audit service, see Chapter 1, “About Auditing in Oracle Solaris”. For planning suggestions, see Chapter 2, “Planning for Auditing”. For reference information, see Chapter 7, “Auditing Reference”.

Troubleshooting the Audit Service

This section covers various auditing error messages, preferences, and the auditing that is provided by other tools to help you debug audit problems.

Typically, different notices are sent to alert you of errors in the audit service. Review your email and the log files if you think that problems exist with the audit service.

- Read the email sent to the `audit_warn` alias.

  The `audit_warn` script sends alert messages to the `audit_warn` email alias. In the absence of a correctly configured alias, the messages are sent to the `root` account.
- Review the log files for the audit service.

  The output from the `svcs -s auditd` command lists the full path to the audit logs that the audit service produces.
- Review the system log files.

  The `audit_warn` script writes `daemon.alert` messages to the `/var/log/syslog` file.

  The `/var/adm/messages` file might contain information.

After you locate and fix the problems, enable or restart the audit service.

```
# audit -s
```

The following sections describe possible problem cases and the steps to resolve them.

Note - Before you perform any troubleshooting tasks, ensure that you have the proper authorization. For example, to configure auditing, you must become an administrator who is assigned the Audit Configuration rights profile. For more information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2 ".

Audit Records Are Not Being Logged

Auditing is enabled by default. If you believe that auditing has not been disabled, but no audit records are being sent to the active plugin, the causes might be one or a combination of the following factors discussed in this section. Note that to modify a system file, you must be assigned the `solaris.admin.edit/`*path-to-system-file* authorization. By default, the root role has this authorization.

Audit Service Not Running

To check whether auditing is running, use any of the following methods:

- Verify the current audit condition.

  The following output indicates that auditing is not running:

  ```
  # auditconfig -getcond
  audit condition = noaudit
  ```

  The following output indicates that auditing is running:

  ```
  # auditconfig -getcond
  audit condition = auditing
  ```

- Verify that the audit service is running.

  The following output indicates that auditing is not running:

  ```
  # svcs -x auditd

  svc:/system/auditd:default (Solaris audit daemon)
  State: disabled since Sun Oct 10 10:10:10 2010
  Reason: Disabled by an administrator.
  See: http://support.oracle.com/msg/SMF-8000-05
  See: auditd(1M)
  See: audit(1M)
  See: auditconfig(1M)
  See: audit_flags(5)
  ```

```
See: audit_binfile(5)
See: audit_syslog(5)
See: audit_remote(5)
See: /var/svc/log/system-auditd:default.log
Impact: This service is not running.
```

The following output indicates that the audit service is running:

```
# svcs auditd
STATE          STIME    FMRI
online         10:10:10 svc:/system/auditd:default
```

If the audit service is not running, enable it. For the procedure, see "Enabling and Disabling the Audit Service" on page 42.

No Audit Plugin Active

Use the following command to check if any plugins are active. At least one plugin must be active for the audit service to work.

```
# audit -v
audit: no active plugin found
```

If no plugin is active, make one active.

```
# auditconfig -setplugin audit_binfile active
# audit -v
configuration ok
```

Audit Class Undefined

You might be attempting to use an audit class that has not been defined. For a description of creating the `pf` class, see "How to Add an Audit Class" on page 54.

For example, the following list of flags contains the `pf` class, which Oracle Solaris software did not deliver:

```
# auditconfig -getflags
active user default audit flags = pf,lo(0x0100000000000000,0x0100000000001000)
configured user default audit flags = pf,lo(0x0100000000000000,0x0100000000001000)
```

If you do not want to define the class, run the `auditconfig -setflags` command with valid values to reset the current flags. Otherwise, ensure the following when defining a class:

- The audit class is defined in the `audit_class` file.

  ```
  # grep pf /etc/security/audit_class
  ```

  Verify class exists

```
0x0100000000000000:pf:profile
```

- The mask is unique. If it is not unique, replace the mask.

```
# grep 0x0100000000000000 /etc/security/audit_class
```

Ensure mask is unique

```
0x0100000000000000:pf:profile
```

No Assigned Events to Audit Class

The customized class that you are using, although defined, might not have any events assigned to the class.

To verify whether events are assigned to the customized class, use one of the following methods:

```
# auditconfig -lsevent | egrep " pf|,pf|pf,"
AUE_PFEXEC      116 pf execve(2) with pfexec enabled
```

```
# auditrecord -c pf
```

List of audit events assigned to pf class

If events are not assigned to the class, assign the appropriate events to this class.

Volume of Audit Records Is Large

After you have determined which events must be audited at your site, use the following suggestions to create audit files with just the information that you require. Note that to assign flags to users, roles, and rights profiles, you must assume the `root` role.

- Specifically, avoid adding events and audit tokens to the audit trail. The following policies increase the size of the audit trail.

`arge`	Adds environment variables to `execv` audit events. Although auditing `execv` events can be costly, adding variables to the audit record is not.
argv	Adds command parameters to `execv` audit events. Adding command parameters to the audit record is not costly.
group	Adds a group token to audit events that include an optional `newgroups` token.
path	Adds a `path` token to audit events that include an optional `path` token.

public	If file events are being audited, adds an event to the audit trail every time an auditable event happens to a public object. File classes include `fa`, `fc`, `fd`, `fm`, `fr`, `fw`, and `cl`. For the definition of a public file, see "Audit Terminology and Concepts" on page 10.
seq	Adds a sequence token to every audit event.
trail	Adds a trailer token to every audit event.
windata_down	On a system that is configured with Trusted Extensions, adds events when information in a labeled window is downgraded.
windata_up	On a system that is configured with Trusted Extensions, adds events when information in a labeled window is upgraded.
zonename	Adds the zone name to every audit event. If the global zone is the only configured zone, adds the string `zone, global` to every audit event.

The following audit record shows the use of the `ls` command. The `ex` class is being audited and the default policy is in use:

```
header,129,2,AUE_EXECVE,,mach1,2010-10-14 11:39:22.480 -07:00
path,/usr/bin/ls
attribute,100555,root,bin,21,320271,18446744073709551615
subject,jdoe,root,root,root,root,2404,50036632,82 0 mach1
return,success,0
```

The following is the same record when all policies are turned on:

```
header,1578,2,AUE_EXECVE,,mach1,2010-10-14 11:45:46.658 -07:00
path,/usr/bin/ls
attribute,100555,root,bin,21,320271,18446744073709551615
exec_args,2,ls,/etc/security
exec_env,49,MANPATH=/usr/share/man,USER=jdoe,GDM_KEYBOARD_LAYOUT=us,EDITOR=gedit,
LANG=en_US.UTF-8,GDM_LANG=en_US.UTF-8,PS1=#,GDMSESSION=gnome,SESSIONTYPE=1,SHLVL=2,
HOME=/home/jdoe,LOGNAME=jdoe,G_FILENAME_ENCODING=@locale,UTF-8, PRINTER=example-dbl,
...
path,/lib/ld.so.1
attribute,100755,root,bin,21,393073,18446744073709551615
subject,jdoe,root,root,root,root,2424,50036632,82 0 mach1
group,root,other,bin,sys,adm,uucp,mail,tty,lp,nuucp,daemon
return,success,0
zone,global
sequence,197
trailer,1578
```

- Use the `audit_syslog` plugin to send some audit events to `syslog`.

  Do not send those audit events to the `audit_binfile` or `audit_remote` plugin. This strategy works only if you are not required to keep binary records of the audit events that you send to the `syslog` logs.
- Set fewer system-wide audit flags and audit individual users.

  Reduce the amount of auditing for all users by reducing the number of audit classes that are audited system-wide.

  Use the `audit_flags` keyword to the `roleadd`, `rolemod`, `useradd`, and `usermod` commands to audit events for specific users and roles. For examples, see Example 4-11 and the `usermod`(1M) man page.

  Use the `always_audit` and `never_audit` properties of the `profiles` command to audit events for specific rights profiles. For information, see the `profiles`(1) man page.

Note - Like other security attributes, audit flags are affected by search order. For more information, see "Order of Search for Assigned Rights" in "Securing Users and Processes in Oracle Solaris 11.2 ".

- Create your own customized audit class.

  You can create audit classes at your site. Into these classes, put only those audit events that you need to monitor. For the procedure, see "How to Add an Audit Class" on page 54.

Note - For information about the effects of modifying an audit configuration file, see "Audit Configuration Files and Packaging" on page 115.

Binary Audit File Sizes Grow Without Limit

As an administrator who is assigned the Audit Review rights profile, you can limit the size of binary files to facilitate archiving and searching. You can also create smaller binary files from the original file by using one of the options described in this section.

- Use the `p_fsize` attribute to limit the size of individual binary audit files.

  For a description of the `p_fsize` attribute, see the OBJECT ATTRIBUTES section of the `audit_binfile`(5) man page.

  For an example, see Example 4-3.
- Use the `auditreduce` command to select records and write those records to a smaller file for further analysis.

  The `auditreduce` *-lowercase* options find specific records.

The `auditreduce -Uppercase` options write your selections to a file. For more information, see the `auditreduce`(1M) man page. See also "Displaying Audit Trail Data" on page 89.

Logins From Other Operating Systems Not Being Audited

The Oracle Solaris OS can audit all logins independent of source. If logins are not being audited, then the `lo` class for both attributable and non-attributable events is probably not set, This class audits logins, logouts, and screen locks. These classes are audited by default.

Note - To audit `ssh` logins, your system must be running the `ssh` daemon from Oracle Solaris. This daemon is modified for the audit service on an Oracle Solaris system. For more information, see "Secure Shell and the OpenSSH Project" in "Managing Secure Shell Access in Oracle Solaris 11.2 ".

EXAMPLE 6-1 Ensuring That Logins Are Audited

In this example, the output of the first two commands shows that the `lo` class for attributable and non-attributable events is not set. Then, the last two commands set the `lo` class to enable auditing of login events.

```
# auditconfig -getflags
active user default audit flags = as,st(0x20800,0x20800)
configured user default audit flags = as,st(0x20800,0x20800)

# auditconfig -getnaflags
active non-attributable audit flags = na(0x400,0x400)
configured non-attributable audit flags = na(0x400,0x400)

# auditconfig -setflags lo,as,st
user default audit flags = as,lo,st(0x21800,0x21800)

# auditconfig -setnaflags lo,na
non-attributable audit flags = lo,na(0x1400,0x1400)
```

♦ ♦ ♦ CHAPTER 7

Auditing Reference

This chapter describes the important components of auditing, and covers the following topics:

For an overview of auditing, see Chapter 1, "About Auditing in Oracle Solaris". For planning suggestions, see Chapter 2, "Planning for Auditing". For procedures to configure auditing at your site, see the following chapters:

- Chapter 3, "Managing the Audit Service"
- Chapter 4, "Monitoring System Activities"
- Chapter 5, "Working With Audit Data"
- Chapter 6, "Analyzing and Resolving Audit Service Issues"

Audit Service

The audit service, `auditd`, is enabled by default. To find out how to enable, refresh, or disable the service, see "Enabling and Disabling the Audit Service" on page 42.

Without customer configuration, the following defaults are in place:

- All login events are audited.

  Both successful and unsuccessful login attempts are audited.
- All users are audited for login and logout events, including role assumption and screen lock.
- The `audit_binfile` plugin is active. The `/var/audit` directory stores audit records, the size of an audit file is not limited, and the queue size is 100 records.
- The `cnt` policy is set.

  When audit records fill the available disk space, the system tracks the number of dropped audit records. A warning is issued when one percent of available disk space remains.
- The following audit queue controls are set:
  - Maximum number of records in the audit queue before generating the records locks is 100
  - Minimum number of records in the audit queue before blocked auditing processes unblock is 10
  - Buffer size for the audit queue is 8192 bytes
  - Interval between writing audit records to the audit trail is 20 seconds

To display the defaults, see "Displaying Audit Service Defaults" on page 40.

The audit service enables you to set temporary, or active, values. These values can differ from configured, or property, values.

- Temporary values are not restored when you refresh or restart the audit service.

  Audit policy and audit queue controls accept temporary values. Audit flags do not have a temporary value.
- Configured values are stored as property values of the service, so they are restored when you refresh or restart the audit service.

Rights profiles control who can administer the audit service. For more information, see "Rights Profiles for Administering Auditing" on page 114.

By default, all zones are audited identically. See "Auditing and Oracle Solaris Zones" on page 114.

Audit Service Man Pages

The following table summarizes the major administrative man pages for the audit service.

Man Page	Summary
`audit`(1M)	Command that controls the actions of the audit service `audit -n` starts a new audit file for the `audit_binfile` plugin.

Man Page	Summary
	`audit -s` enables and refreshes auditing. `audit -t` disables auditing. `audit -v` verifies that at least one plugin is active.
`audit_binfile`(5)	Default audit plugin, which sends audit records to a binary file. See also "Audit Plugins" on page 116.
`audit_remote`(5)	Audit plugin that sends audit records to a remote receiver.
`audit_syslog`(5)	Audit plugin that sends text summaries of audit records to the `syslog` utility.
`audit_class`(4)	File that contains the definitions of audit classes. The eight high-order bits are available for customers to create new audit classes. For more information about the effect of modifying this file on system upgrade, see "How to Add an Audit Class" on page 54.
`audit_event`(4)	File that contains the definitions of audit events and maps the events to audit classes. The mapping can be modified. For more information about the effect of modifying this file on system upgrade, see "How to Change an Audit Event's Class Membership" on page 55.
`audit_flags`(5)	Describes the syntax of audit class preselection, the prefixes for selecting only failed events or only successful events, and the prefixes that modify an existing preselection.
`audit.log`(4)	Describes the naming of binary audit files, the internal structure of a file, and the structure of every audit token.
`audit_warn`(1M)	Script that notifies an email alias when the audit service encounters an unusual condition while writing audit records. You can customize this script for your site to warn of conditions that might require manual intervention or can specify how to handle those conditions automatically.
`auditconfig`(1M)	Command that retrieves and sets audit configuration parameters. Issue this `auditconfig` with no options to display a list of parameters that can be retrieved and set.
`auditrecord`(1M)	Command that displays the definition of audit events in the `/etc/security/audit_event` file. For sample output, see "Displaying Audit Record Definitions" on page 89.
`auditreduce`(1M)	Command that post-selects and merges audit records that are stored in binary format. The command can merge audit records from one or more input audit files. The records remain in binary format. Uppercase options affect file selection. Lowercase options affect record selection.
`auditstat`(1M)	Command that displays kernel audit statistics. For example, the command can display the number of records in the kernel audit queue, the number of dropped records, and the number of audit records that user processes produced in the kernel as a result of system calls.
`praudit`(1M)	Command that reads audit records in binary format from standard input and displays the records in a presentable format. The input can be piped from the `auditreduce` command or from a single audit file or a list of audit files. Input can also be produced with the `tail -0f` command for a current audit file. For sample output, see "Viewing the Contents of Binary Audit Files" on page 93.

Man Page	Summary
syslog.conf(4)	File that is configured to send text summaries of audit records to the `syslog` utility for the `audit_syslog` plugin.

Rights Profiles for Administering Auditing

Oracle Solaris provides rights profiles for configuring the audit service, for enabling and disabling the service, and for analyzing the audit trail. You must have the privileges of `root` to edit an audit configuration file.

- **Audit Configuration –** Enables an administrator to configure the parameters of the audit service and to run the `auditconfig` command.
- **Audit Control –** Enables an administrator to start, refresh, and disable the audit service and to run the `audit` command to start, refresh, or stop the service.
- **Audit Review –** Enables an administrator to analyze audit records. This rights profile grants authorization to read audit records with the `praudit` and `auditreduce` commands. This administrator can also run the `auditstat` command.
- **System Administrator –** Includes the Audit Review rights profile. An administrator with the System Administrator rights profile can analyze audit records.

To configure roles to handle the audit service, see "Creating a Role" in "Securing Users and Processes in Oracle Solaris 11.2 ".

Auditing and Oracle Solaris Zones

Non-global zones can be audited exactly as the global zone is audited, or non-global zones can set their own flags, storage, and audit policy.

When all zones are being audited identically, the `audit_class` and `audit_event` files in the global zone provide the class-event mappings for auditing in every zone. The `+zonename` policy option is useful for post-selecting records by zone name.

Zones can also be audited individually. When the policy option, `perzone`, is set in the global zone, each non-global zone runs its own audit service, handles its own audit queue, and specifies the content and location of its audit records. A non-global zone can also set most audit policy options. It cannot set policy that affects the entire system, so a non-global zone cannot set the `ahlt` or `perzone` policy. For further discussion, see "Auditing on a System With Oracle Solaris Zones" on page 24 and "Planning Auditing in Zones" on page 26.

To learn about zones, see “Introduction to Oracle Solaris Zones and Oracle Solaris 10 Zones ”.

Audit Configuration Files and Packaging

The audit configuration files in Oracle Solaris are marked in the package with the `preserve=renamenew` package attribute. This attribute preserves any modifications you make to the files across updates. For information about the effects of the `preserve` values, see the `pkg`(5) man page.

These configuration files are also marked with the `overlay=allow` package attribute. This attribute enables you to create your own package that contains these files and replace the Oracle Solaris files with files from your package. When you set the `overlay` attribute to `true` in your package, the `pkg` subcommands, such as `verify`, `fix`, `revert`, and so on, will return results on your packages. For more information, see the `pkg`(1) and `pkg`(5) man pages.

Audit Classes

Oracle Solaris defines audit classes as convenient containers for large numbers of audit events.

You can reconfigure audit classes and make new audit classes. Audit class names can be up to 8 characters in length. The class description is limited to 72 characters. Numeric and non-alphanumeric characters are allowed. For more information, see the `audit_class`(4) man page and “How to Add an Audit Class” on page 54.

Caution - The `all` class can generate large amounts of data and quickly fill disks. Use the `all` class only if you have extraordinary reasons to audit all activities.

Audit Class Syntax

Events in an audit class can be audited for success, for failure, and for both.

- Without a prefix, a class of events is audited for success and for failure.
- With a plus (+) prefix, a class of events is audited for success only.
- With a minus (-) prefix, a class of events is audited for failure only.
- To modify a current preselection, add a caret (^) preceding a prefix or an audit flag. For example:

- If `ot` is preselected for the system, and a user's preselection is `^ot`, that user is not audited for events in the `other` class.
- If `+ot` is preselected for the system, and a user's preselection is `^+ot`, that user is not audited for successful events in the `other` class.
- If `-ot` is preselected for the system, and a user's preselection is `^-ot`, that user is not audited for failed events in the `other` class.

To review the syntax of audit class preselection, see the `audit_flags`(5) man page.

The audit classes and their prefixes can be specified in the following commands:

- As arguments to the `auditconfig` command options `-setflags` and `-setnaflags`.
- As values for the `p_flags` attribute to the `audit_syslog` plugin. You specify the attribute as an option to the `auditconfig -setplugin audit_syslog active` command.
- As values for the `-K audit_flags=`*always-audit-flags*`:`*never-audit-flags* option to the `useradd`, `usermod`, `roleadd`, and `rolemod` commands.
- As values for the `-always_audit` and `-never_audit` properties of the `profiles` command.

Audit Plugins

Audit plugins specify how to handle the audit records in the audit queue. The audit plugins are specified by name: `audit_binfile`, `audit_remote`, and `audit_syslog`, as arguments to the `auditconfig -setplugin` command. The plugins can be further specified by the following attributes:

- `audit_binfile` plugin

  `p_dir` attribute – Where to send binary data

  `p_minfree` attribute – Minimum space remaining on a disk before the administrator is warned.

  `p_fsize` attribute – Maximum size of an audit file.
- `audit_remote` plugin

  `p_hosts` attribute– Remote authenticated audit server to which to send the binary audit data.

  `p_retries` attribute – Number of attempts to make to reach a remote authenticated audit server.

  `p_timeout` attribute – Number of seconds between attempts to reach a remote authenticated audit server.
- `audit_syslog` plugin

  `p_flags` attribute – Selection of text summaries of audit records to be sent to `syslog`

- For all plugins, the maximum number of audit records that are queued for the plugin - `qsize` attribute

Refer to the `audit_binfile`(5), `audit_remote`(5), `audit_syslog`(5), and `auditconfig`(1M) man pages.

Audit Remote Server

The Audit Remote Server (ARS) receives audit records over a secure link from audited systems and stores the records.

The reception relies on the following being configured:

- A Kerberos realm with specific audit principals and a GSS-API mechanism
- The ARS with at least one configured and active *connection group*
- At least one audited system in the connection group and a configured and active `audit_remote` plugin

A connection group is specified in the `group` property of the ARS. For file management, `group` can limit the size of an audit file and specify the minimum free space. The primary reason to specify different connection groups is to specify different storage locations on the ARS, as shown in Example 4-9.

For more information about the ARS, see the `ars`(5) man page. For ARS configuration information, see the `-setremote` options in the `auditconfig`(1M) man page.

To configure the audited systems, see the `audit_remote`(5) man page and the `-setplugin` option in the `auditconfig`(1M) man page.

Audit Policy

Audit policy determines whether additional information is added to the audit trail.

The following policies add tokens to audit records: `arge`, `argv`, `group`, `path`, `seq`, `trail`, `windata_down`, `windata_up`, and `zonename`. The `windata_down` and `windata_up` policies are used by the Trusted Extensions feature of Oracle Solaris. For more information, see Chapter 22, "Trusted Extensions and Auditing," in "Trusted Extensions Configuration and Administration ".

The remaining policies do not add tokens. The `public` policy limits auditing of public files. The `perzone` policy establishes separate audit queues for non-global zones. The `ahlt` and `cnt` policies determine what happens when audit records cannot be delivered. For details, see "Audit Policies for Asynchronous and Synchronous Events" on page 118.

The effects of the different audit policy options are described in "Understanding Audit Policy" on page 33. For a description of audit policy options, see the `-setpolicy` option in the `auditconfig`(1M) man page. For a list of available policy options, run the command `auditconfig -lspolicy`. For the current policy, run the command `auditconfig -getpolicy`.

Audit Policies for Asynchronous and Synchronous Events

Together, the `ahlt` policy and the `cnt` policy govern what happens when the audit queue is full and cannot accept more events.

Note - The `cnt` or `ahlt` policies are not triggered if the queue for at least one plugin can accept audit records.

The `cnt` and `ahlt` policies are independent and related. The combination of the policies has the following effects:

- `-ahlt +cnt` is the default policy that is shipped. This default allows the processing of an audited event even if the event cannot be logged.

  The `-ahlt` policy states that if an audit record of an asynchronous event cannot be placed in the kernel audit queue, the system will count the events and continue processing. In the global zone, the `as_dropped` counter records the count.

  The `+cnt` policy states that if a synchronous event arrives and the event cannot be placed in the kernel audit queue, the system will count the event and continue processing. The zone's `as_dropped` counter records the count.

  The `-ahlt +cnt` configuration is generally used at sites where processing must continue, even if continued processing could result in a loss of audit records. The `auditstat drop` field shows the number of audit records that are dropped in a zone.

- The `+ahlt -cnt` policy states that processing halts when an asynchronous event cannot be added to the kernel audit queue.

  The `+ahlt` policy states that if an audit record of an asynchronous event cannot be placed in the kernel audit queue, all processing is stopped. The system will panic. The asynchronous event will not be in the audit queue and must be recovered from pointers on the call stack.

  The `-cnt` policy states that if a synchronous event cannot be placed in the kernel audit queue, the thread that is attempting to deliver the event will be blocked. The thread is placed in a sleep queue until audit space becomes available. No count is kept. Programs might appear to hang until audit space becomes available.

The `+ahlt -cnt` configuration is generally used at sites where a record of every audit event takes precedence over system availability. The `auditstat wblk` field shows the number of times that threads were blocked.

However, if an asynchronous event occurs, the system will panic, leading to an outage. The kernel queue of audit events can be manually recovered from a saved crash dump. The asynchronous event will not be in the audit queue and must be recovered from pointers on the call stack.

- The `-ahlt -cnt` policy states that if an asynchronous event cannot be placed in the kernel audit queue, the event will be counted and processing will continue. When a synchronous event cannot be placed in the kernel audit queue, the thread that is attempting to deliver the event will be blocked. The thread is placed in a sleep queue until audit space becomes available. No count is kept. Programs might appear to hang until audit space becomes available.

  The `-ahlt -cnt` configuration is generally used at sites where the recording of all synchronous audit events takes precedence over some potential loss of asynchronous audit records. The `auditstat wblk` field shows the number of times that threads were blocked.

- The `+ahlt +cnt` policy states that if an asynchronous event cannot be placed in the kernel audit queue, the system will panic. If a synchronous event cannot be placed in the kernel audit queue, the system will count the event and continue processing.

Process Audit Characteristics

The following audit characteristics are set at initial login:

- **Process preselection mask –** A combination of the system-wide audit mask and the user-specific audit mask, if a user audit mask has been specified. When a user logs in, the login process combines the preselected classes to establish the *process preselection mask* for the user's processes. The process preselection mask specifies the events that generate audit records.

  The following algorithm describes how the system obtains the user's process preselection mask:

  `(system-wide default flags` + *always-audit-classes*`)` - *never-audit-classes*

  Add the system-wide audit classes from the results of the `auditconfig -getflags` command to the classes from the *always-audit-classes* value for the user's `always_audit` keyword. Then, from the total, subtract the classes from the user's *never-audit-classes*. See also the `audit_flags`(5) man page.

- **Audit user ID –** A process acquires an immutable audit user ID when the user logs in. This ID is inherited by all child processes that were started by the user's initial process. The audit user ID helps enforce accountability. Even after a user assumes a role, the audit

user ID remains the same. The audit user ID that is saved in each audit record enables you to always trace actions back to the login user.

- **Audit session ID –** The audit session ID is assigned at login. This ID is inherited by all child processes.
- **Terminal ID –** For a local login, the terminal ID consists of the local system's IP address, followed by a unique number that identifies the physical device on which the user logged in. Most often, the login is through the console. The number that corresponds to the console device is `0,0`. For a remote login, the terminal ID consists of a the remote host's IP address followed by the remote port number and the local port number.

Audit Trail

The *audit trail* contains binary audit files. The trail is created by the `audit_binfile` plugin. The audit service collects the records in the audit queue and sends them to the plugin, which writes them to disk.

Conventions for Binary Audit File Names

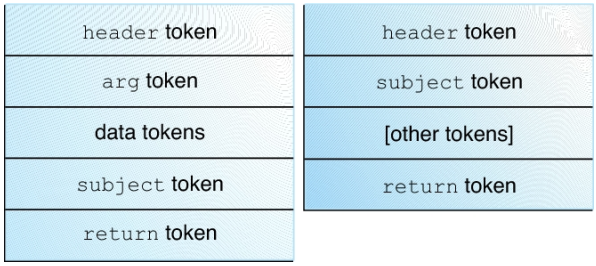
The `audit_binfile` plugin creates binary audit files. Each binary audit file is a self-contained collection of records. The file's name identifies the time span during which the records were generated and the system that generated them. The time stamps that indicate the time span are specified in Coordinated Universal Time (UTC) to ensure that they sort in proper order, even across time zones.

For more information, see the `audit.log`(4) man page. For examples of open and closed audit file names, see "How to Clean Up a `not_terminated` Audit File" on page 99.

Audit Record Structure

An audit record is a sequence of audit tokens. Each audit token contains event information such as user ID, time, and date. A `header` token begins an audit record, and an optional `trailer` token concludes the record. Other audit tokens contain information relevant to the audit event. The following figure shows a typical kernel audit record and a typical user-level audit record.

FIGURE 7-1 Typical Audit Record Structures



Audit Record Analysis

Audit record analysis involves post-selecting records from the audit trail. You can use one of two approaches to parsing the binary data that was collected.

- You can use the `praudit` command. Options to the command provide different text output. For example, the `praudit -x` command provides XML for input into scripts and browsers. `praudit` output does not include fields whose sole purpose is to help to parse the binary data. Note that the order and format of `praudit` output is not guaranteed between Oracle Solaris releases.

  For examples of `praudit` output, see "Viewing the Contents of Binary Audit Files" on page 93.

  For examples of `praudit` output for each audit token, see the individual tokens in "Audit Token Formats" on page 122.
- You can write a program to parse the binary data stream. The program must take into account the variants of an audit record. For example, the `ioctl` system call creates an audit record for "Bad file name." This record contains different tokens from the `ioctl` audit record for "Invalid file descriptor."
  - For a description of the order of binary data in each audit token, see the `audit.log`(4) man page.
  - For manifest values, see the `/usr/include/bsm/audit.h` file.
  - To view the order of tokens in an audit record, use the `auditrecord` command. Output from the `auditrecord` command includes the different tokens for different manifest values. Square brackets (`[]`) indicate that an audit token is optional. For more information, see the `auditrecord`(1M) man page.

Audit Token Formats

Each audit token has a token type identifier, which is followed by data that is specific to the token. The following table shows the token names with a brief description of each token. Obsolete tokens are maintained for compatibility with previous Solaris releases.

TABLE 7-1 Audit Tokens for Auditing

Token Name	Description	For More Information
`acl`	Access Control Entry (ACE) and Access Control List (ACL) information	“`acl` Token” on page 123
`arbitrary`	Data with format and type information	`audit.log`(4) man page
`argument`	System call argument value	“`argument` Token” on page 124
`attribute`	File `vnode` information	“`attribute` Token” on page 124
`cmd`	Command arguments and environment variables	“`cmd` Token” on page 124
`exec_args`	Exec system call arguments	“`exec_args` Token” on page 125
`exec_env`	Exec system call environment variables	“`exec_env` Token” on page 125
`exit`	Program exit information	`audit.log`(4) man page
`file`	Audit file information	“`file` Token” on page 125
`fmri`	Framework Management Resource Indicator	“`fmri` Token” on page 126
`group`	Process groups information	“`group` Token” on page 126
`header`	Indicates start of audit record	“`header` Token” on page 126
`ip`	IP header information	`audit.log`(4) man page
`ip address`	Internet address	“`ip address` Token” on page 127
`ip port`	Internet port address	“`ip port` Token” on page 127
`ipc`	System V IPC information	“`ipc` Token” on page 127
`IPC_perm`	System V IPC object access information	“`IPC_perm` Token” on page 128
`opaque`	Unstructured data (unspecified format)	`audit.log`(4) man page
`path`	Path information	“`path` Token” on page 128
`path_attr`	Access path information	“`path_attr` Token” on page 128
`privilege`	Privilege set information	“`privilege` Token” on page 129
`process`	Process information	“`process` Token” on page 129

Token Name	Description	For More Information
`return`	Status of system call	“`return` Token” on page 129
`sequence`	Sequence number	“`sequence` Token” on page 130
`socket`	Socket type and addresses	“`socket` Token” on page 130
`subject`	Subject information (same format as `process`)	“`subject` Token” on page 130
`text`	ASCII string	“`text` Token” on page 131
`trailer`	Indicates end of audit record	“`trailer` Token” on page 131
`use of authorization`	Use of authorization	“`use of authorization` Token” on page 131
`use of privilege`	Use of privilege	“`use of privilege` Token” on page 131
`user`	User ID and user name	“`user` Token” on page 132
`xclient`	X client identification	“`xclient` Token” on page 132
`zonename`	Name of zone	“`zonename` Token” on page 132
Trusted Extensions tokens	`label` and X Window System information	“Trusted Extensions Audit Reference” in “Trusted Extensions Configuration and Administration ”

The following tokens are obsolete:

- `liaison`
- `host`
- `tid`

For information about obsolete tokens, see the reference material for the release that included the token.

An audit record always begins with a `header` token, which indicates where the audit record begins in the audit trail. In the case of attributable events, the `subject` and the `process` tokens refer to the values of the process that caused the event. In the case of non-attributable events, the `process` token refers to the system.

`acl` Token

The `acl` token uses different formats to record information about Access Control Entries (ACEs) for a ZFS file system and about Access Control Lists (ACLs) for a legacy UFS file system.

When the `acl` token is recorded for a UFS file system, the `praudit -x` command shows the fields as follows:

```
<acl type="1" value="root" mode="6"/>
```

When the `acl` token is recorded for a ZFS dataset, the `praudit -x` command shows the fields as follows:

```
<acl who="root" access_mask="default" flags="-i,-R" type="2"/>
```

`argument` Token

The `argument` token contains information about the arguments to a system call: the argument number of the system call, the argument value, and an optional description. This token allows a 32-bit integer system-call argument in an audit record.

The `praudit -x` command shows the fields of the `argument` token as follows:

```
<argument arg-num="2" value="0x5401" desc="cmd"/>
```

`attribute` Token

The `attribute` token contains information from the file `vnode`.

The `attribute` token usually accompanies a `path` token. The `attribute` token is produced during path searches. If a path-search error occurs, `vnode` is not available to obtain the necessary file information. Therefore, the `attribute` token is not included as part of the audit record. The `praudit -x` command shows the fields of the `attribute` token as follows:

```
<attribute mode="20620" uid="root" gid="tty" fsid="0" nodeid="9267" device="108233"/>
```

`cmd` Token

The `cmd` token records the list of arguments and the list of environment variables that are associated with a command.

The `praudit -x` command shows the fields of the `cmd` token. The following example is a truncated `cmd` token. The line is wrapped for display purposes.

```
<cmd><arge>WINDOWID=6823679</arge>
<arge>COLORTERM=gnome-terminal</arge>
<arge>...LANG=C</arge>...<arge>HOST=machine1</arge>
```

```
<arge>LPDEST=printer1</arge>...</cmd>
```

`exec_args` Token

The `exec_args` token records the arguments to an `exec` system call.

The `praudit -x` command shows the fields of the `exec_args` token as follows:

```
<exec_args><arg>/usr/bin/sh</arg><arg>/usr/bin/hostname</arg></exec_args>
```

Note - The `exec_args` token is output only when the `argv` audit policy option is active.

`exec_env` Token

The `exec_env` token records the current environment variables to an `exec` system call.

The `praudit -x` command shows the fields of the `exec_env` token. The line in the following example is wrapped for display purposes.

```
<exec_env><env>_=/usr/bin/hostname</env>
<env>LANG=C</env><env>PATH=/usr/bin</env>
<env>LOGNAME=jdoe</env><env>USER=jdoe</env>
<env>DISPLAY=:0</env><env>SHELL=/bin/csh</env>
<env>HOME=/home/jdoe</env><env>PWD=/home/jdoe</env><env>TZ=US/Pacific</env>
</exec_env>
```

Note - The `exec_env` token is output only when the `arge` audit policy option is active.

`file` Token

The `file` token is a special token that marks the beginning of a new audit file and the end of an old audit file as the old file is deactivated. The initial `file` token identifies the previous file in the audit trail. The final `file` token identifies the next file in the audit trail. These tokens link successive audit files into one audit trail.

The `praudit -x` command shows the fields of the `file` token. The line in the following example is wrapped for display purposes.

```
<file iso8601="2009-04-08 14:18:26.200 -07:00">
/var/audit/machine1/files/20090408211826.not_terminated.machine1</file>
```

`fmri` Token

The `fmri` token records the use of a fault management resource indicator (FMRI). For more information, see the `smf`(5) man page.

The `praudit -x` command shows the content of the `fmri` token as follows:

```
<fmri service_instance="svc:/system/cryptosvc"</fmri>
```

`group` Token

The `group` token records the group entries from the process's credential. The `group` token is output only when the `group` audit policy option is active.

The `praudit -x` command shows the fields of the `group` token as follows:

```
<group><gid>staff</gid><gid>other</gid></group>
```

`header` Token

The `header` token is special in that it marks the beginning of an audit record. The `header` token combines with the `trailer` token to bracket all the other tokens in the record.

Infrequently, a `header` token can include one or more event modifiers:

- `fe` indicates a failed audit event
- `fp` indicates the failed use of privilege
- `na` indicates a non-attributable event

  ```
  header,52,2,system booted,na,mach1,2011-10-10 10:10:20.564 -07:00
  ```

- `rd` indicates that data is read from the object
- `sp` indicates the successful use of privilege

  ```
  header,120,2,exit(2),sp,mach1,2011-10-10 10:10:10.853 -07:00
  ```

- `wr` indicates that data is written to the object

The `praudit` command displays the `header` token as follows:

```
header,756,2,execve(2),,machine1,2010-10-10 12:11:10.209 -07:00
```

The `praudit -x` command displays the fields of the `header` token at the beginning of the audit record. The line in the following example is wrapped for display purposes.

```
<record version="2" event="execve(2)" host="machine1"
iso8601="2010-10-10 12:11:10.209 -07:00">
```

`ip address` Token

The `ip address` token contains an Internet Protocol address (IP address). The IP address can be displayed in IPv4 or IPv6 format. The IPv4 address uses 4 bytes. The IPv6 address uses 1 byte to describe the address type, and 16 bytes to describe the address.

The `praudit -x` command shows the content of the `ip address` token as follows:

```
<ip_address>machine1</ip_address>
```

`ip port` Token

The `ip port` token contains the TCP or UDP port address.

The `praudit` command displays the `ip port` token as follows:

```
ip port,0xf6d6
```

`ipc` Token

The `ipc` token contains the System V IPC message handle, semaphore handle, or shared-memory handle that is used by the caller to identify a particular IPC object.

The IPC object identifiers violate the context-free nature of the audit tokens. No global “name” uniquely identifies IPC objects. Instead, IPC objects are identified by their handles. The handles are valid only during the time that the IPC objects are active. However, the identification of IPC objects should not be a problem. The System V IPC mechanisms are seldom used, and the mechanisms all share the same audit class.

The following table shows the possible values for the IPC object type field. The values are defined in the `/usr/include/bsm/audit.h` file.

TABLE 7-2 Values for the IPC Object Type Field

Name	Value	Description
AU_IPC_MSG	1	IPC message object
AU_IPC_SEM	2	IPC semaphore object
AU_IPC_SHM	3	IPC shared-memory object

The praudit -x command shows the fields of the ipc token as follows:

```
<IPC ipc-type="shm" ipc-id="15"/>
```

IPC_perm Token

The IPC_perm token contains a copy of the System V IPC access permissions. This token is added to audit records that are generated by IPC shared-memory events, IPC semaphore events, and IPC message events.

The praudit -x command shows the fields of the IPC_perm token. The line in the following example is wrapped for display purposes.

```
<IPC_perm uid="jdoe" gid="staff" creator-uid="jdoe"
creator-gid="staff" mode="100600" seq="0" key="0x0"/>
```

The values are taken from the IPC_perm structure that is associated with the IPC object.

path Token

The path token contains access path information for an object.

The praudit -x command shows the content of the path token as follows:

```
<path>/export/home/srv/.xsession-errors</path>
```

path_attr Token

The path_attr token contains access path information for an object. The access path specifies the sequence of attribute file objects below the path token object. Systems calls such as openat access attribute files. For more information about attribute file objects, see the fsattr(5) man page.

The praudit command displays the path_attr token as follows:

```
path_attr,1,attr_file_name
```

privilege Token

The privilege token records the use of privileges on a process. The privilege token is not recorded for privileges in the basic set. If a privilege has been removed from the basic set by administrative action, then the use of that privilege is recorded. For more information about privileges, see “Process Rights Management” in “Securing Users and Processes in Oracle Solaris 11.2 ”.

The praudit -x command shows the fields of the privilege token.

```
<privilege set-type="Inheritable">ALL</privilege>
```

process Token

The process token contains information about a user who is associated with a process, such as the recipient of a signal.

The praudit -x command shows the fields of the process token. The line in the following example is wrapped for display purposes.

```
<process audit-uid="-2" uid="root" gid="root" ruid="root"
rgid="root" pid="567" sid="0" tid="0 0 0.0.0.0"/>
```

return Token

The return token contains the return status of the system call (u_error) and the process return value (u_rval1).

The return token is always returned as part of kernel-generated audit records for system calls. In application auditing, this token indicates exit status and other return values.

The praudit command displays the return token for a system call as follows:

```
return,failure: Operation now in progress,-1
```

The praudit -x command shows the fields of the return token as follows:

```
<return errval="failure: Operation now in progress" retval="-1/">
```

`sequence` Token

The `sequence` token contains a sequence number. The sequence number is incremented every time an audit record is added to the audit trail. The `sequence` token is output only when the `seq` audit policy option is active. This token is useful for debugging.

The `praudit -x` command shows the content of the `sequence` token:

```
<sequence seq-num="1292"/>
```

`socket` Token

The `socket` token contains information that describes an Internet socket. In some instances, the token includes only the remote port and remote IP address.

The `praudit` command displays this instance of the `socket` token as follows:

```
socket,0x0002,0x83b1,localhost
```

The expanded token adds information, including socket type and local port information.

The `praudit -x` command displays this instance of the `socket` token as follows. The line in the following example is wrapped for display purposes.

```
<socket sock_domain="0x0002" sock_type="0x0002" lport="0x83cf"
laddr="example1" fport="0x2383" faddr="server1.Subdomain.Domain.COM"/>
```

`subject` Token

The `subject` token describes a user who performs or attempts to perform an operation. The format is the same as the `process` token.

The `subject` token is always returned as part of kernel-generated audit records for system calls. The `praudit` command displays the `subject` token as follows:

```
subject,jdoe,root,root,root,root,1631,1421584480,8243 65558 machine1
```

The `praudit -x` command shows the fields of the `subject` token. The line in the following example is wrapped for display purposes.

```
<subject audit-uid="jdoe" uid="root" gid="root" ruid="root"
rgid="root" pid="1631" sid="1421584480" tid="8243 65558 machine1"/>
```

`text` Token

The `text` token contains a text string.

The `praudit -x` command shows the content of the `text` token as follows:

```
<text>booting kernel</text>
```

`trailer` Token

The two tokens, `header` and `trailer`, are special in that they distinguish the beginning and end points of an audit record and bracket all the other tokens. A `header` token begins an audit record. A `trailer` token ends an audit record. The `trailer` token is an optional token, and is added as the last token of each record only when the `trail` audit policy option has been set.

When an audit record is generated with trailers turned on, the `auditreduce` command can verify that the `trailer` token correctly points back to the record header. The `trailer` token supports backward seeks of the audit trail.

The `praudit` command displays the `trailer` token as follows:

```
trailer,136
```

`use of authorization` Token

The `use of authorization` token records the use of authorizations.

The `praudit` command displays the `use of authorization` token as follows:

```
use of authorization,solaris.role.delegate
```

`use of privilege` Token

The `use of privilege` token records the use of privileges.

The `praudit -x` command shows the fields of the `use of privilege` token as follows:

```
<use_of_privilege result="successful use of priv">proc_setid</use_of_privilege>
```

`user` Token

The `user` token records the user name and user ID. This token is present if the user name is different from the caller.

The `praudit -x` command shows the fields of the `user` token as follows:

```
<user uid="123456" username="tester1"/>
```

`xclient` Token

The `xclient` token contains the number of the client connection to the X server.

The `praudit -x` command shows the content of the `xlient` token as follows:

```
<X_client>15</X_client>
```

`zonename` Token

The `zonename` token records the zone in which the audit event occurred. The string “global” indicates audit events that occur in the global zone.

The `praudit -x` command shows the content of the `zonename` token as follows:

```
<zone name="graphzone"/>
```

Security Glossary

Access Control List (ACL) An access control list (ACL) provides finer-grained file security than traditional UNIX file protection provides. For example, an ACL enables you to allow group read access to a file, while allowing only one member of that group to write to the file.

admin principal A user principal with a name of the form *username*`/admin` (as in `jdoe/admin`). An admin principal can have more privileges (for example, to change policies) than a regular user principal. See also principal name, user principal.

AES Advanced Encryption Standard. A symmetric 128-bit block data encryption technique. The U.S. government adopted the Rijndael variant of the algorithm as its encryption standard in October 2000. AES replaces user principal encryption as the government standard.

algorithm A cryptographic algorithm. This is an established, recursive computational procedure that encrypts or hashes input.

application server See network application server.

asynchronous audit event Asynchronous events are the minority of system events. These events are not associated with any process, so no process is available to be blocked and later woken up. Initial system boot and PROM enter and exit events are examples of asynchronous events.

audit files Binary audit logs. Audit files are stored separately in an audit file system.

audit policy The global and per-user settings that determine which audit events are recorded. The global settings that apply to the audit service typically affect which pieces of optional information are included in the audit trail. Two settings, `cnt` and `ahlt`, affect the operation of the system when the audit queue fills. For example, audit policy might require that a sequence number be part of every audit record.

audit trail The collection of all audit files from all hosts.

authenticated rights profile A rights profile that requires the assigned user or role to type a password before executing an operation from the profile. This behavior is similar to `sudo` behavior. The length of time that the password is valid is configurable.

authentication The process of verifying the claimed identity of a principal.

authenticator Authenticators are passed by clients when requesting tickets (from a KDC) and services (from a server). They contain information that is generated by using a session key known only by the client and server, that can be verified as of recent origin, thus indicating that the transaction is secure. When used with a ticket, an authenticator can be used to authenticate a user principal. An authenticator includes the principal name of the user, the IP address of the user's host, and a time stamp. Unlike a ticket, an authenticator can be used only once, usually when access to a service is requested. An authenticator is encrypted by using the session key for that client and that server.

authorization 1. In Kerberos, the process of determining if a principal can use a service, which objects the principal is allowed to access, and the type of access that is allowed for each object.

2. In user rights management, a right that can be assigned to a role or user (or embedded in a rights profile) for performing a class of operations that are otherwise prohibited by security policy. Authorizations are enforced at the user application level, not in the kernel.

basic set The set of privileges that are assigned to a user's process at login. On an unmodified system, each user's initial inheritable set equals the basic set at login.

Blowfish A symmetric block cipher algorithm that takes a variable-length key from 32 bits to 448 bits. Its author, Bruce Schneier, claims that Blowfish is optimized for applications where the key does not change often.

client Narrowly, a process that makes use of a network service on behalf of a user; for example, an application that uses `rlogin`. In some cases, a server can itself be a client of some other server or service.

More broadly, a host that a) receives a Kerberos credential, and b) makes use of a service that is provided by a server.

Informally, a principal that makes use of a service.

client principal (RPCSEC_GSS API) A client (a user or an application) that uses RPCSEC_GSS-secured network services. Client principal names are stored in the form of `rpc_gss_principal_t` structures.

clock skew The maximum amount of time that the internal system clocks on all hosts that are participating in the Kerberos authentication system can differ. If the clock skew is exceeded between any of the participating hosts, requests are rejected. Clock skew can be specified in the `krb5.conf` file.

confidentiality See privacy.

consumer In the Cryptographic Framework feature of Oracle Solaris, a consumer is a user of the cryptographic services that come from providers. Consumers can be applications, end users, or kernel operations. Kerberos, IKE, and IPsec are examples of consumers. For examples of providers, see provider.

credential An information package that includes a ticket and a matching session key. Used to authenticate the identity of a principal. See also ticket, session key.

credential cache A storage space (usually a file) that contains credentials that are received from the KDC.

cryptographic algorithm See algorithm.

DES Data Encryption Standard. A symmetric-key encryption method developed in 1975 and standardized by ANSI in 1981 as ANSI X.3.92. DES uses a 56-bit key.

device allocation Device protection at the user level. Device allocation enforces the exclusive use of a device by one user at a time. Device data is purged before device reuse. Authorizations can be used to limit who is permitted to allocate a device.

device policy Device protection at the kernel level. Device policy is implemented as two sets of privileges on a device. One set of privileges controls read access to the device. The second set of privileges controls write access to the device. See also policy.

Diffie-Hellman protocol Also known as public key cryptography. An asymmetric cryptographic key agreement protocol that was developed by Diffie and Hellman in 1976. The protocol enables two users to exchange a secret key over an insecure medium without any prior secrets. Diffie-Hellman is used by Kerberos.

digest See message digest.

DSA Digital Signature Algorithm. A public key algorithm with a variable key size from 512 to 4096 bits. The U.S. Government standard, DSS, goes up to 1024 bits. DSA relies on SHA1 for input.

ECDSA Elliptic Curve Digital Signature Algorithm. A public key algorithm that is based on elliptic curve mathematics. An ECDSA key size is significantly smaller than the size of a DSA public key needed to generate a signature of the same length.

effective set The set of privileges that are currently in effect on a process.

flavor Historically, *security flavor* and *authentication flavor* had the same meaning, as a flavor that indicated a type of authentication (AUTH_UNIX, AUTH_DES, AUTH_KERB). RPCSEC_GSS is also a security flavor, even though it provides integrity and privacy services in addition to authentication.

forwardable ticket A ticket that a client can use to request a ticket on a remote host without requiring the client to go through the full authentication process on that host. For example, if the user `david` obtains a forwardable ticket while on user `jennifer`'s machine, `david` can log in to his own machine without being required to get a new ticket (and thus authenticate himself again). See also proxiable ticket.

FQDN Fully qualified domain name. For example, `central.example.com` (as opposed to simply `denver`).

GSS-API The Generic Security Service Application Programming Interface. A network layer that provides support for various modular security services, including the Kerberos service.

GSS-API provides for security authentication, integrity, and privacy services. See also authentication, integrity, privacy.

hardening The modification of the default configuration of the operating system to remove security vulnerabilities that are inherent in the host.

hardware provider In the Cryptographic Framework feature of Oracle Solaris, a device driver and its hardware accelerator. Hardware providers offload expensive cryptographic operations from the computer system, thus freeing CPU resources for other uses. See also provider.

host A system that is accessible over a network.

host principal A particular instance of a service principal in which the principal (signified by the primary name `host`) is set up to provide a range of network services, such as `ftp`, `rcp`, or `rlogin`. An example of a host principal is `host/central.example.com@EXAMPLE.COM`. See also server principal.

inheritable set The set of privileges that a process can inherit across a call to `exec`.

initial ticket A ticket that is issued directly (that is, not based on an existing ticket-granting ticket). Some services, such as applications that change passwords, might require tickets to be marked `initial` so as to assure themselves that the client can demonstrate a knowledge of its secret key. This assurance is important because an initial ticket indicates that the client has recently authenticated itself (instead of relying on a ticket-granting ticket, which might existed for a long time).

instance The second part of a principal name, an instance qualifies the principal's primary. In the case of a service principal, the instance is required. The instance is the host's fully qualified domain name, as in `host/central.example.com`. For user principals, an instance is optional. Note, however, that `jdoe` and `jdoe/admin` are unique principals. See also primary, principal name, service principal, user principal.

integrity A security service that, in addition to user authentication, provides for the validity of transmitted data through cryptographic checksumming. See also authentication, privacy.

invalid ticket A postdated ticket that has not yet become usable. An invalid ticket is rejected by an application server until it becomes validated. To be validated, an invalid ticket must be presented to the KDC by the client in a TGS request, with the `VALIDATE` flag set, after its start time has passed. See also postdated ticket.

KDC Key Distribution Center. A machine that has three Kerberos V5 components:

- Principal and key database
- Authentication service
- Ticket-granting service

Each realm has a master KDC and should have one or more slave KDCs.

Kerberos An authentication service, the protocol that is used by that service, or the code that is used to implement that service.

The Kerberos implementation in Oracle Solaris that is closely based on Kerberos V5 implementation.

While technically different, “Kerberos” and “Kerberos V5” are often used interchangeably in the Kerberos documentation.

Kerberos (also spelled Cerberus) was a fierce, three-headed mastiff who guarded the gates of Hades in Greek mythology.

Kerberos policy A set of rules that governs password usage in the Kerberos service. Policies can regulate principals' accesses, or ticket parameters, such as lifetime.

key 1. Generally, one of two main types of keys:

- A *symmetric key* – An encryption key that is identical to the decryption key. Symmetric keys are used to encrypt files.
- An *asymmetric key* or *public key* – A key that is used in public key algorithms, such as Diffie-Hellman or RSA. Public keys include a private key that is known only by one user, a public key that is used by the server or general resource, and a private-public key pair that combines the two. A private key is also called a *secret* key. The public key is also called a *shared* key or *common* key.

2. An entry (principal name) in a keytab file. See also keytab file.

3. In Kerberos, an encryption key, of which there are three types:

- A *private key* – An encryption key that is shared by a principal and the KDC, and distributed outside the bounds of the system. See also private key.
- A *service key* – This key serves the same purpose as the private key, but is used by servers and services. See also service key.
- A *session key* – A temporary encryption key that is used between two principals, with a lifetime limited to the duration of a single login session. See also session key.

keystore A keystore holds passwords, passphrases, certificates, and other authentication objects for retrieval by applications. A keystore can be specific to a technology, or a location that several applications use.

keytab file A key table file that contains one or more keys (principals). A host or service uses a keytab file in the much the same way that a user uses a password.

kvno Key version number. A sequence number that tracks a particular key in order of generation. The highest kvno is the latest and most current key.

least privilege A security model which gives a specified process only a subset of superuser powers. The least privilege model assigns enough privilege to regular users that they can perform personal administrative tasks, such as mount file systems and change the ownership of files. On the

other hand, processes run with just those privileges that they need to complete the task, rather than with the full power of superuser, that is, all privileges. Damage due to programming errors like buffer overflows can be contained to a non-root user, which has no access to critical abilities like reading or writing protected system files or halting the machine.

limit set The outside limit of what privileges are available to a process and its children.

MAC 1. See message authentication code (MAC).

2. Also called labeling. In government security terminology, MAC is Mandatory Access Control. Labels such as Top Secret and Confidential are examples of MAC. MAC contrasts with DAC, which is Discretionary Access Control. UNIX permissions are an example of DAC.

3. In hardware, the unique system address on a LAN. If the system is on an Ethernet, the MAC is the Ethernet address.

master KDC The main KDC in each realm, which includes a Kerberos administration server, `kadmind`, and an authentication and ticket-granting daemon, `krb5kdc`. Each realm must have at least one master KDC, and can have many duplicate, or slave, KDCs that provide authentication services to clients.

MD5 An iterative cryptographic hash function that is used for message authentication, including digital signatures. The function was developed in 1991 by Rivest. Its use is deprecated.

mechanism 1. A software package that specifies cryptographic techniques to achieve data authentication or confidentiality. Examples: Kerberos V5, Diffie-Hellman public key.

2. In the Cryptographic Framework feature of Oracle Solaris, an implementation of an algorithm for a particular purpose. For example, a DES mechanism that is applied to authentication, such as CKM_DES_MAC, is a separate mechanism from a DES mechanism that is applied to encryption, CKM_DES_CBC_PAD.

message authentication code (MAC) MAC provides assurance of data integrity and authenticates data origin. MAC does not protect against eavesdropping.

message digest A message digest is a hash value that is computed from a message. The hash value almost uniquely identifies the message. A digest is useful for verifying the integrity of a file.

minimization The installation of the minimal operating system that is necessary to run the server. Any software that does not directly relate to the operation of the server is either not installed, or deleted after the installation.

name service scope The scope in which a role is permitted to operate, that is, an individual host or all hosts that are served by a specified naming service such as NIS LDAP.

network application server A server that provides a network application, such as `ftp`. A realm can contain several network application servers.

network policies The settings that network utilities configure to protect network traffic. For information about network security, see "Securing the Network in Oracle Solaris 11.2 ".

nonattributable audit event An audit event whose initiator cannot be determined, such as the AUE_BOOT event.

NTP Network Time Protocol. Software from the University of Delaware that enables you to manage precise time or network clock synchronization, or both, in a network environment. You can use NTP to maintain clock skew in a Kerberos environment. See also clock skew.

PAM Pluggable Authentication Module. A framework that allows for multiple authentication mechanisms to be used without having to recompile the services that use them. PAM enables Kerberos session initialization at login.

passphrase A phrase that is used to verify that a private key was created by the passphrase user. A good passphrase is 10-30 characters long, mixes alphabetic and numeric characters, and avoids simple prose and simple names. You are prompted for the passphrase to authenticate use of the private key to encrypt and decrypt communications.

password policy The encryption algorithms that can be used to generate passwords. Can also refer to more general issues around passwords, such as how often the passwords must be changed, how many password attempts are permitted, and other security considerations. Security policy requires passwords. Password policy might require passwords to be encrypted with the AES algorithm, and might make further requirements related to password strength.

permitted set The set of privileges that are available for use by a process.

policy Generally, a plan or course of action that influences or determines decisions and actions. For computer systems, policy typically means security policy. Your site's security policy is the set of rules that define the sensitivity of the information that is being processed and the measures that are used to protect the information from unauthorized access. For example, security policy might require that systems be audited, that devices must be allocated for use, and that passwords be changed every six weeks.

For the implementation of policy in specific areas of the Oracle Solaris OS, see audit policy, policy in the Cryptographic Framework, device policy, Kerberos policy, password policy, and rights policy.

policy for public key technologies In the Key Management Framework (KMF), policy is the management of certificate usage. The KMF policy database can put constraints on the use of the keys and certificates that are managed by the KMF library.

policy in the Cryptographic Framework In the Cryptographic Framework feature of Oracle Solaris, policy is the disabling of existing cryptographic mechanisms. The mechanisms then cannot be used. Policy in the Cryptographic Framework might prevent the use of a particular mechanism, such as `CKM_DES_CBC`, from a provider, such as DES.

postdated ticket A postdated ticket does not become valid until some specified time after its creation. Such a ticket is useful, for example, for batch jobs that are intended to run late at night, since the

ticket, if stolen, cannot be used until the batch job is run. When a postdated ticket is issued, it is issued as `invalid` and remains that way until a) its start time has passed, and b) the client requests validation by the KDC. A postdated ticket is normally valid until the expiration time of the ticket-granting ticket. However, if the postdated ticket is marked `renewable`, its lifetime is normally set to be equal to the duration of the full life time of the ticket-granting ticket. See also invalid ticket, renewable ticket.

primary The first part of a principal name. See also instance, principal name, realm.

principal 1. A uniquely named client/user or server/service instance that participates in a network communication. Kerberos transactions involve interactions between principals (service principals and user principals) or between principals and KDCs. In other words, a principal is a unique entity to which Kerberos can assign tickets. See also principal name, service principal, user principal.

2. (RPCSEC_GSS API) See client principal, server principal.

principal name 1. The name of a principal, in the format *primary/instance@REALM*. See also instance, primary, realm.

2. (RPCSEC_GSS API) See client principal, server principal.

principle of least privilege See least privilege.

privacy A security service, in which transmitted data is encrypted before being sent. Privacy also includes data integrity and user authentication. See also authentication, integrity, service.

private key A key that is given to each user principal, and known only to the user of the principal and to the KDC. For user principals, the key is based on the user's password. See also key.

private-key encryption In private-key encryption, the sender and receiver use the same key for encryption. See also public-key encryption.

privilege 1. In general, a power or capability to perform an operation on a computer system that is beyond the powers of a regular user. Superuser privileges are all the rights that superuser is granted. A privileged user or privileged application is a user or application that has been granted additional rights.

2. A discrete right on a process in an Oracle Solaris system. Privileges offer a finer-grained control of processes than does `root`. Privileges are defined and enforced in the kernel. Privileges are also called *process privileges* or *kernel privileges*. For a full description of privileges, see the `privileges`(5) man page.

privilege escalation Gaining access to resources that are outside the range of resources that your assigned rights, including rights that override the defaults, permit. The result is that a process can perform unauthorized operations.

privilege model A stricter model of security on a computer system than the superuser model. In the privilege model, processes require privilege to run. Administration of the system can be divided into discrete parts that are based on the privileges that administrators have in their processes. Privileges can be assigned to an administrator's login process. Or, privileges can be assigned to be in effect for certain commands only.

privilege set A collection of privileges. Every process has four sets of privileges that determine whether a process can use a particular privilege. See limit set, effective set set, permitted set set, and inheritable set set.

Also, the basic set set of privileges is the collection of privileges that are assigned to a user's process at login.

privilege-aware Programs, scripts, and commands that turn on and off the use of privilege in their code. In a production environment, the privileges that are turned on must be supplied to the process, for example, by requiring users of the program to use a rights profile that adds the privileges to the program. For a full description of privileges, see the `privileges`(5) man page.

privileged application An application that can override system controls. The application checks for security attributes, such as specific UIDs, GIDs, authorizations, or privileges.

privileged user A user who is assigned rights beyond the rights of regular user on a computer system. See also trusted users.

profile shell In rights management, a shell that enables a role (or user) to run from the command line any privileged applications that are assigned to the role's rights profiles. The profile shell versions correspond to the available shells on the system, such as the `pfbash` version of `bash`.

provider In the Cryptographic Framework feature of Oracle Solaris, a cryptographic service that is provided to consumers. PKCS #11 libraries, kernel cryptographic modules, and hardware accelerators are examples of providers. Providers plug in to the Cryptographic Framework, so are also called *plugins*. For examples of consumers, see consumer.

proxiable ticket A ticket that can be used by a service on behalf of a client to perform an operation for the client. Thus, the service is said to act as the client's proxy. With the ticket, the service can take on the identity of the client. The service can use a proxiable ticket to obtain a service ticket to another service, but it cannot obtain a ticket-granting ticket. The difference between a proxiable ticket and a forwardable ticket is that a proxiable ticket is only valid for a single operation. See also forwardable ticket.

public object A file that is owned by the `root` user and readable by the world, such as any file in the `/etc` directory.

public-key encryption An encryption scheme in which each user has two keys, one public key and one private key. In public-key encryption, the sender uses the receiver's public key to encrypt the message, and the receiver uses a private key to decrypt it. The Kerberos service is a private-key system. See also private-key encryption.

QOP Quality of Protection. A parameter that is used to select the cryptographic algorithms that are used in conjunction with the integrity service or privacy service.

RBAC Role-based access control, the user rights management feature of Oracle Solaris. See rights.

RBAC policy See rights policy.

realm 1. The logical network that is served by a single Kerberos database and a set of Key Distribution Centers (KDCs).

2. The third part of a principal name. For the principal name `jdoe/admin@CORP.EXAMPLE.COM`, the realm is `CORP.EXAMPLE.COM`. See also principal name.

reauthentication The requirement to provide a password to perform a computer operation. Typically, `sudo` operations require reauthentication. Authenticated rights profiles can contain commands that require reauthentication. See authenticated rights profile.

relation A configuration variable or relationship that is defined in the `kdc.conf` or `krb5.conf` files.

renewable ticket Because having tickets with very long lives is a security risk, tickets can be designated as `renewable`. A renewable ticket has two expiration times: a) the time at which the current instance of the ticket expires, and b) maximum lifetime for any ticket. If a client wants to continue to use a ticket, the client renews the ticket before the first expiration occurs. For example, a ticket can be valid for one hour, with all tickets having a maximum lifetime of ten hours. If the client that holds the ticket wants to keep it for more than an hour, the client must renew the ticket. When a ticket reaches the maximum ticket lifetime, it automatically expires and cannot be renewed.

rights An alternative to the all-or-nothing superuser model. User rights management and process rights management enable an organization to divide up superuser's privileges and assign them to users or roles. Rights in Oracle Solaris are implemented as kernel privileges, authorizations, and the ability to run a process as a specific UID or GID. Rights can be collected in a rights profile and a role.

rights policy The security policy that is associated with a command. Currently, `solaris` is the valid policy for Oracle Solaris. The `solaris` policy recognizes privileges and extended privilege policy, authorizations, and `setuid` security attributes.

rights profile Also referred to as a profile. A collection of security overrides that can be assigned to a role or user. A rights profile can include authorizations, privileges, commands with security attributes, and other rights profiles that are called supplementary profiles.

role A special identity for running privileged applications that only assigned users can assume.

RSA A method for obtaining digital signatures and public key cryptosystems. The method was first described in 1978 by its developers, Rivest, Shamir, and Adleman.

scan engine A third-party application, residing on an external host, that examines a file for known viruses.

SEAM The product name for the initial version of Kerberos on Solaris systems. This product is based on the Kerberos V5 technology that was developed at the Massachusetts Institute of Technology. SEAM is now called the Kerberos service. It continues to differ slightly from the MIT version.

secret key See private key.

Secure Shell A special protocol for secure remote login and other secure network services over an insecure network.

security attributes Overrides to security policy that enable an administrative command to succeed when the command is run by a user other than superuser. In the superuser model, the `setuid root` and `setgid` programs are security attributes. When these attributes are applied to a command, the command succeeds no matter who runs the command. In the privilege model, kernel privileges and other rights replace `setuid root` programs as security attributes. The privilege model is compatible with the superuser model, in that the privilege model also recognizes the `setuid` and `setgid` programs as security attributes.

security flavor See flavor.

security mechanism See mechanism.

security policy See policy.

security service See service.

seed A numeric starter for generating random numbers. When the starter originates from a random source, the seed is called a *random seed*.

separation of duty Part of the notion of least privilege. Separation of duty prevents one user from performing or approving all operations that complete a transaction. For example, in RBAC, you can separate the creation of a login user from the assignment of security overrides. One role creates the user. A separate role can assign security attributes, such as rights profiles, roles, and privileges to existing users.

server A principal that provides a resource to network clients. For example, if you `ssh` to the system `central.example.com`, then that system is the server that provides the `ssh` service. See also service principal.

server principal (RPCSEC_GSS API) A principal that provides a service. The server principal is stored as an ASCII string in the form *service*@*host*. See also client principal.

service 1. A resource that is provided to network clients, often by more than one server. For example, if you `rlogin` to the machine `central.example.com`, then that machine is the server that provides the `rlogin` service.

	2. A security service (either integrity or privacy) that provides a level of protection beyond authentication. See also integrity and privacy.
service key	An encryption key that is shared by a service principal and the KDC, and is distributed outside the bounds of the system. See also key.
service principal	A principal that provides Kerberos authentication for a service or services. For service principals, the primary name is a name of a service, such as `ftp`, and its instance is the fully qualified host name of the system that provides the service. See also host principal, user principal.
session key	A key that is generated by the authentication service or the ticket-granting service. A session key is generated to provide secure transactions between a client and a service. The lifetime of a session key is limited to a single login session. See also key.
SHA1	Secure Hashing Algorithm. The algorithm operates on any input length less than 2^{64} to produce a message digest. The SHA1 algorithm is input to DSA.
single-system image	A single-system image is used in Oracle Solaris auditing to describe a group of audited systems that use the same naming service. These systems send their audit records to a central audit server, where the records can be compared as if the records came from one system.
slave KDC	A copy of a master KDC, which is capable of performing most functions of the master. Each realm usually has several slave KDCs (and only one master KDC). See also KDC, master KDC.
software provider	In the Cryptographic Framework feature of Oracle Solaris, a kernel software module or a PKCS #11 library that provides cryptographic services. See also provider.
stash file	A stash file contains an encrypted copy of the master key for the KDC. This master key is used when a server is rebooted to automatically authenticate the KDC before it starts the `kadmind` and `krb5kdc` processes. Because the stash file includes the master key, the stash file and any backups of it should be kept secure. If the encryption is compromised, then the key could be used to access or modify the KDC database.
superuser model	The typical UNIX model of security on a computer system. In the superuser model, an administrator has all-or-nothing control of the system. Typically, to administer the machine, a user becomes superuser (`root`) and can do all administrative activities.
synchronous audit event	The majority of audit events. These events are associated with a process in the system. A non-attributable event that is associated with a process is a synchronous event, such as a failed login.
TGS	Ticket-Granting Service. That portion of the KDC that is responsible for issuing tickets.
TGT	Ticket-Granting Ticket. A ticket that is issued by the KDC that enables a client to request tickets for other services.

ticket An information packet that is used to securely pass the identity of a user to a server or service. A ticket is valid for only a single client and a particular service on a specific server. A ticket contains the principal name of the service, the principal name of the user, the IP address of the user's host, a time stamp, and a value that defines the lifetime of the ticket. A ticket is created with a random session key to be used by the client and the service. Once a ticket has been created, it can be reused until the ticket expires. A ticket only serves to authenticate a client when it is presented along with a fresh authenticator. See also authenticator, credential, service, session key.

ticket file See credential cache.

trusted users Users whom you have decided can perform administrative tasks at some level of trust. Typically, administrators create logins for trusted users first and assign administrative rights that match the users' level of trust and ability. These users then help configure and maintain the system. Also called *privileged users*.

user principal A principal that is attributed to a particular user. A user principal's primary name is a user name, and its optional instance is a name that is used to described the intended use of the corresponding credentials (for example, `jdoe` or `jdoe/admin`). Also known as a user instance. See also service principal.

virtual private network (VPN) A network that provides secure communication by using encryption and tunneling to connect users over a public network.

Index

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z