Audit Trail Administration

CONCURRENT COMPUTER CORPORATION®

0890431-030
August 2003

Revision History:	Level:	Effective With:
Original Release -- July, 1994	000	PowerMAX OS 1.1
Previous Release - May 1995	010	PowerUX 2.1
Previous Release - February 1997	020	PowerMAX OS 4.1
Current Release - August 2003	030	PowerMAX OS 6.0

Preface

Scope of Manual

This guide shows you how to administer the auditing subsystem. Specifically, it has been written to help you understand the job of an auditing administrator and find out exactly how to install, configure, and maintain the auditing subsystem.

The guide assumes that you know such UNIX®[1] system fundamentals as the directory structure and the shell and that you understand basic UNIX system administration. You should also feel comfortable using the computer itself; you should know how to boot it, how to mount file systems, how to install software packages, and how to copy files to archival media, such as magnetic tape. It is helpful if you understand how to write simple shell scripts. You should have the System Administration guides and the *Command Reference* manuals available for reference when you are using this guide.

Structure of Manual

The chapters of this manual are arranged as follows:

- Chapter 1— "Overview of the Auditing Subsystem": This chapter describes the features of the auditing subsystem.
- Chapter 2— "Installing the Auditing Subsystem": This chapter describes the installation of the auditing software from the distribution media and an overview of the configuration process.
- Chapter 3— "Auditable Events": This chapter describes all the auditable events and provides guidelines for choosing a set of events to audit.
- Chapter 4— "Configuring Auditing": This chapter describes the configuration process and how to enable auditing.
- Chapter 5— "Displaying Audit Trail Information": This chapter describes how to display audit data from the audit event log file.
- Chapter 6— "Maintaining the Auditing Subsystem": This chapter describes how to recover audit data after a system crash and the archiving of audit event log files.
- Appendix A - "Summary of Auditable Events and Classes" This appendix is a summary of the auditable events and auditable classes discussed in this manual.

1. UNIX is a registered trademark of The Open Group.

Syntax Notation

The following notation is used throughout this guide:

italic	Books, reference cards, and items that the user must specify appear in *italic* type. Special terms may also appear in *italic*.
`list bold`	User input appears in **`list bold`** type and must be entered exactly as shown. Names of directories, files, commands, options and man page references also appear in **`list bold`** type.
`list`	Operating system and program output such as prompts and messages and listings of files and programs appears in `list` type.
`[ ]`	Brackets enclose command options and arguments that are optional. You do not type the brackets if you choose to specify such option or arguments

Other convention notations include:

- Instructions to the reader to type input usually do not include explicit instructions to press the RETURN key at the appropriate times (such as after entering a command or a menu choice) because this instruction is implied for all UNIX system commands and menus.
- Control characters are shown by the string CTRL-*char* where *char* is a character such as “d” (for example, CTRL-d). To enter a control character, hold down the CTRL key and press the letter shown. Be sure to type the letter exactly as specified: when a lower case letter is shown (such as the “d” in the example above), enter that lower case letter. If a character is shown in upper case (such as CTRL-D), you should enter an upper case letter.
- The system prompt sign shown in examples of interactive sessions is the pound sign (#).

Referenced Publications

The following publications are referenced in this document:

0890429	System Administration (Volume 1)
0890430	System Administration (Volume 2)

Contents

Illustrations

Tables

Screens

1
Overview of the Auditing Subsystem

1
Overview of the Auditing Subsystem

Introduction

This chapter contains an overview of the auditing subsystem. It describes, in general terms:

- the purpose of auditing
- how the auditing subsystem works
- the events mechanism
- the audit event log file
- the auditing commands
- the Operations, Administration, and Maintenance (OAM) interface for auditing

This chapter is intended to give you a high-level understanding of the features of the auditing subsystem and their interactions. In most cases, it does not contain detailed technical information. Later chapters provide such information.

The Purpose of Auditing

The main goal of an auditing facility is to record information about actions that may affect the security of a computer system. In particular, an auditing facility should be able to record any action by any user that may represent a breach of system security. For each action, the auditing facility should record enough information about those actions to verify.

- the user who performed the action
- the exact date and time it was performed
- the success or failure of the action
- the name, type, device, inode and the file system id of any data object involved

The presence of auditing may also deter attempted security breaches, which can allow you to take action to contain the problem. Even if you do not detect a security breach as it occurs, you can use the audit trail to determine the extent of any security problems and to recover from them.

Other features of the operating system can provide information about system activity. The system activity reporting facility, **`sar(1M),`** provides information about the use of system resources, such as the CPU and disks. The process accounting system can provide information of the overall usage of various commands. Both can be useful supplements to the auditing subsystem; if you are not familiar with them, see the “Accounting” and “Managing System Performance” chapters of the manual. However, neither of these facilities provides enough information to meet the requirements of a secure auditing facility. Both report averages for system use, not information on individual actions. They cannot detect specific breaches of security by specific users.

In most cases, security breaches are detected by patterns of usage, not by single actions. A single failed login on a terminal, for example, may indicate that a user had trouble typing a password correctly. Several failed logins on a terminal may indicate that a malicious user is trying to guess a password. To detect such patterns, you often need to record many events that are a normal part of daily activity on the system.

How Auditing Works

The auditing subsystem is an event based system, in which data is recorded whenever an audited event occurs. An event represents a single action that may affect the security of the system. Events are triggered by system calls or by packets received from user level commands. For auditable events triggered by system calls, the kernel writes the audit data in the format of an audit record. For auditable events triggered by user level commands, the command invokes the audit system call `auditdmp()` to record the audit record.

As the audit administrator, you select which events are to be audited. The selected events are recorded and maintained in data structures referred to as event masks.

The following subsections explain the use of event masks and the kernel processing required to determine when an audit record is generated.

Audit Event Masks

The auditing subsystem uses event masks to determine which events are currently being audited. Event masks are data structures that contain 8 words, or 256 bits. There is a bit for each possible event on the system; if the bit is set (that is, has a value of 1), the event is audited. See Appendix A for a list of all events. Figure 1-1 shows a conceptual illustration of an event mask structure.

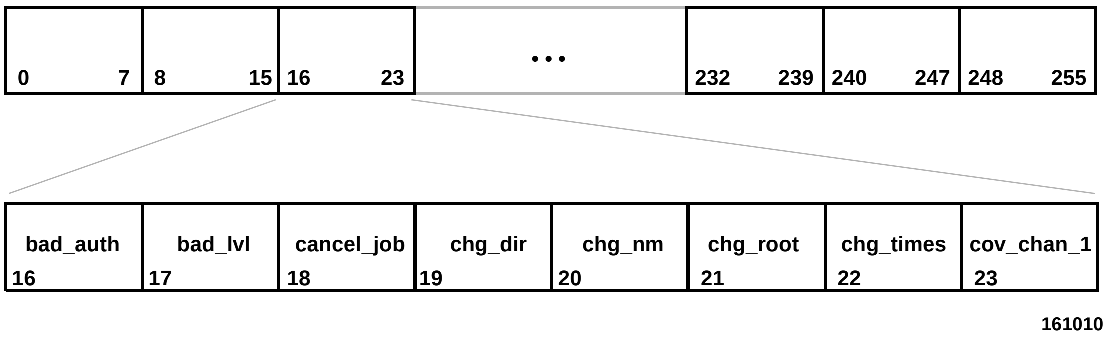


Figure 1-1. Audit Event Mask

NOTE

Some events may not be valid for a particular release. For example, events are added for a new release to cover functionality new for that release. Also, some events may only be valid when a particular package is installed. Events and their ordering in the event mask are carried from release to release to preserve compatibility, enabling the auditing subsystem in the current release to properly process audit trails generated by previous releases of the auditing subsystem. See Appendix A for a list of auditing events.

The auditing subsystem uses several different event masks to determine when to generate an audit record for an event. The auditing subsystem maintains the following event masks:

- a system wide standard event mask
- a user event mask for each active user
- a process event mask for each process
- a lightweight process event mask for each LWP
- a system wide object level event mask

The system event masks contain the set of events that are audited for every process on the system. The system event masks are stored in global kernel space. Please note that kernel processes marked as SYSTEM or ZOMBIE are exempt from auditing. In addition, there are trusted user level processes that bypass the event mask mechanisms. For example, the user level command **`auditmap`** through its invocation of the audit system call `auditevt()` marks itself exempt from auditing.

A user event mask contains the set of events that are audited for a specific user. This mask is set as part of the login procedure or when **`cron`** starts a job for a user; the user mask associated with the specific user is extracted from the **`/etc/security/ia/master`** file and is stored in the `a_useremask` field of the audit process structure. However, if processes owned by the user already exist on the system, such as might occur when a user is already logged into the system from another terminal, then the `a_useremask` field for that user is used, instead. This ensures that all of the processes owned by a user are audited in a consistent manner

Changes to the user's event mask may be made statically or dynamically. To change the user event mask statically, the data in the **`/etc/security/ia/audit`** and **`/etc/security/ia/master`** files are updated. Modification to the user event mask will only take effect for future login sessions or **`cron`** jobs for that user. Note, however, that the effect will be delayed if the change is made while the affected user owns active processes on the system. In this case, as long as the user owns active processes on the system, any new processes created (such as those created by a second login session or a **`cron`** job), will use the `a_useremask` for the existing processes as its user event mask. Only when the user subsequently has no active processes on the system will the static change to the user event mask take effect.

To change the user event mask dynamically, for a user that currently owns active processes on the system, the `a_useremask` field in the audit process structure is updated, using the **`auditset`** command. Dynamic changes are reflected in the current login session, however they are lost once the user no longer has any active processes on the system.

A process event mask contains the set of events audited for all non-exempt processes. The process event mask is the union of the system event and user event mask. The process event mask is stored in the `a_emask` field of the audit process structure and is used to determine if an event is being audited. The process event mask is updated every time the user event mask or the system event mask is modified.

A lightweight process event mask contains the set of events audited for all non-exempt lightweight processes. The LWP Event mask `al_emask` is a copy of the process event mask `a_emask` and is used to determine if an event is being audited.

The standard object level event masks contain the set of auditable events that apply to every object on the system (files, IPCs, etc) whose security level matches an individual level or is within a specified level range. The level event masks may be manipulated by using the **`auditset`** command via the **`auditevt`** system call.

Figure 1-2 illustrates how the event masks are used to determine when to generate an audit record for an event:

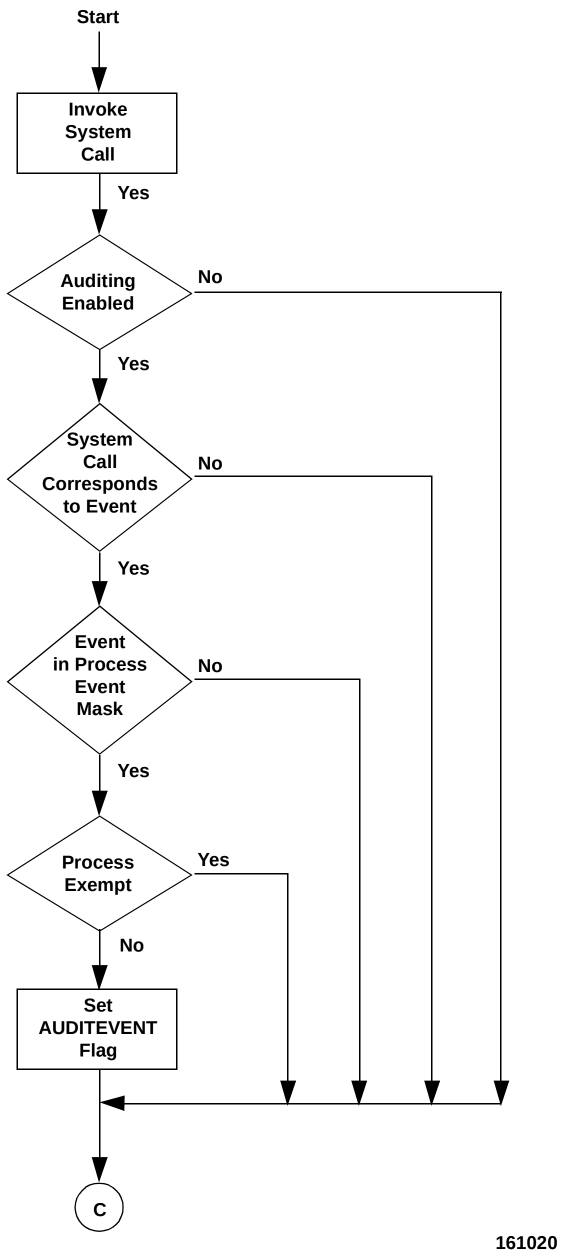


Figure 1-2. System Call Auditing Algorithm (Sheet 1)

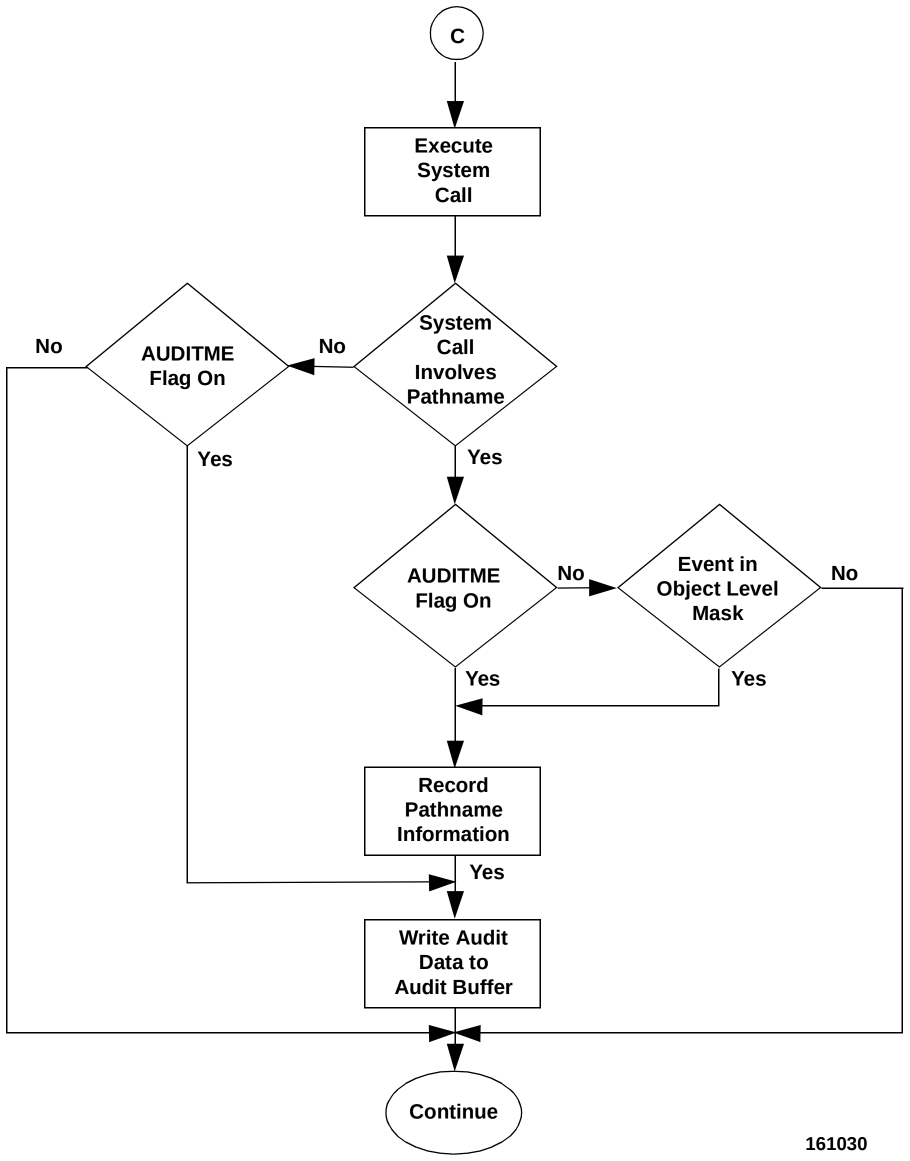

C
Execute System Call
System Call Involves Pathname
No
AUDITME Flag On
No
Yes
Yes
AUDITME Flag On
No
Event in Object Level Mask
No
Yes
Yes
Record Pathname Information
Yes
Write Audit Data to Audit Buffer
Continue
161030

Figure 1-2. Auditing Algorithm (Cont.)

System Call Audit Processing

Whenever a system call is invoked, the kernel invokes the `systrap` routine to perform system call processing. This routine checks a global variable to see if auditing is enabled, and if it is, `systrap` invokes an audit check function to see if the system call corresponds to an auditable event. The audit check function returns either zero (if the system call does not correspond to an auditable event) or a number that represents the auditable event. The auditing subsystem uses this number to check the process event mask and see if the event is being audited. If the event is being audited, the auditing subsystem sets a flag in the process and LWP audit structures to indicate that the event is being audited. Event type, current time and an event sequence number are also saved in the LWP audit structure.

After the audit check function completes processing, the `systrap` routine executes the system call. Any system call that involves a path name calls a series of kernel routines to obtain path name information. If the event is being audited (the flag value in the LWP audit structure is checked) object information is recorded. After the system call returns, if the event is auditable the systrap_cleanup routine calls a recording function that writes the audit record. For reasons of efficiency, the auditing subsystem records much of its data in numeric format. The audit record is usually written to one of a number of audit buffers in the kernel. The tunable parameter `ADT_NBUF` controls the number of audit buffers in the kernel and `ADT_BSIZE` controls the size of each of the buffers. This is defined in **`/etc/conf/mtune.d/audit`**.

The high water mark controls the amount of data stored in an audit buffer before auditing switches to the next available buffer. Whenever an audit record would make the amount of data in a buffer larger than the high water mark the auditing subsystem flags the buffer as writable and switches to a new buffer. It also awakens the audit daemon process which will write the flagged buffer to the audit event log file and return the buffer to the pool of available buffers. If all available buffers are full, the process that is generating the audit record sleeps until a buffer becomes available.

Overview of Auditable Event Types and Classes

An auditable event type (also referred to as an event) represents a single action (either a command system call) that may affect the security of the system. There are two classifications of events: fixed and selectable. See Appendix A for a complete list of events.

Fixed events are always audited when the auditing subsystem is enabled and can not be altered. The fixed events, which are intentionally limited to a subset of all auditable events, include

- all actions relating to the audit subsystem itself
- all attempts to change the system date
- all actions relating to group and user attributes
- changes of init states
- remote access to the system (telnet ftp, etc)

The fixed events represent actions that must be recorded to ensure the integrity and accuracy of the data in the audit event log file (also referred to as the log file). Recording only the fixed events will not give you a complete record of all actions that affect system security.

NOTE

The default audit event masks contains only the fixed event types.

The remaining events are selectable; that is, they are audited only if you select them for auditing. Therefore, you can tailor the information recorded to meet the needs of your site. The user level commands **`auditset,`** **`useradd,`** and **`usermod`** are used to select the events that are to be audited. When you decide which events to record, you are pre-selecting the events. That is, the events are being selected for recording in the audit event log file before they happen.

When you decide which events to report, you are post-selecting the events. That is, you are selecting which events will be reported out of all the ones that have been recorded. The user level command **`auditrpt`** is used to post-select events. Note that if you did not pre-select an event, the audit event log file will not contain any record for that event type.

Typing a long series of events can be both tedious and error prone. To alleviate this problem, the auditing subsystem categorizes event types into event classes. An event class is a name given to a collection of event types. Event classes are defined in the file **`/etc/security/audit/classes.`** A number of pre-defined event classes are provided. See Appendix A for a complete list of event classes. Please note that not all event types have been categorized into event classes. You can edit the **`/etc/security/audit/classes`** file to define new event classes to meet your site's needs.

The "Auditable Events" chapter of this guide contains a complete list of the events and event classes, along with guidelines for selecting events.

Audit Event Log File

Audit records are by default written to one of a number of audit buffers in main memory. When an audit buffer reaches the designated high water mark the audit daemon process switches to the next available buffer and marks the full one as writable. The daemon process writes the audit buffer to the audit event log file and returns the buffer to the pool of available buffers. By default, the audit event log file is in the directory specified by the `AUDIT_DEFPATH` parameter in the **`/etc/default/audit`** file, and is named **`/var/audit`**. You can override the default directory for the audit event log file with the **`auditlog`** command or by reassigning the `AUDIT_DEFPATH` parameter through the use of the **`defadm`** command. The default audit event log file is a regular file, however special character devices, such as a tape drive, may be used as an audit event log file.

The default audit event log file has a seven digit number for a file name. The first four digits are the current month and day, with the final three being a sequence number. Thus the audit event log file **`/var/audit/0415477`** is a log file created on April 15, with a sequence number of 477. For a given day, the sequence number starts at 001 and increments to 999. If all 999 audit event log files exist, a new log file can not be generated unless one or more existing log files are removed. The sequence number of the removed file(s) can be reused. The sequence number is automatically incremented each time auditing is enabled or the log full condition of `SWITCH` occurs. The log full condition is defined as the action to be taken when the current audit event log file has reached its maximum size. The possible values for the log full condition are `SHUTDOWN, DISABLE,` or `SWITCH.` When the value is `SWITCH,` a pre-configured alternate audit event log file becomes the primary audit event log file. If the date has not changed, the new log file has a sequence number one greater than the previous log file. The ability to switch to an alternate log file allows you to audit continuously. The alternate log file may be configured by the **`audit-log`** command or through the `AUDIT_LOGFULL`, `AUDIT_DEFPATH`, `AUDIT_NODE`, and `AUDIT_PGM` parameters in the **`/etc/default/audit`** file.

If you have several machines at a site that generate audit data, it can be very difficult to distinguish which machine generated which audit event log files. To help identify audit event log files, you can configure the auditing subsystem to append a node name to the seven digit log file name (*MMDD###*). A node name is a string containing up to seven characters. A node name can be assigned by the **`auditlog`** command or through the `AUDIT_NODE` parameter in the **`/etc/default/audit`** file. If you use the names (or abbreviations of the names) of your machines as node names, you can easily tell which machine generated a log file.

CAUTION

You should not assign the log file to be a printer, tty, or any device from which data is not retrievable. The audit data is in binary format, which requires translation to ASCII by the command **`auditrpt.`** It is also recommended that you do not direct the output of **`auditrpt`** to a printer in a public area. An unauthorized user may then acquire audit data. If you must print the audit data, take special care to ensure that it is not disclosed to unauthorized users.

Default Settings and Tunable Parameters

You can control several aspects of the auditing subsystem by setting parameters in the **`/etc/default/audit`** file. This file contains the following parameters:

`AUDIT_LOGERR`	The value of this parameter controls the action taken if there is any error involving the auditing subsystem. The allowable values are `DISABLE`, which disables the auditing subsystem, and `SHUTDOWN`, which shuts the computer system down. The value for this parameter upon initial installation is `DISABLE`.
`AUDIT_LOGFULL`	The value of this parameter controls the action taken when the audit event log file becomes full. The allowable values are `DISABLE`, which disables the auditing subsystem, `SHUTDOWN`, which shuts the computer system down, and `SWITCH`, which switches to an alternate audit event log file. The value for this parameter upon initial installation is `DISABLE`.
`AUDIT_PGM`	This parameter defines the absolute path name to an executable file that will be executed if the log full condition of `SWITCH` occurs. The executable can be either a program or a shell script. There is no default value for this parameter upon initial installation.

`AUDIT_DEFPATH`	This parameter defines the absolute path name of the directory for the audit event log files. The value for this parameter upon initial installation is **`/var/audit`**.
`AUDIT_NODE`	This parameter defines the node name to be appended to the system generated audit event log file name. The node name may contain up to seven characters, but must not contain a slash `(/)`. There is no default value for this parameter upon initial installation.

The “Configuring Auditing” chapter provides detailed instructions how to set the values of these parameters.

In addition there are tunable parameters that control the amount of memory allocated in the operating system's kernel for auditing:

`ADT_BSIZE`	This parameter controls the size, in bytes, of an audit buffer. Upon initial installation, the system uses a value of 20 Kilobytes (20480) for this parameter. A large audit buffer may improve system performance by allowing you to store more data in main memory and to reduce the number of disk writes. However, an overly large audit buffer may use space required for other data structures and thus reduce performance.
`ADT_LWP_BSIZE`	This parameter controls the size of the buffer allocated for each lightweight process (LWP). Events generated by a particular LWP are placed in its own buffer, and this buffer is dumped into an audit buffer when it is full. Upon initial installation, the system uses a value of 256 bytes.
`ADT_NBUF`	This parameter controls the number of audit buffers available on the system. Upon initial installation, the system uses a value of two for this parameter.

This is defined in **`/etc/conf/mtune.d/audit`**.

Auditing Commands

The auditing subsystem provides eight user level commands for controlling audit functions. The auditing commands are shown in Table 1-1.

Table 1-1. Auditing Commands

auditcnv	This command is used when the auditing package is first installed to create the audit mask file for the user login interface.
auditfltr	This command provides for the portability of audit event log files among different machine architectures.
auditlog	This command is used to display or set audit event log file attributes.
auditmap	This command creates the audit map files. The audit map files are used by the **auditrpt** command to translate numeric data into character strings
auditoff	This command disables the auditing subsystem.
auditon	This command enables the auditing subsystem. Before you start auditing, you may want to specify the log file characteristics with the **auditlog** command and select auditing criteria with the **auditset** command.
auditrpt	This command is used to display information from the audit event log file.
auditset	This command is used to display or set auditing criteria. The **useradd** and **usermod** commands, which are not part of the auditing subsystem, can also be used to set audit criteria for independent users.

The following subsections describe the auditing commands in more detail.

auditcnv

The **auditcnv** command is invoked automatically during the Audit package installation to create the audit mask file, **/etc/security/ia/audit.** Information from the **/etc/passwd** and **/etc/default/useradd** files is used to assign an initial default audit mask for every user on the system. The data contained in the audit mask file is used by the **creatiadb(1M)** command to create a portion of the **/etc/security/ia/master** file. The **/etc/security/ia/master** file is used during the login procedure to obtain the user event mask for the specific user, except if there are already processes on the system that are owned by that user; for example, the user may be currently logged in on another terminal, or a **cron** job for that user may be currently running. In such cases, the `a_useremask` field for those already existing processes is used. Any time that the **/etc/passwd** and **/etc/default/useradd** files are modified, information in the **/etc/security/ia/audit** and **/etc/security/ia/master** files are updated automatically.

You will never need to invoke this command directly unless the audit mask file is corrupted. If the file is corrupted, you must remove it before running **auditcnv** to create the new audit mask file. The **auditcnv** command is intended for use while the system is in maintenance mode. More information on maintenance mode can be found in the *System Administration* manual.

auditfltr

The **`auditfltr`** command provides for the portability of audit event log files among different machine architectures. The **`auditfltr`** command relies on XDR (External Data Representation) for the description and encoding of the data contained in the audit event log file. An audit event log file may be translated from either native machine format to XDR format or from XDR format to native machine format.

The **`auditfltr`** command is intended for use while the system is in maintenance mode.

auditlog

The **`auditlog`** command displays or sets audit event log file attributes and certain audit actions.

- Display (invoke **`auditlog`** with no options):
  - the current status of auditing
  - the full pathname of the primary audit event log file
  - the audit buffer high water mark
  - the maximum file size of the audit event log file
  - the action to be taken upon a log full condition
  - the action to be taken upon an error condition
  - the full pathname of the alternate audit event log file
  - the program to be run upon a log full condition of `SWITCH`
- Set:
  - the directory of the primary audit event log file (**`-P`** option)
  - a node name for the primary audit event log file (**`-p`** option)
  - the high water mark (**`-v`** option)
  - the maximum size of the audit event log file (**`-x`** option)
  - the log full condition to `SHUTDOWN` (**`-s`** option)
  - the log full condition to `DISABLE` (**`-d`** option)
  - the log full condition to `SWITCH` and set the directory of the alternate audit event log file (**`-A`** option)
  - the log full condition to `SWITCH` and set the node name of the alternate audit event log file (**`-a`** option)
  - a program to be run if the log full condition is already `SWITCH` (**`-n`** option)

The **auditlog** command is discussed in greater detail in the chapter “Configuring Auditing”.

auditmap

The **auditmap** command is used to create the audit map files. By default, these files reside in the directory **/var/audit/auditmap.** The **-m** option allows the auditor to choose a directory where the audit map file(s) will reside. The **auditrpt** command uses the audit map files to translate numeric data into character strings. For example, group IDs and user IDs are recorded as numbers in the audit event log and are displayed by the **auditrpt** command as group names and login names. The audit map files are shown below

auditmap	File identification information, including audit software version, timezone information, privilege mechanism information, system name, machine node name, operating system release and version, and machine type; login names and user IDs; group names and group IDs; event names and event numbers; event classes and their corresponding event types; privilege names and their numbers, and system call names and their numbers

The **auditmap** command is automatically invoked whenever auditing is enabled. If the audit map file(s) already exist they will be renamed. The auditmap file, and each of the LTDB (Level Translation Database) files will be prefixed with an “o”. The new audit map files will then be created.

If any of the information used to create the audit map files changes (for example, new users are added to the system or new event classes are added to **/etc/security/audit/classes**) the audit map files are not automatically updated. The auditor should execute the **auditmap** command to create a new set of audit map files. However, please note that any modification to the audit map files may result in an **auditrpt** failure to translate previously recorded audit data.

The **auditmap** command is discussed in greater detail in the chapter “Displaying Audit Trail Information”.

auditoff

The **auditoff** command disables auditing. Additionally, any log file attributes or auditing actions set by a prior execution of the **auditlog** command is lost.

The **auditoff** command is discussed in greater detail in the chapter “Configuring Auditing”

auditon

The **auditon** command enables auditing. The **auditon** command will create an audit event log file and set audit actions (for example, log full condition) based on the parameters in the **`/etc/default/audit`** file and/or by a previous invocation of the **`auditlog`** command. Once auditing is enabled, the auditing subsystem will begin to record the fixed events and any selectable events specified by the **`auditset, useradd,`** and **`usermod`** commands.

The **`auditon`** command is discussed in greater detail in the chapter "Configuring Auditing".

auditrpt

The **`auditrpt`** command is used to display information from the current audit event log file or one or more specified audit event log files. Information may be extracted and displayed based on one or more of the following criteria:

- time interval (**`-s`** and/or **`-h`** options)
- type of event (**`-e`**, **`-n,`** or **`-z`** option)
- user identity (**`-u`** option)
- object id (**`-f`** option)
- object type (**`-t`** option)
- miscellaneous subtype records (-v option)
- privileges used (**`-p`** option)
- event status - failure or success (**`-a`** option)

The output of the **`auditrpt`** command consists of three sections. The first section is the command line entered by the administrator. The remaining two sections are repeated for each audit event log file that is being processed. The second section contains log file and system identification information. The third section contains the audit record(s) that match the selection criteria specified on the command line. The output of **`auditrpt`** is ASCII text and can be read without any special filters. By default the information is displayed in chronological order, however if requested the most recent data can be displayed first (**`-b`** option). The **`auditrpt`** command has the capability of displaying audit data as it is written to the log file (**`-w`** option). To use this feature the high water mark should be set to 0. This will force all audit records to bypass the audit buffers and be written directly to the log file. Auditing does not need to be enabled to process a log file.

The **`auditrpt`** command has a large number of options that allow you to tailor the reports to meet specific needs. The **`auditrpt`** command is discussed in further detail in the chapter "Displaying Audit Trail Information" and in the **`auditrpt(1M)`** man page.

auditset, useradd, usermod

The **`auditset`** command displays or sets auditing criteria.

- Display:
  - the system audit criteria (**`-d`** or **`-Z`** option)

  - the user audit criteria for all active users only (**`-d`** and **`-a`** options)
  - the user audit criteria for one or more specific active users (**`-d`** and **`-u`** options)
- Set:
  - the system audit criteria (**`-s`** or **`-x`** options)
  - the user audit criteria for all active users (**`-s`** or **`-x`**) and **`-a`** and (**`-e`** or **`-z`**)
  - the user audit criteria for one or more specific active users (**`-s`** or **`-x`**) and **`-u`** and (**`-e`** or **`-z`**)

You can also use the **`useradd`** and **`usermod`** commands to set the user event mask. The **`auditset`** command sets user audit criteria dynamically. That is, the new user mask takes effect during the user's current login session.

The **`useradd`** and **`usermod`** commands set user audit criteria statically. That is, the new user mask will not take effect until there are no active processes owned by that user on the system. Once this has occurred, the new user mask will take effect for all subsequent login sessions and **`cron`** jobs for this user.

The **`auditset, useradd,`** and **`usermod`** commands are discussed in greater detail in the chapter "Auditable Events".

CAUTION

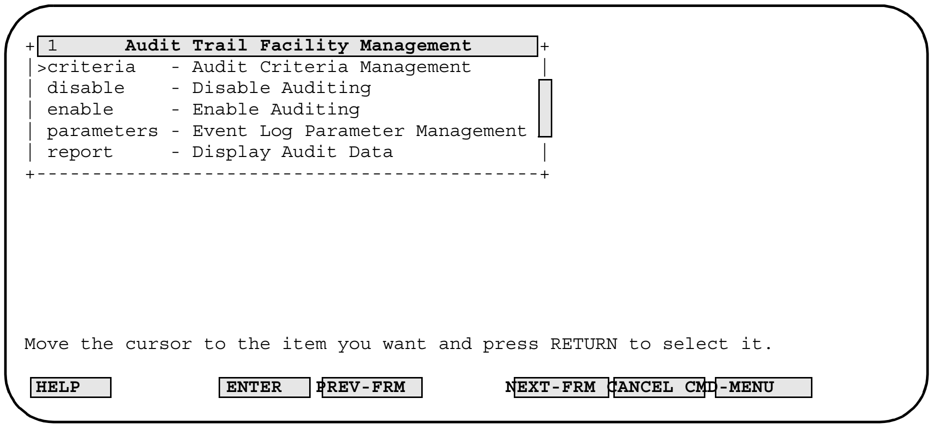
Auditing is considered an essential part of a secure computer system. If auditing is disabled, you will not have a detailed record of any actions that affect the security of your system.

OAM Interface for Auditing

Instead of using the auditing commands, you can control the auditing subsystem by using the OAM menu interface. The OAM system is a menu-driven interface for system administrators. The auditing menu is one of the choices under the security menu. To get the auditing menu, type the following command:

```
sysadm auditing
```

The menu choices for auditing are shown in Screen 1-1.



```
+ 1       Audit Trail Facility Management          +
|>criteria   - Audit Criteria Management            |
| disable    - Disable Auditing
| enable     - Enable Auditing
| parameters - Event Log Parameter Management
| report     - Display Audit Data                   |
+---------------------------------------------------+



Move the cursor to the item you want and press RETURN to select it.

 HELP          ENTER    PREV-FRM        NEXT-FRM CANCEL CMD-MENU
```

Screen 1-1. OAM Menu for Auditing

The relationship between these menu choices and the shell commands is shown in Table 1-2.

Table 1-2. OAM Interface for Auditing

OAM menu item	Task	Shell commands
criteria	setting audit criteria	**`auditset`**
disable	disable auditing	**`auditoff`**
enable	enable auditing	**`auditon`**
parameters	set event log parameters	**`auditlog`**
report	display audit data	**`auditrpt`**

The OAM menu interface is self-documenting. For more information on use of OAM, see the **`sysadm(1M)`** manual page in the online man pages.

2 Installing the Auditing Subsystem

2
Installing the Auditing Subsystem

Introduction

This chapter describes the installation of the auditing software and gives an overview of the configuration process. Specific configuration details are presented in the chapters entitled “Auditable Events” and “Configuring Auditing”.

Installing the Auditing Subsystem

To install the auditing software, use the **pkgadd** command. The argument to the **pkgadd** command is the name of a device that can read the auditing software from its distribution media. For example, if your auditing software is distributed on tape, you would start the installation by typing the following command:

```
pkgadd -d tape_device
```

where *tape_device* is a full pathname or an alias (e.g., `ctape1` or `ctape2`) for the appropriate device.

The **pkgadd** command will prompt you to insert the media containing the auditing software. The **pkgadd** command will prompt you throughout the whole installation process, which is self-documenting. For more information on using the **pkgadd** command, see **pkgadd(1M)** in the online man pages and the “Installing Add-on Software” chapter of the *System Administration* manual.

Versions of the Auditing subsystem are associated with releases of the Operating System as shown in Table 2-1. Discretionary Access Control (DAC) file permission settings for the audit user level commands and system files are listed in Figure 2-1.

Table 2-1. Access Permissions for Audit Files

Item	Owner	Group	Permissions
auditcnv command	root	audit	`r-xr-x---`
auditfltr command	root	audit	`r-xr-x---`
auditlog command	root	audit	`r-xr-x---`
auditmap command	root	audit	`r-xr-x---`
auditon command	root	audit	`r-xr-x---`

Table 2-1. Access Permissions for Audit Files (Cont.)

Item	Owner	Group	Permissions
auditoff command	root	audit	`r-xr-x---`
auditset command	root	audit	`r-xr-x---`
auditrpt command	root	audit	`r-xr-x---`
/etc/security/audit	root	audit	`drwxrwxr-x`
/etc/security/audit/classes	root	audit	`rw-rw-r--`
/etc/security/ia/audit	root	sys	`r--------`
/etc/default/audit	root	sys	`r--r--r--`
/etc/init.d/audit	root	audit	`r--r--r--`
/var/audit	root	audit	`drwxrwx---`
/var/audit/*MMDD###* (log files)	root	audit	`r--r-----`
/var/audit/auditmap	root	audit	`drwxrwx---`
/var/audit/auditmap/*files*	root	audit	`rw-rw----`

The following procedure(s) enable you to install packages distributed on a CD-ROM as you choose.

Note

The system must be configured with the **gr** (generic CD-ROM device driver) **gr(7)** and **cdfs** (CD-ROM filesystem) modules and, in addition, an entry must exist for the CD-ROM device nodes in **/etc/conf/node.d/gr**.

In the example shown below, the CD-ROM device is **/dev/cd/0** and the CD-ROM disc is mounted under **/mnt/cdrom**.

1. Insert the distribution CD-ROM into the CD-ROM drive and mount the file system as shown below.

   ```
   mkdir /mnt/cdrom
   ```

   ```
   mount -F cdfs -r /dev/cd/0 /mnt/cdrom
   ```

2. Get a list of the packages on the CD-ROM disc and verify that the pacage you wish to install is on the CD-ROM disc.

   ```
   pkginfo -d /mnt/cdrom/pkgs.dstream
   ```

3. Multiple packages may be installed at once by specifying more than one package name, as in:

```
 pkgadd -d /mnt/cdrom/pkgs.dstream nsu base-001 \
base-002 base-003
```

4. The package(s) to be installed may be selected from a menu of all the packages available on the CD-ROM disc by not specifying any package name, as in:

```
 pkgadd -d /mnt/cdrom/pkgs.dstream
```

5. View the service release README File to get information about the contents of the service release.

```
pkginfo -Rd /mnt/cdrom/pkgs.dstream
```

Overview of Configuring the Auditing Subsystem

The auditing subsystem as delivered will run without any customization. The default attributes are as follows:

- Auditing is enabled when the system enters multiuser mode.
- Auditing is disabled when the system enters either single user or power off state.
- The audit event log file is in the directory **/var/audit,** with a seven digit number as the file name.
- The auditing subsystem records only the fixed events. These events include all actions relating to the auditing subsystem itself, all attempts to change the system date, all changes relating to user and group attributes, and all changes of init states.
- There is no limit on the size of the audit event log file. The log file will continue to grow until it occupies all available space in the file system that contains the **/var/audit** directory.
- The high water mark is set to the system tunable ADT_BSIZE. This is defined in **/etc/conf/mtune.d/audit**. Audit data will be written to the current audit buffer until it is full or the next audit record exceeds the buffer size.
- The audit buffer size will be set to 20480 bytes.
- The per-LWP buffer size will be set to 256 bytes.
- The number of audit buffers configured will be set to 2.
- The number of levels that can be checked for object level auditing will be set to 4.
- When the log full condition occurs, auditing is disabled. The log full condition is met when the current audit event log file has reached its maximum size.

- When a log error condition occurs, auditing is disabled. A log error condition is met when an error occurs in the auditing subsystem.

NOTE

If the audit event log file is a regular file in the **/** or **/var** file system, it is possible for programs that create files in that file system, such as **mail** in the case of **/var,** to cause the file system to become full unexpectedly. Depending on the way auditing is configured, this may lead to unplanned system shutdowns or to unexpected disabling of auditing. It is recommended that the auditing subsystem have a dedicated file system for the audit event log files. For information on creating a file system, see the *System Administration* manual.

In most cases, you will want to tailor the configuration of the auditing subsystem to meet your site's requirements. In general, there are four main steps in configuring the auditing subsystem:

1. select the events to be audited
2. configure the audit event log file
3. set the tunable parameters
4. edit the **/etc/init.d/audit** script to add your site dependent audit configuration

The rest of this section describes these areas briefly. The "Auditable Events" and "Configuring Auditing" chapters of this guide contain detailed information on the configuration process.

Deciding What Events to Audit

As previously mentioned, there are two classifications of events: fixed and selectable. See Appendix A for a complete list of events.

Fixed events are always audited when the auditing is enabled and can not be altered. The fixed events, which are intentionally limited to a subset of all auditable events, include:

- all actions relating to the audit subsystem itself
- all attempts to change the system date
- all actions relating to group and user attributes
- all definitions and deletions of MAC level names and level identifiers (LIDs)
- changes of init states
- remote access (telnet, jtp, etc)

The fixed events represent actions that must be recorded to ensure the integrity and accuracy of the data in the audit event log file. Recording only the fixed events will not give you a complete record of all actions that affect system security.

The selectable events are only audited if they are first selected and subsequently set by the **`auditset,`** or **`usermod`** or **`useradd`** commands. Therefore, you can tailor the information recorded to meet the needs of your site. The decision as to which events should be audited depends on a number of factors. For example, the amount of resources, in both computer hardware and administrative time, that you can devote to maintaining the audit trail. Auditing all events will provide the highest possible level of security. However, it will also generate a large amount of audit data. If the auditing subsystem is configured for continuous auditing (the log full condition of `SWITCH`) it will be necessary to archive old audit event log files. It is possible to automate this process, if the system has sufficient disk space and a dedicated tape drive for receiving archived log files.

The "Auditable Events" chapter presents detailed information about each of the selectable events. In addition, the chapter contains guidelines for choosing a set of events to be audited. After reading that chapter, you should be able to choose a subset of the selectable events that meets the audit requirements of your site.

Configuring the Audit Event Log File

The attributes of the log file and certain audit actions are controlled by the parameters in the **`/etc/default/audit`** file and/or through the user level command **`auditlog.`** The user level command **`defadm`** is used to modify the parameters `(AUDIT_LOGERR/ AUDIT_LOGFULL/AUDIT_DEFPATH/AUDIT_NODE/AUDIT_PGM`*)* in the **`/etc/ default/audit`** file. The user level command **`auditlog`** may be used to override all but the `AUDIT_LOGERR` parameter. In addition, the **`auditlog`** command may be used to specify the maximum size of the audit event log file, and the high water mark of an audit buffer.

When auditing is enabled the parameters from the **`/etc/default/audit`** file are retrieved and validated. Therefore by assigning site specific values to the parameters, you can reduce the number of options you need to use with the **`auditlog`** command.

The "Configuring Auditing" chapter presents detailed information on how to configure the auditing subsystem.

Tunable Parameters for Auditing

There are a number of tunable parameters you can use to control how the auditing subsystem uses memory. The setting of these parameters also can impact system performance. The parameters are stored in **`/etc/conf/mtune.d/audit`**. A list of the parameters follows:

`ADT_NBUF`	the number of audit buffers that will be configured
`ADT_BSIZE`	the size of each audit buffer
`ADT_LWP_BSIZE`	the size of the buffers allocated to each individual LWP

ADT_NLVLS	the number of levels that can be tracked for object-level criteria

Further information on setting these parameters is contained in the “Configuring Auditing” chapter

The /etc/init.d/audit File

The **`/etc/init.d`** directory contains a series of files, each of which consists of commands that **`init`** executes when the operating system changes init states. The **`audit`** file, as distributed, invokes the **`auditon`** command to enable auditing when the system enters multiuser mode and the **`auditoff`** command to disable auditing when the system enters either single user mode or power off state. If the system is going to multiuser mode and the **`auditon`** command fails the system returns to single user mode.

If you do not want to start auditing automatically, edit this file and comment out the lines that invoke **`auditon.`** If you want to start auditing automatically, you may edit this file and add additional audit user level commands. For example, the inclusion of the commands **`auditlog`** and **`auditset`** will allow your site specific requirements to be automatically configured each time auditing is enabled. Further information on the **`/etc/init.d/audit`** file is contained in the “Configuring Auditing” chapter.

3
Auditable Events

3
Auditable Events

Introduction

This chapter discusses fixed events, selectable events and event classes. After reading this chapter, you should understand what information you can gain from auditing various events. This will allow you to determine the events that you want to audit at your site.

This chapter contains the following sections:

- An overview of events
- A description of the fixed events
- A description of the selectable events
- A description of event classes
- Guidelines for deciding which events to audit

Overview of Events

An auditable event represents a single action (either a command or system call) that may affect the security of the system. There are two classifications of events: fixed and selectable. Fixed events are always audited when the auditing is enabled and can not be altered. Selectable events are only recorded if they have been selected for auditing. The user level commands **`auditset, useradd,`** and **`usermod`** may be used to select events. The ability to select events, allows you to tailor the recording of audit data to the requirements of your site. You can thus record only those events that are important to you and minimize the potential size of the audit event log files.

Events are triggered by certain UNIX system calls or by selected user level processes (for example, **`passwd`**). The majority of user level commands are not audited directly. Instead, an audit record is generated each time that the command executes a system call corresponding to an auditable event. For example, the **`ls`** command may trigger the `open_rd`, `fcntl, status, access, iocntl, pm_denied`, or `sym_status` events.

This chapter describes all the auditable events and the predefined event classes. For many of the selectable events, there is a brief description of the types of security problems you are most likely to detect if the events are audited. Since many events correspond to UNIX system calls, the actions of commands which provide normal user services, such as opening a file for an editor, will be recorded. Thus, many events only provide background information about the usage of your system unless there are unusual patterns of actions.

The following information is recorded for every event and is referred to as the “common” data:

- the time and date the event occurred
- the event type
- the success or failure of the event
- user identification information (real and effective user and group IDs, as well as multiple groups, if present). The string “nobody” is recorded when the information is not available.

The following information is recorded for all events that involve an object (for example, the `open_rd` event) and is referred to as the “object” data:

- the object name
- the object type
- the object’s device number
- the major number component of the object’s device
- the minor number component of the object’s device
- the object’s inode number
- the object’s file system id

The remaining information recorded is unique to each event type. For example the `ulimit` event will have the requested “new limit” recorded. This information is referred to as the “unique” data. The **`auditrpt(1M)`** manual page (see the online man page) contains a description of the unique data recorded for each event type.

It is also possible for application programs with the p_`auditwr` privilege to write miscellaneous records to the audit event log file. Audit records created by application programs are of the event type `misc`. The application program invokes the audit system call `auditdmp()` to record the audit record.

Events may be selected for auditing either before or after auditing is enabled. If you add or delete events for auditing with **`auditset,`** the changes take effect immediately. If you use the **`useradd`** or **`usermod`** commands, to change the events recorded for a user, the changes take effect the next time the user logs in and there are no processes owned by that user already on the system. If processes owned by the user already exist, the new processes will use the same user mask as the existing processes.

As discussed in the chapter “Overview of the Auditing Subsystem”, the auditing subsystem maintains a system wide event mask, a user event mask for each active user, a process event mask for each process, a lightweight process event mask for each LWP, and an object level event mask. The **`auditset`** command is discussed in detail in the “Configuring Auditing” chapter.

Fixed Events

As previously mentioned, fixed events are always audited when auditing is enabled and can not be altered. Therefore, when auditing is enabled, the system wide event masks will always contain the fixed events. The fixed events, which are intentionally limited to a subset of all auditable events, include

- all actions relating to the audit subsystem itself
- all attempts to change the system date
- all actions relating to group and user attributes
- changes of init states

The fixed events represent actions that must be recorded to ensure the integrity and accuracy of the data in the audit event log file. Recording only the fixed events will not give you a complete record of all actions that affect system security.

The fixed events are presented in table format. For each event, Table 3-1 lists the event, a brief description of the event, the name of the manual page for the command or system call that triggers the event, and an indication if the event may be used for object level auditing.

Table 3-1. Fixed Events

Event	Description	Man Page	Object Audit
`add_grp`	adding groups	**`groupadd(1M)`**	N
`add_usr`	adding user attributes	**`useradd(1M)`**	N
`add_usr_grp`	adding group members	**`addgrpmem(1M)`, `useradd(1M)`, `usermod(1M)`**	N
`audit_buf`	set audit buffer attributes	**`auditbuf(2)`**	N
`audit_ctl`	enable or disable auditing	**`auditoff(1M)`, `auditon(1M)`, `auditctl(2)`**	N
`audit_dmp`	`auditdmp` failures	**`auditdmp(2)`**	N
`audit_evt`	set auditable events	**`auditset(1M)`, `auditevt(2)`**	N
`audit_log`	set log file attributes	**`auditlog(1M)`, `auditlog(2)`**	N
`audit_map`	create audit map file	**`auditmap(1M)`**	N

Table 3-1. Fixed Events (Cont.)

Event	Description	Man Page	Object Audit
`date`	change the date	**`adjtime(2)`**, **`posix_clocks(2`**, **`stime(2)`**, **`settimeofday(2)`**	N
`init`	change of init state	**`init(1M)`**	N
`mod_grp`	change group information	**`groupmod(1M)`**	N
`mod_usr`	change user information	**`usermod(1M)`**	N

The `audit_buf`, `audit_ctl`, `audit_dmp`, `audit_evt`, `audit_log`, and `audit_map` events are recorded to ensure that you can always verify the state of the auditing subsystem and the correctness of the log file. The date of an event is an important part of the audit record. Therefore, all changes to the system date (the `date` event) are recorded to ensure the integrity of the audit records. The `add_grp`, `add_usr`, `add_usr_grp`, `mod_grp`, and `mod_usr` events are recorded to ensure that you can always verify the accuracy of the user and group attributes recorded in the audit event log file.

If any of the user, or group information changes on the system, the auditor should execute the **`auditmap`** command to create new audit map files. However, please note that any modification to the audit map files may result in **`auditrpt`**'s failure to translate previously recorded audit data.

An audit record generated by a fixed event will always contain the "common" data. The **`auditrpt(1M)`** manual page contains a description of the "unique" data recorded for each fixed event.

Selectable Events

As previously mentioned, selectable events are only recorded if they have been selected for auditing. Recording all selectable events provide the highest potential level of security; however, it generally makes the log file grow very rapidly. Using the user level commands **`auditset, usermod`** and **`useradd`** the auditor may select events specific to the site's requirements.

There are two event keywords, `all` and `none,` which enable you to audit every selectable event or none, respectively. When setting the system wide event masks or a user event mask, the keyword `all` implies all fixed and selectable events will be recorded and the keyword `none` implies only fixed events will be recorded. For the object level event mask the keyword `all` implies all selectable events related to objects will be recorded and the keyword `none` implies no events will be recorded.

The remaining subsections discuss all the selectable events, grouped by functional areas. Information on each event is presented in table format. The tables include the event, a brief description of the event, and the name of the manual page for the command or system call that triggers the event.

Access Control Events

The following events record actions related to file access, control and creation. In general, these events can be expected to occur during normal system operations. However, they may indicate a security problem if they occur in unusual patterns. For example, several changes of the access permissions to the same object may indicate that two processes may be attempting to signal each other, based on the accessibility of a file.

Note that much of the security of the system depends on proper use of the access control mechanisms. If access permissions are not set appropriately, it is possible for users to see data that they should not be allowed to view. It is a good idea to audit all events in this group to verify that the system's access permissions are always set appropriately.

LWP Events

Individual lightweight processes may be controlled through system calls. These events are shown in Table 3-2.

Table 3-2. Lightweight Process Events

Event	Description	Man Page	Object Audit
`lwp_create`	create a new process	**`_lwp_create(2)`**	N
`lwp_setbias`	change processor for `lwp`	**`cpu_bias(2)`**	N
`lpw_setrun`	change processor for `lwp`	**`cpu_bias(2)`**	N
`lwp_exit`	terminate `lwp`	**`_lwp_exit(2)`**	N
`lwp_kill`	post signal to `lwp`	**`_lwp_kill(2)`**	N

Discretionary Access Control (DAC)

The events listed in Table 3-3 record changes in the Discretionary Access Control (DAC) permissions for objects (that is, file permissions). Access permissions are set by object owners at their discretion.

Table 3-3. Discretionary Access Control Events

Event	Description	Man Page	Object Audit
`dac_mode`	change mode of an object	**`chmod(2)`**, **`fchmod(2)`**	Y
`dac_own_grp`	change owner of an object	**`chown(2)`**, **`fchown(2)`**, **`lchown(2)`**,	Y

Directory and File Access

The events listed in Table 3-4 record changes in directory and file access. The occurrence of directory and file access events are part of the normal activity of a system. However, these events may indicate problems if they occur in unusual patterns. For example, it is possible for two processes to signal each other, based on the accessibility of a file. These signals are used to pass data between the processes in violation of access control permissions. In this case, a process would have an unusual number of `access` events for the same object, and the events would alternate between success and failure.

Table 3-4. Directory and File Access Events

Event	Description	Man Page	Object Audit
`access`	determine accessibility of a file	**`access(2)`**	Y
`chg_times`	change file access and modification times	**`utime(2)`**	Y
`open_rd`	open an object for reading	**`open(2)`**	Y
`open_wr`	open an object for writing	**`open(2)`**	Y
`recvfd`	receive file descriptor	NA	Y
`status`	get file status	**`stat(2)`**, **`fstat(2)`**, **`xstat(2)`**, **`fxstat(2)`**	Y

Directory and File Creation

The events listed in Table 3-5 record changes in directory and file creation.The occurrence of directory and file creation events are part of the normal activity of a system. However, these events may indicate problems if they occur in unusual patterns.

Symbolic Links

The events shown in Table 3-6 record actions that involve symbolic links. Symbolic links are inodes that contain the path name of another file system object. References to the symbolic link become references to the named object. Symbolic links can be used to create links between objects that span file systems.

.

Table 3-5. Directory and File Creation Events

Event	Description	Man Page	Object Audit
`create`	create a new file system object	**`creat(2)`**	Y
`link`	create a link to an object	**`link(2)`**	Y
`mk_dir`	make a directory	**`mkdir(2)`**	Y
`rm_dir`	remove a directory	**`rmdir(2)`**	Y
`unlink`	unlink an object	**`unlink(2)`**	Y

Table 3-6. Symbolic Link Events

Event	Description	Man Page	Object Audit
`sym_create`	create a symbolic link	**`symlink(2)`**	Y
`sym_status`	get status of symbolic link	**`lstat(2)`, `symlink(2)`, `lxstat(2)`**	Y

Changes of Path

The events shown in Table 3-7 record actions that involve path changes.

Table 3-7. Path Change Events

Event	Description	Man Page	Object Audit
`chg_dir`	change working directory	**`chdir(2)`**, **`fchdir(2)`**	Y
`chg_root`	change root directory	**`chroot(2)`**	Y
`chg_nm`	change file name	**`rename(2)`**	Y

System Administration Events

The events shown in Table 3-8 are triggered by commands or system calls that require privileges and are usually executed only by administrators.

Table 3-8. System Administration Events

Event	Description	Man Page	Object Audit
`acct_off`	disable accounting	**`acct(2)`**	N
`acct_on`	enable accounting	**`acct(2)`**	N
`acct_sw`	switch accounting files	**`acct(2)`**	N
`file_priv`	change privileges on a file	**`filepriv(2)`**	Y
`lp_admin`	administrative use of lp system	**`lpadmin(1M)`**	N
`mk_node`	make a special file	**`mknod(2)`**, **`xmknod(2)`**	Y
`mount`	mount a device or file system	**`mount(2)`**	Y
`pm_denied`	failed use of privilege	NA	N
`sched_lk`	lock a process into memory	**`plock(2)`**, **`memcntl(2)`**	N
`sched_fp`	fixed priority scheduler operations	**`priocntl(2)`**	N
`sched_fc`	fixed class scheduler operations	**`priocntl(2)`**	N
`sched_ts`	time-sharing scheduler operations	**`priocntl(2)`**	N

Table 3-8. System Administration Events (Cont.)

Event	Description	Man Page	Object Audit
`setrlimit`	set resource limits	**`setrlimit(2)`**	N
`tfadmin`	administrative command	**`tfadmin(1M)`**	N
`ulimit`	resource limits	**`ulimit(2)`**	N
`umount`	unmount a device or file system	**`umount(2)`**	Y

Line Printer System Events

The events shown in Table 3-9 record actions related to the line printer system.

Table 3-9. Line Printer Events

Event	Description	Man Page	Object Audit
`cancel_job`	cancel a print job	**`cancel(1)`**, **`lpsched(1M)`**	N
`lp_admin`	administrative use of lp system	**`lpadmin(1M)`**, **`lpfilter(1M)`**, **`lpforms(1M)`**, **`lpmove(1M)`**, **`lpsched(1M)`**, **`lpshut(1M)`**, **`lpusers(1M)`**	N
`lp_misc`	miscellaneous use of lp system	**`lpsched(1M)`**	N
`prt_job`	start/end of a print job	**`lp(1)`**	N

Interprocess Communication Events

The events shown in Table 3-10 record actions related to interprocess communication on the system. The various interprocess communications mechanisms (messages, semaphores, and shared memory) provide an efficient means of sharing data. If malicious users gain access to an IPC object, they can read a great deal of data in a short time.

Table 3-10. Interprocess Communication Events

Event	Description	Man Page	Object Audit
`msg_ctl`	message control operations	**`msgctl(2)`**	Y
`msg_get`	get message queue	**`msgget(2)`**	Y
`msg_op`	message operations	**`msgop(2)`**	Y
`pipe`	create a unnamed pipe	**`pipe(2)`**	Y
`shm_bind`	shared memory binding	**`shmbind(2)`**	Y
`sem_ctl`	semaphore control operations	**`semctl(2)`**	Y
`sem_get`	get the set of semaphores	**`semget(2)`**	Y
`sem_op`	semaphore operations	**`semop(2)`**	Y
`shm_ctl`	shared memory control operations	**`shmctl(2)`**	Y
`shm_get`	get shared memory identifier	**`shmget(2)`**	Y
`shm_op`	shared memory operations	**`shmop(2)`**	Y

Process Control Events

The events shown in Table 3-11 record actions related to the control of processes in the operating system. The majority of these events can be expected to occur frequently during normal use of the system. Therefore, the presence of these events in the log file does not automatically indicate a security problem. However, malicious users may try to use the `setgid` or `setuid` system calls to read data that they are not normally allowed to access. You may want to audit the set_gid and set_uid events to ensure that these system calls are always being used correctly.

Table 3-11. Process Control Events

Event	Description	Man Page	Object Audit
exec	execute an object	**exec(2), exece(2)**	N
exit	terminate a process	**exit(2)**	N
fork	create a new process	**fork(2), vfork(2), fork1(2), forkall(2)**	N
kill	post a signal	**kill(2), sigsendset(2)**	N
set_gid	change group ID	**setgid(2)**	N
set_grps	set multiple groups	**setgroups(2)**	N
set_pgrps	set process groups	**setpgrp(2), setpgid(2)**	N
set_sid	assign a session ID	**setsid(2)**	N
set_uid	change user ID	**setuid(2)**	N

User Authentication Events

All of the events shown in Table 3-12 provide valuable information about the users on your system. The bad_auth event records all attempts to login that fail because of an invalid password or logname. A high number of such failures may indicate that someone who does not know a valid logname or password is trying to log in. It is especially important to audit this event if you have dial-up terminal ports on your system. If you chose to audit the bad_auth, bad_lvl, and def_lvl events, they must be set in the system wide event mask, not in a specific user event mask. A user event mask will take effect once a user has successfully logged on.

The bad_lvl and def_lvl events are recorded only on systems that have the Enhanced Security Utilities installed.

Table 3-12. User Authentication Events

Event	Description	Man Page	Object Audit
bad_auth	bad logname/password	**login(1)**	N
bad_lvl	bad login level	**login(1)**	N
cron	cron job	**cron(1M)**	N

Table 3-12. User Authentication Events (Cont.)

Event	Description	Man Page	Object Audit
`def_lvl`	change a user's default level	**`login(1)`**	N
`login`	use a login scheme	**`login(1)`**	N
`logoff`	terminate a login session	**`ttymon(1)`**	N
`passwd`	change password	**`passwd(1)`**	N

I/O Control Events

The events shown in Table 3-13 record input/output control operations.

Table 3-13. I/O Control Events

Event	Description	Man Page	Object Audit
`fcntl`	file control operations	**`fcntl(2)`**	Y
`iocntl`	control I/O operations	**`ioctl(2)`**	Y

Dynamic Loadable Module Events

The events shown in Table 3-14 record Dynamically Loadable Module (DLM) operations.

Table 3-14. Dynamically Loadable Module Events

Event	Description	Man Page	Object Audit
`modadm`	register a module	**`modadm(2)`**	N
`modload`	load a module	**`modload(2)`**	N
`modpath`	modify module search path	**`modpath(2)`**	Y
`moduload`	unload a module	**`moduload(2)`**	N

Processor State Events

Processors may be brought online and offline as required. The events shown in Table 3-15 record actions related to bring processes online and taking them offline.

Table 3-15. Processor State Events

Event	Description	Man Page	Object Audit
`keyctl`	allow processors to be used	**`keyctl(2)`**	N
`online`	take processor on/ offline	**`online(2)`**	N

Event Classes

An event class is a name given to a collection of event types. An event class will allow you to refer to a related group of events with a single name. For example, the event class `file_make` contains the events `create, link, mk_node, sym_create,` and `unlink,` which are all the events related to file creation or deletion. Event classes may be used as input to the **`auditset`** and **`auditrpt`** commands.

Event classes are defined in the **`/etc/security/audit/classes`** file. The following is an entry from this file:

```
alias file_make
    create link mk_node sym_create unlink
```

Each line begins with the word `alias,` followed by the event class name and then by a list of the events that comprise that class. There should be a space between the events listed for the class. A new line ends the list. The entries are similar to those in a **`.mailrc`** file.

Site specific event classes may be added to the **`/etc/security/audit/classes`** file. The administrator may edit the file with any standard editor. If you edit the file, do not delete the predefined event classes. There is no limit to the number of event classes that can be defined. It is important to note that not all event types are contained in the predefined event classes.

The following are the predefined event classes:

`acct`	This event class includes the `acct_off, acct_sw,` and `acct_on` events.
`audit`	This event class contains the `audit_buf, audit_ctl, audit_dmp, audit_evt, audit_log,` and `audit_map` events. All the events in this class are fixed.

`dac`	This event class contains the `dac_mode`, `dac_own_grp`, `file_acct`, and `ipc_acl` events.
`device`	This event class contains the `disp_attr`, `mount`, `set_attr`, `set_lvl_rng`, and `umount` events.
`dir_access`	This event class contains the `access`, `chg_dir`, `chg_root`, `chg_times`, `status`, and `sym_status` events.
`dir_make`	This event class contains the `link`, `mk_dir`, `mk_mld`, `rm_dir`, `sym_create`, and `unlink` events.
`file_attr`	This event class contains the `add_grp`, `add_usr`, `add_usr_grp`, `mod_grp`, and `mod_usr` events. All the events in this class are fixed.
`file_make`	This event class contains the `create`, `link`, `mk_node`, `sym_create`, and `unlink` events.
`id_auth`	This event class contains the `bad_auth`, `bad_lvl`, `cron`, `def_lvl`, `login`, and `passwd` events.
`io_cntl`	This event class contains the `fcntl` and `iocntl` events.
`mac_name`	This event class contains the `assign_lid`, `assign_nm`, `deactivate_lid` and `del_nm` events. All of the events in this class are fixed.
`mac_obj`	This event class contains the `file_lvl` event.
`module`	This event class contains the `modadm`, `modload`, `modpath`, and `moduload` events.
`msg`	This event class contains the `msg_ctl`, `msg_get`, and `msg_op` events.
`path`	This event class contains the `chg_dir` and `chg_root` events.
`printer`	This event class contains the `cancel_job`, `lp_admin`, `lp_misc`, `page_lvl`, `prt_job`, `prt_lvl`, and `trunc_lvl` events.
`priv`	This event class contains the file_`priv` and `pm_denied` events.
`process`	This event class contains the `exec`, `exit`, `fork`, `kill`, `set_gid`, `set_grps`, `set_pgrps`, `set_sid`, and `set_uid` events.
`res_limit`	This event contains the `setrlimit` and `ulimit` events.
`sched`	This event class contains the `sched_lk`, `sched_fp`, `sched_fc`, and `sched_ts` events.
`sem`	This event class contains the `sem_ctl`, `sem_get`, and `sem_op` events.

`shm`	This event class contains the `shm_ctl`, `shm_get`, and `shm_op` events.
`sym_link`	This event class contains the `sym_create` and `sym_status` events.
`tli`	This event class includes the `bind`, `connect`, `listen`, `accept` and `optmgmt` events.
`use_lwp`	This event class contains the `lwp_create`, `lwp_exit`, and `lwp_kill` events.

Deciding Which Events to Audit

The best strategy for determining which events to audit is to start by auditing all events for a reasonable period of time (a few days, for example). At the end of that time, you should examine the audit event log files in detail to see the following:

- which events occur most frequently
- which events seem to reveal unusual patterns
- which events seem to give the most information about important actions at your site

By looking at actual activity at your site, you can determine whether recording an event produces useful information or not. You can then decide whether you want to continue auditing the events. Auditing subsystems for trusted computer systems are expected to be able to record a variety of different actions. These actions are listed below, along with the specific selectable events corresponding to these actions in the auditing subsystem for the operating system.

- use of identification and authentication mechanisms

  The corresponding events are `bad_auth`, `bad_lvl`, `cron`, `def_lvl`, `login` and `passwd`. The `set_uid` event, though it is not in the set of user identification and authentication events, also provides information on the identity of the user executing a process.

- actions of computer operators and system administrators

  The corresponding events are `acct_off`, `acct_on`, `acct_sw`, `file_priv`, `lp_admin`, `mk_node`, `mk_mld`, `mount`, `sched_lk`, `sched_fp`, `sched_fc`, `sched_ts`, `tfadmin`, `ulimit`, and `umount`. In addition, all the fixed events provide information in this area.

- production of printed output

  The corresponding event is `prt_job`; the `cancel_job` event also provides useful information in this area.

- introduction of objects into a user’s address space

The corresponding events are the interprocess communication events and the `create`, `link`, `mk_dir`, `open_rd`, `open_wr`, and `sym_create` events.

- deletion of objects from a user's address space

  The corresponding events are `unlink` and `rm_dir`.

- any override of the security level markings produced on printer output.

  The corresponding event is the `prt_lvl` event; the `page_lvl` and `trunc_lvl` events also provide important information in this area.

- any setting of security levels or security level ranges for I/O devices or communications channels.

  The corresponding events are `file_lvl`, `set_attr`, `set_lvl_rng`. The `disp_attr` event also provides useful information in this area.

- all security relevant events

  The definition of a security relevant event varies, depending on the needs of a site. Of the events not listed above, the `pm_denied` event and the events associated with discretionary access controls will probably be relevant to the security of most sites.

Even if you need to minimize the amount of data recorded, you should consider auditing the following set of events.

- All events relating to system configuration activities.
- All events relating to use of privileges
- All events relating to user identification and authentication.
- The `set_uid` event.

4
Configuring Auditing

4
Configuring Auditing

Introduction

This chapter provides detailed information on the following:

- setting the auditing tunable parameters
- setting the audit event log file characteristics with the **`auditlog`** command.
- setting the audit criteria with the **`auditset, useradd,`** and **`usermod`** commands.
- enable auditing with the **`auditon`** command.
- disable auditing with the **`auditoff`** command.
- the use of the **`/etc/init.d/audit`** file

The examples shown in this chapter assume that you are typing commands at the shell level on a running system.

Setting the Tunable Parameters

The settings of the tunable parameters for auditing control how much kernel memory is used by the auditing subsystem. The amount of memory used by auditing may influence performance. Setting these parameters allows you to decide the appropriate trade-off between the amount of memory consumed by the auditing subsystem, and hence, the kernel, and the performance of your system.

NOTE

Any changes you make to these tunable parameters will not take effect until the kernel is subsequently rebuilt, and the system then booted using the new kernel.

The file containing the tunable parameters for auditing is **`/etc/conf/mtune.d/audit`**. Each line of the file should contain a tunable name, and three numbers, as shown in Screen 4-1.

```
cat /etc/conf/mtune.d/audit
ADT_BSIZE       20480   10240   20480
ADT_LWP_BSIZE   256     256     20480
ADT_NBUF        2       0       5
```

Screen 4-1. Audit Tunable Parameters

The three numbers, left to right, represent the current setting, recommended minimum value, and recommended maximum value, respectively, for the named tunable parameter. The second and third numbers on each line, as recommended values, should not be changed. The first of the three numbers, the current setting, may be changed as you wish. However, the current setting should never be less than the recommended minimum value, as this may severely degrade system performance. The advisability of changing the current setting of any of these tunable parameters to a value exceeding the recommended minimum depends heavily on the setup of your system, especially the amount of memory and the speed of input/output operations. Although exceeding the recommended maxima may improve performance on some systems, the recommended values are adequate for most cases.

The tunable parameters, and their effect, are as follows.

`ADT_NBUF`
: The auditing subsystem buffers the writing of audit records to the audit trail to improve performance. Instead of writing each record to the audit trail as it is completed, records are written to an audit buffer in memory, and a write to the audit trail is performed only when the buffer is full, or the amount of data in the buffer reaches the high-water mark set by using the **`auditlog`** command. This combines several write operations into one and, thus, saves time.

Each buffer you allocate takes up memory, making the kernel larger. If you do not allocate enough buffers, performance will suffer as processes wishing to write audit records must wait for buffers to be emptied and made available. Allocating too many buffers simply wastes memory, which can also hurt system performance. The default value for this parameter is 2, with a recommended range of 0 to 5.

`ADT_BSIZE`
: This parameter controls the size of each audit buffer. If the buffer size is too small, the system will spend proportionally more time emptying audit buffers, and performance will suffer. Larger and larger buffer sizes give a diminishing improvement in performance, and increase the amount of data that can be lost during a system crash. The **`crash`** command can be used to retrieve audit data in such cases. The default size for a buffer is 20480 bytes. The recommended size range is 10240 to 20480 bytes.

`ADT_LWP_BSIZE`
: A buffer is allocated for each light weight process (LWP). Audit records for a particular LWP are written into its buffer, which, when full, is dumped into the audit buffers for the entire system. This second layer of buffering limits contention for the global buffers as records are being assembled.

The default size for these buffers is 256 bytes. This is just large enough to hold an audit record in most cases. Reducing the size of these buffers below 256 bytes is not recommended, as the buffers will not be large enough to hold an entire audit record, resulting in an increase in the number of writes to the global audit buffers and a consequent decrease in system performance. Because a buffer is allocated for each LWP, a large buffer size could take up a substantial amount of memory on a busy system, and so degrade performance. The size of these buffers should never exceed the size of the global audit buffers. The recommended range for the LWP buffer size is 256 to 20480 bytes.

Configuring the Log File and Audit Actions

The attributes of the audit event log file and certain audit actions are controlled by the parameters in the **`/etc/default/audit`** file and through the user level command **`auditlog.`** The user level command **`defadm`** is used to assign values to the parameters in the **`/etc/default/audit`** file. The following subsections will describe in detail how to use both the **`/etc/default/audit`** file and the **`auditlog`** command. Although the **`/etc/default/audit`** file and the **`auditlog`** command overlap in certain functionality they are both needed to achieve a good working configuration. For example, the parameters in the **`/etc/default/audit`** file are intended to be used as default settings. Each time auditing is enabled these values are read, validated and used to establish the log file and certain audit actions. The **`auditlog`** command provides you with the ability to override these values. Additionally, the **`auditlog`** command offers the administrator the ability to display current settings and to define the maximum size of a log file, the high water mark, and a location and node name of an alternate log file which differs from the primary log file. The **`auditlog`** manual page can provide further detail

The /etc/default/audit File

The parameters in the **`/etc/default/audit`** file control certain default actions of the auditing subsystem and log file attributes. The **`auditlog`** command may be used to override all but the `AUDIT_LOGERR` parameter. The parameters are as follows:

`AUDIT_DEFPATH`	This parameter defines the absolute path name of either the directory where the log file will reside or the special character device which will serve as the log file. The default for the distributed system is **`/var/audit.`**
`AUDIT_LOGERR`	The value of this parameter controls the action taken if there is any error involving the auditing subsystem. The allowable values are `DISABLE,` which disables the auditing subsystem, and `SHUTDOWN,` which shuts the computer system down. The value for this parameter in the distributed system is `DISABLE`.
`AUDIT_LOGFULL`	The value of this parameter controls the action taken when the audit event log file becomes full. The allowable values are `DIS-`

	ABLE, which disables the auditing subsystem, SHUTDOWN, which shuts the computer system down, and SWITCH, which switches to an alternate audit event log file. The value for this parameter in the distributed system is DISABLE .
AUDIT_NODE	This parameter defines the node name to be appended to the system generated audit event log file name. The node name may contain up to seven characters, but must not contain a slash (/). There is no default value for this parameter in the distributed system.
AUDIT_PGM	This parameter defines the absolute path name to an executable file that will be executed if the log full condition of SWITCH occurs. The executable can be either a program or a shell script. There is no default value for this parameter in the distributed system.

The values of the preceding parameters are set by using the **defadm** command as follows, replacing *parameter* and *value* with appropriate strings:

defadm audit *parameter=value*

For example, to set auditing to be disabled upon a log full condition, enter the following command:

```
defadm audit AUDIT_LOGFULL=DISABLE
```

Be careful when using **defadm** to change the values of parameters, because the **defadm** command does not validate the values specified for the parameters. Validation of the parameters is done when auditing is enabled. If the values of the parameters are invalid, the auditing subsystem takes the following actions:

- If the value for AUDIT_LOGFULL is invalid, a warning message is displayed and the DISABLE condition is set.
- If the value for AUDIT_LOGERR is invalid, a warning message is displayed and the DISABLE condition is set.
- If the value for AUDIT_DEFPATH is invalid, a warning message is displayed and the default directory **/var/audit** is set.
- If the value for AUDIT_NODE is invalid, no default value is used.
- If the value for AUDIT_PGM is invalid, no default value is used.

Specifying the Type and Location of the Audit Event Log File

The audit event log file may be either a regular file or a character special device. The value of the parameter in the **/etc/default/audit** file controls the default location for the log file. As distributed, the system creates the log file in the directory **/var/audit**. To place the log file in another directory or to use a character special device, either use the **defadm** command to change the value of AUDIT_DEFPATH or use the **-P** option of the **auditlog** command. If specifying a character special device (for example, tape), it is necessary to allocate the device to the appropriate user. The audit administrator role, AUD, does not allow for this capability. Refer to the *System Administration* manual for detailed information on allocating devices.

If the argument to the **`-P`** option is not an absolute pathname to either a directory or special character device that exists, the following error message is displayed:

```
    UX:auditlog: ERROR: full pathname not specified
```

For example, if you have a **`/sysadm`** file system for system administrators and you want to put the audit event log file in its **`audit`** directory, you would use the command:

```
    auditlog -P /sysadm/audit
```

If you use a directory other than the default, you should ensure that the directory is properly protected. The owner of the directory should be `root,` the group should be `audit,` and the file permissions should be read, write and execute for the owner and group. For example, the permissions would look like this:

```
    ls -ld /sysadm/audit
drwxrwx--- 1 root   audit   512 Dec 19 10:51 /sysadm/audit
```

NOTE

The **`-P`** option of **`auditlog`** can be used only when auditing is disabled. If it is used when auditing is enabled, an error message is printed.

Specifying the Name of the Audit Event Log File

The auditing subsystem generates the name of the log file. It is always a seven digit number containing a date stamp and a sequence number. The first four digits indicate the month and day the log file was created, while the last three digits are the sequence number. Thus the audit event log file **`/var/audit/0415477`** is a log file created on April 15, with a sequence number of 477.

The administrator may append up to seven characters to the system generated log file name. The additional characters are called the node name. The node name is set by the `AUDIT_NODE` parameter in the **`/etc/default/audit`** file. In the distributed system, there is no default value assigned to the `AUDIT_NODE` parameter.

You may also use the **`-p`** option of the **`auditlog`** command to specify a node name. This option takes a character string as an argument and appends that string (the node name) to the audit event log file name. The string must contain no more than seven characters; if the string is longer than seven characters, **`auditlog`** prints the following error message:

```
    UX:auditlog: ERROR: event log node must be < 8 characters
```

In addition, the node name used as the argument to the **`-p`** option must not contain a slash. It it does, **`auditlog`** prints the following error message:

```
    UX:auditlog: ERROR: event log node may not contain a
    slash
```

Appending a character string to the log file name is useful if you have several machines in a network, because it lets you identify the machines on which the logs were created. For example, assume that you have three machines called `beowulf`, `wiglaf`, and `unferth`. Because all these names are seven characters or less, you could append the machine name to the log file name on each of these machines. For example, the command

```
auditlog -p beowulf
```

would add the string `beowulf` to the log file created in the **/var/audit** directory. In that case, the log file name would look like this:

```
/var/audit/1219001beowulf
```

If you have more than one machine generating audit event log files, it is recommended that you add the machine name or an abbreviation of it to the log file name.

NOTE

The **-p** option of **auditlog** can be used only when auditing is disabled. If it is used when auditing is enabled, an error message is printed.

Specifying the High Water Mark

The high water mark controls the amount of data stored in an audit buffer before the auditing subsystem switches to the next available buffer. Whenever an audit record would make the amount of data in the buffer larger than the high water mark, the auditing subsystem flags the buffer as writable and switches to the next available buffer. It also awakens a daemon process that writes the flagged buffer to the audit event log file and returns it to the set of available buffers. The default high water mark is the size of the audit buffer itself. The audit buffer size is the system tunable parameter `ADT_BSIZE`. This is defined in **/etc/conf/mtune.d/audit**.

To specify a new high water mark, use the **-v** option of the **auditlog** command. This option takes an integer as an argument; the integer specifies the size of the high water mark in bytes. The valid range of values is from zero (0) to `ADT_BSIZE`. If the value for the high water mark is not within the correct range, **auditlog** prints the following error message:

```
UX:auditlog: ERROR: invalid high water mark specified
Audit Buffer High Water Mark Must Be >= 0 or <= ADT_BSIZE bytes
```

The **auditlog** command replaces the string *current buffer size in bytes* with the appropriate number, based on your system's value for `ADT_BSIZE`.

To set the high water mark to 1024 bytes, for example, type the following command:

```
auditlog -v 1024
```

To determine an appropriate value for the high water mark, you need to evaluate the trade-offs. A relatively low value ensures that the buffers are written to the log file more frequently, however this increases system overhead. A relatively high value can improve performance but also increases the risk that large amounts of audit data will be in a buffer if the system crashes.

If you set the high water mark to a value of zero, the auditing subsystem bypasses the audit buffer and writes audit records directly to the log file. This direct write allows you to use the **`-w`** option of **`auditrpt`** to monitor events as they occur. Bypassing the audit buffer increases the number of disk writes and process switches that take place. This extra processing will affect the performance of the machine and significantly reduce system throughput.

Specifying the Size of the Log File

By default, the audit event log file will grow until it consumes all space available on the device which contains it. In many cases, you may want to limit the size of the log file to avoid the problems caused by a full device. You can control the size of the audit event log file with the **`-x`** option of **`auditlog`**.

The **`-x`** option takes a positive integer as an argument; the integer specifies the size of the log file in blocks. (Each block is 512 bytes.) For example, the following command specifies that the maximum log file size is 100 blocks:

```
auditlog -x 100
```

The size of the log file must be greater than or equal to the size of the audit buffer, which is set by the system tunable parameter `ADT_BSIZE`. This is defined in **`/etc/conf/mtune.d/audit`**. If the size specified by **`-x`** is not greater than or equal to `ADT_BSIZE`, **`auditlog`** prints the following error message:

```
UX:auditlog: ERROR: invalid max_size specified
Audit Log File Size Must Be >= max_size (512 byte)blocks
```

The **`-x`** option is valid only if the log file is a regular file. If the log file is not a regular file and you use the **`-x`** option, **`auditlog`** prints the following warning message:

```
UX:auditlog: ERROR: max_size applies only to regular
files
```

A value of 0 (zero), indicates that the audit event log file is unbounded. The log file continues to grow until there is no space left on the device.

Specifying the Action to Take on Error

The `AUDIT_LOGERR` parameter of the **`/etc/default/audit`** specifies the action to be taken when an auditing subsystem error occurs. An example of an error would be a failure to write to a log file that has not yet reached its maximum size. The possible actions are `DISABLE`, which disables the auditing subsystem, and `SHUTDOWN`, which shuts the computer system down. The value for this parameter in the distributed system is `DISABLE`. The value of `SHUTDOWN` will result in a sudden loss of computer services for users of your system, however it will provide for the highest security. There will always be an audit record covering all the times when the system was in multiuser mode.

The value of `DISABLE,` may result in a gap in the audit trail. That is, there will be no audit records for the time between the occurrence of the audit subsystem error and the next time auditing is enabled. However, there will also be no sudden loss of service to the users of the computer system.

You can set the value of the `AUDIT_LOGERR` parameter with the **`defadm`** command. The **`auditlog`** command can not be used to override the `AUDIT_LOGERR` parameter.

Specifying the Action to Take When the Log File Is Full

A log full condition is reached if one of the following occurs:

- The log file is a regular file and it has reached the size specified by the **`-x`** option of the **`auditlog`** command.
- The log file is a regular file and the file system it resides in runs out of space.
- The log file is a special character device and the device cannot hold any more data.

The auditing subsystem takes one of the following actions when a log full condition is reached:

- disable auditing
- shut down the computer system
- switch to an alternate log file and (if desired) run a program

The action taken depends on the value of the `AUDIT_LOGFULL` parameter in the **`/etc/default/audit`** file. The value for this parameter in the distributed system is `DISABLE` (disable auditing). You can set the value of the `AUDIT_LOGFULL` parameter with the **`defadm`** command. For example, to set auditing to be disabled upon a log full condition, enter the following command:

```
defadm audit AUDIT_LOGFULL=DISABLE
```

You can override the value of the `AUDIT_LOGFULL` parameter with the **`-d`**, **`-s`**, **`-a`**, and **`-A`** options of the **`auditlog`** command. The **`-d`** option specifies that auditing will be disabled, the **`-s`** option specifies that the computer system will be shutdown and the **`-a`** and **`-A`** options specify that the auditing subsystem will switch to an alternate log file. The ability to switch to an alternate log file, when the primary log file is full, allows for continuous auditing. You should consider configuring your system to switch to an alternate log file and to execute a program when the log switch occurs. By doing so, you can create a continuous series of log files, without losing any audit data. The next subsections present information on ways to accomplish this.

If you want the highest possible level of security and you cannot configure an alternate log, you should shut your system down when the log file becomes full.

Specifying Continuous Auditing

To configure a system for continuous auditing it is necessary to set the log full condition to `SWITCH` and establish an alternate log file. The parameters in the **`/etc/default/audit`** file and/or the **`auditlog`** command may be used. In the simplest example, the `AUDIT_LOGFULL` parameter is set to `SWITCH.` In this case, the auditing subsystem will create a new audit event log file in the directory specified by the value of the `AUDIT_DEFPATH` parameter whenever the primary log file becomes full. If the `AUDIT_NODE` parameter has a value, the node name will be appended to the new audit event log file. It is important to note, that if the size of the primary log file was not limited by the **`-x`** option of the **`auditlog`** command, the auditing subsystem will not be able to open a new file in this directory. In addition, if you want to run a program every time a log switch occurs, you can specify the path name of that program with the `AUDIT_PGM` parameter.

If you have an archival storage device, such as a tape drive, that can be dedicated to receiving audit event log files, you can configure the auditing subsystem so that it automatically archives old log files and maintains continuous auditing. The next subsections have information on using the features of the auditing subsystem to provide for continuous auditing.

Specifying an Alternate Log File

As previously mentioned, both the **`auditlog`** command and the parameters in the **`/etc/default/audit`** file may be used to establish continuous auditing. The **`-A`** and **`-a`** options of the **`auditlog`** command set the log full condition to `SWITCH` and allows the administrator to specify an alternate log file and alternate node name. These options, like the **`-P`**, require the absolute pathname of either a directory or special character device that already exists. The **`-a`** option like the **`-p`**, requires a character string of less than eight characters and must not contain the character slash (`/`).

The **`-n`** option allows the administrator to specify a program to be executed when the primary log file becomes full. The argument to the **`-n`** option must be an absolute pathname to either a shell script or binary executable. If not, one of the following error messages will be displayed:

`UX:auditlog: ERROR: cannot open/access path or device` *pgm_name*

or

`UX:auditlog: ERROR:` *pgm_name* `is not an executable file`

In the following example, the **`auditlog`** command is used to specify a maximum log file size, a log full condition of `SWITCH,` the location of the alternate log file, and a program to be executed. Note that when the **`-n`** option is used, either the **`-A`** or the **`-a`** option must be used.

```
auditlog -x 100 -A /var/audit -n /etc/security/audit/auditmail
```

The primary log file is created in the directory specified by the `AUDIT_DEFPATH` parameter. In this case, the default directory **`/var/audit`** is assigned to `AUDIT_DEFPATH.` The **`-x`** option limits the size of the audit event log file to 100 blocks. Limiting the size of the log file is necessary because the alternate log file (specified by the **`-A`** option) is in the same file system as the primary log file. If the **`-x`** option were not used, the primary log file would grow until there was no more room on the file system. It would then be impossible to open the alternate log file, and audit data could be lost. The value chosen for **`-x`** depends on the amount of free space available in the file system that contains the audit event log file.

When the primary log file reaches 100 blocks, the log full condition occurs and the auditing subsystem creates a new log file in the directory specified by the **`-A`** option. This file will have a sequence number one greater than that of the old primary log file, if the log switch occurs on the same day that the primary log was created. For example, if the old primary log is **`/var/audit/0415477`** and if the day is still April 15, the new log file is **`/var/audit/0415478.`** When the new log file has been opened successfully, the program specified by the **`-n`** option is executed. In this case, **`/etc/security/audit/auditmail`** is a simple shell script created by the audit administrator that sends mail to the user `auditor`. The script might contain a command similar to the following:

```
/usr/bin/mailx -s 'Audit log switch occurred' auditor \
                  < /dev/null > /dev/null
```

Upon receiving the mail from the script, you would archive the old audit event log file, then delete it to restore space in the **`/var/audit`** directory. You might also want to archive the audit map files at the same time. A detailed discussion of the audit map files is contained in the chapter entitled “Displaying Audit Trail Information”.

Upon completion of a log switch the values set by the previous invocation of **`auditlog`** are lost. In the example above, the maximum file size would return to its default setting of zero and the log full condition, alternate log file and the program settings would revert back to the values assigned to the parameters `AUDIT_LOGFULL, AUDIT_DEFPATH and AUDIT_PGM,` respectively. If the `AUDIT_LOGFULL` parameter was set to `SWITCH` and the `AUDIT_PGM` parameter to the path name of the program to be invoked, you will need to only reissue the **`auditlog`** command to set the maximum log file size.

This scheme requires a great deal of attention from the audit administrator. For example if a log switch occurs at night or when the audit administrator is busy, the alternate log file may become full before the old primary log is archived. To avoid these problems, you can use the **`-n`** option to specify a program that will automatically archive the old log file, remove the old log file to free up storage space, and then reissue the **`auditlog`** command to reset any log file attributes that changed during the log switch. Therefore, continuous auditing and the maintenance of the audit event log file can be largely automated.

Displaying Information About Auditing

`auditlog,` when invoked with no options will display the following auditing information:

- the current status of auditing
- the full pathname of the primary audit event log file
- the audit buffer high water mark
- the maximum file size of the audit event log file
- the action to be taken upon a log full condition

- the action to be taken upon an error condition
- the full pathname of the alternate audit event log file
- the program to be run upon a log full condition of `SWITCH`

Screen 4-2 illustrates the output of **`auditlog:`**

```
 auditlog

Current Status of Auditing:                      ON
Current Event Log:                               /var/audit/0725001
Current Audit Buffer High Water Mark:            20480 bytes
Current Maximum File Size Setting:               none
Action To Be Taken Upon Full Event Log:          system shutdown
Action To Be Taken Upon Error:                   system shutdown
Next Event Log To Be Used:                       none
Program To Run When Event Log Is Full:           none
```

Screen 4-2. auditlog Output

The first line of the output reports the status of auditing; it will be either `ON` or `OFF`. If auditing is on, the rest of the display reports the current audit settings. If auditing is off, the display reports the settings that will take effect when auditing is enabled next.

Setting Audit Criteria

The first step in setting audit criteria, is for the administrator to decide which events to audit. The chapter entitled “Auditable Events”, provides a description of the auditable events along with guidelines for determining which events to audit. In addition, Appendix A provides a summary of all events and a listing of all event classes.

The **`auditset`** command provides the ability to set and display the audit criteria.

- Display:
  - the system audit criteria (**`-d`** or **`-Z`** option)
  - the user audit criteria for all active users (**`-d`** and **`-a`** options)
  - the user audit criteria for one or more specific active users (**`-d`** and **`-u`** options)
- Set:
  - the system audit criteria (**`-s`** or **`-x`** option)
  - the user audit criteria for all active users (**`-s`** or **`-x`**) and **`-a`** and (**`-e`** or **`-z`**)
  - the user audit criteria for one or more specific active users (**`-s`** or **`-x`**) and **`-a`** and (**`-e`** or **`-z`**)

As explained in the chapter “Overview of the Auditing Subsystem”, subsection “Audit Event Masks”, the auditing subsystem employs an event mask mechanism to determine which events are currently being audited. There is a system event mask, a user event mask for each active user, a process event mask for each process, and an object level event mask. The following subsections will provide information on setting the events for the system and user masks.

Setting System Wide Audit Criteria

The system event mask applies to all non-exempt processes on the system and at a minimum always contains the fixed events. Selectable events may be added and deleted at the discretion of the administrator, but the fixed events can not be altered. Audit criteria may be set before or after auditing is enabled.

The **`-s`** and **`-x`** options of **`auditset`** are used to set the system wide audit criteria (the **`-x`** option is for extended trusted application audit criteria). The argument to the **`-s`** or **`-x`** option, referred to as an event list, may consist of one or more events or event classes. If the event list contains more than one event, each item in the list must be separated by a comma. Additionally, `all` and `none` may be used as event keywords. For the system event mask, `all` is defined to be the set of all fixed and selectable events and `none` is defined to be the set of all fixed events. Keyword(s) will be ignored if intermixed with events and event classes. There are three valid operators that may precede an event list. Only one operator may be specified per event list. Table 4-1 describes the meaning of each operator.

Table 4-1. Operators Used in Setting Auditable Events

Operator	Meaning
`[no operator]`	Replace the current set of auditable events with the ones listed.
`+`	Add the event(s) to the current set of auditable events.
`-`	Delete the event(s) from the current set of auditable events.
`!`	Audit all events except the event(s) listed.

For example, if you want to add all events related to file creation to the current system wide audit criteria use the following command:

```
auditset -s +file_make
```

On the other hand, if you want to delete all events related to file creation from the current system wide audit criteria type the following command:

```
auditset -s -file_make
```

If you want the system wide audit criteria to contain only the kill and ulimit events, enter the following command:

```
auditset -s kill,ulimit
```

Setting User Audit Criteria

User audit criteria can be set with either the **auditset, useradd,** or **usermod** commands. Each method has a different effect on the user's audit criteria. The **auditset** command sets the user audit criteria dynamically; the changes take effect immediately during the user's current login session. However, the settings are in effect only until the user no longer owns any active processes on the system; that is, the user is not logged in to the system, and has no **cron** jobs running. If the user logs out from all of their current sessions, and has no **cron** jobs running, and then logs in again, the audit criteria are no longer in effect. The mask is instead taken from the **/etc/security/ia/audit** and **/etc/security/ia/master** files, which are managed by the **useradd** and **usermod** commands. To set user audit criteria, via the **auditset** command, the specified user(s) must be currently logged in.

The **useradd** or **usermod** commands set the user audit criteria statically; the changes take effect the next time the user logs in. However, if the user already owned processes on the system, either because they were currently logged in from somewhere else, or because **cron** was running a job for them, then the audit criteria for the already active processes would apply to the new processes as well. The new user audit criteria is in effect for every subsequent login session until the **usermod** command is invoked again.

In general, you will use **useradd** or **usermod** if there is a set of events that you always want to audit for a given user. You can then use **auditset** to add events to the user audit criteria to meet special needs. For example, if you suspect that a malicious user has guessed the password for the user `wts` and has logged in as that user, you could use the **auditset** command to start auditing all events for the user `wts` immediately. You could then monitor that user's activities and determine if there had indeed been a breach of security.

Setting User Audit Criteria with auditset

The **-e**, **-z, -a**, and **-u** options of the **auditset** command are used to set user audit criteria (the **-z** option is for extended trusted application audit criteria). The **-e** and **-z** options like the **-s** option, require an event list as their option arguments. The keywords `0all` and `none` and the operators described in Table 4-1 are also valid with the **-e** and **-z** options. The **-u** or the **-a** option must be used with the **-e** or **-z** option. The **-a** option sets audit criteria for all active users on the system, while the **-u** option sets audit criteria for a specified active user(s).

The argument to **-u** is either a single user name or a list of user names, each separated by a comma but not a space. (You can use either numeric user IDs or user names.)

For example, to audit all file creations by the user `aeb,` use the following command:

```
auditset -u aeb -e file_make
```

The preceding command replaces the audit criteria for this user with the events in the `file_make` event class.

To add the set_uid event to the audit criteria for users `aeb` and `xyz` use the following command:

```
auditset -u aeb,xyz -e +set_uid
```

Setting User Audit Criteria With useradd or usermod

As mentioned, **`auditset -e`**, **`-z`**, **`-u,`** and **`-a`** set user audit criteria dynamically and thus effect only current existing processes owned by that user, including all current login sessions for that user, and any **`cron`** jobs for that user that may be executing at that time. Any new processes created while such processes exist will also have the same criteria. To set audit criteria for all future login sessions for a user, you need to use the **`-a`** option of the **`useradd`** or **`usermod`** commands.

The **`useradd`** command is used to add a new user to the system. It adds a new user entry to the **`/etc/passwd`** and **`/etc/shadow`** files and can also create a default user event mask for a new user if you specify the **`-a`** option. The argument to this option is either a single event or a list of events, each separated by a comma but not a space. Event classes may also be used as input to the **`-a`** option. You cannot use operators in front of the event(s) with the **`-a`** option of **`useradd.`** For example, to add `dhh` as a new user with all directory creation events audited, use the following command:

```
useradd... -a dir_make dhh
```

The ellipsis (...) indicates that you would normally specify other options which are not related to auditing. For more information, see the **`useradd(1M)`** manual page in the online man page.

The **`usermod`** command can be used to change users' login information after they have been added to the system. To modify the default user event mask or to add one for an existing user, use the **`-a`** option of **`usermod`**. The argument to this option is either a single event or a list of events, each separated by a comma but not a space. Event classes may also be used as input to the **`-a`** option.

For example, earlier in this section the user `dhh` was added to the system, with a user event mask that audited all directory creations (specified by the `dir_make` event class). If you want to change the user event mask to audit all file creations instead of all directory creations, enter the following command:

```
usermod -a file_make dhh
```

You can use operators described in Table 4-1 with the **`-a`** option of **`usermod`**. For example, assume that after you used **`useradd`** to add the user `dhh` with a user event mask that audited all directory creations (specified by the `dir_make` event class) you wanted to change the user event mask to include all file creations as well as directory creations. To change the user event mask for the user `dhh` use the following command:

```
usermod -a +file_make dhh
```

Because the `+` operator is used, this command adds the events defined by the `file_make` event class to the existing audit criteria for the user `dhh`.

NOTE

You must use the login name, not a numeric user ID, to set audit criteria for users with the **`useradd`** or **`usermod`** commands. Also, you can use only one user name as an argument to either of these commands. If you want to set the same audit criteria for two or more users, you must enter a separate command for each.

The following sequence will allow you to set a default audit mask for all users:

`defadm useradd AUDIT_MASK=`*<event list>*
`rm /etc/security/ia/audit`
`auditcnv`
`creatiadb`

Displaying Audit Criteria

The **`auditset`** command invoked with no options displays the system, user, and object level audit criteria. Object level audit criteria are displayed only if the Enhanced Security Utilities are installed. Screen 4-3 illustrates the output of **`auditset.`**

To display selected audit criteria, use the **`-d`** option of **`auditset.`** When the **`-d`** option is used alone, **`auditset`** reports only the system wide audit criteria as shown in Screen 4-4.

```
auditset
System Audit Criteria:
    add_grp add_usr add_usr_grp assign_lid assign_nm audit_buf audit_ctl
audit_dmp audit_evt audit_log audit_map date deactivate_lid del_nm init
mod_grp
mod_usr


User Audit Criteria:
    aeb (201): create link mk_node sym_create unlink
    dhh (225): link mk_dir rm_dir sym_create unlink
    root (0): none
    wts (251): all
```

Screen 4-3. auditset Default Output

```
auditset -d
System Audit Criteria:
    add_grp add_usr add_usr_grp assign_lid assign_nm audit_buf audit_ctl
audit_dmp audit_evt audit_log audit_map date deactivate_lid del_nm init
mod_grp mod_usr
```

Screen 4-4. auditset -d Output

Starting the Audit Subsystem

After specifying the events to be recorded and the characteristics of the log file, you are ready to start auditing. To enable auditing, use the **`auditon`** command, as follows:

```
auditon
```

Once auditing has been successfully started the following message is displayed:

```
UX:auditon: INFO: Auditing enabled audit_file
```

If you invoke **`auditon`** when auditing is already enabled, the following message is displayed:

```
UX:auditon: WARNING: Auditing already enabled
```

While auditing is enabled, the auditing subsystem writes an audit record to the audit event log file each time **`auditon`** is invoked.

Stopping the Audit Subsystem

To stop the auditing subsystem, use the **`auditoff`** command, as follows:

```
auditoff
```

Once auditing has been successfully stopped the following message is displayed:

```
UX:auditon: INFO: Auditing disabled
```

If you invoke **`auditoff`** when auditing is already disabled, the following message is displayed:

```
UX:auditoff: WARNING: Auditing already disabled
```

The auditing subsystem writes an audit record to the audit event log file for the invocation of the **`auditoff`** command and then disables auditing. Any processes that are being audited but have not completed will not generate audit records.

NOTE

Auditing is an important component of a secure computer system. If you run your system with auditing disabled, it cannot be considered secure. You will not have a record of security related events for the time when the system was up but auditing was disabled.

Starting the Audit Subsystem with /etc/init.d/audit

The **`/etc/init.d`** directory contains a series of files, each of which consists of commands that **`init`** executes when the operating system changes init states. The **`audit`** file, as distributed, invokes the **`auditon`** command to enable auditing when the system enters multiuser mode and the **`auditoff`** command to disable auditing when the system enters either single user mode or power off state. If the system is going to multiuser mode and the **`auditon`** command fails the system returns to single user mode. The following is the **`/etc/init.d/audit`** file as distributed:

The administrator may edit this file and add additional auditing commands. This will allow site specific audit requirements to be configured each time auditing is enabled. For example, the addition of the auditset command to set the audit criteria and the auditlog command to set log file characteristics.

```
#
if [ ! -d /usr/bin ]
then        # /usr not mounted
    exit
fi
# Check to see if auditing is enabled
/usr/sbin/auditlog >/dev/null 2>&1
if [ $? -eg 3 ]
then        # Auditing not enabled
    exit
fi

case "$1" in
´start´)
    # Enable Auditing when going to multi-user mode
    /usr/sbin/auditon

    # The system returns to single-user mode if auditing can not be enabled
    # The following return codes are defined as success:
    # 0-auditing successfully enabled or already enabled
    # 3 -audit package is installed but not included in the current "unix"
    if [ $? -ne 0 -a $? -ne 3 ]
    then
        echo "Cannot enable auditing, changing to single user mode"
        /sbin/init S
    fi
    ;;
´stop´)
    # Disable Auditing when going down to single user mode,
    # FIRMWARE or power off.
    /usr/sbin/auditoff
    ;;
*)
    echo "Usage: /etc/init.d/audit { start | stop }"
    ;;
esac
```

CAUTION

Do not add entries to **`/etc/init.d/audit`** unless you are sure that you have invoked the audit commands correctly. Otherwise, auditing may not function properly when the system enters multiuser mode. If the **`auditon`** command fails, the system will be brought back to single user mode.

A Quick Reference to Enabling Audit

- Setting the path name of the primary log file:

  defadm audit AUDIT_DEFPATH=/absolute/pathname
  auditlog -P /absolute/pathname

- Setting the node name of the primary log file:

  defadm audit AUDIT_NODE=*string*
  auditlog -p *string*

- Setting the action to be taken when the log file is full:

  defadm audit AUDIT_LOGFULL=[SHUTDOWN|DISABLE|SWITCH]
  auditlog [-s | -d | [-A /absolute/pathname [-a *string***]]]**

- Specifying an alternate log file:

  defadm audit AUDIT_LOGFULL=SWITCH
  defadm audit AUDIT_DEFPATH=/absolute/pathname
  auditlog -A /absolute/pathname

- Setting the node name of the alternate log file:

  defadm audit AUDIT_LOGFULL=SWITCH
  defadm audit AUDIT_NODE=*string*
  auditlog -a *string*

- Specifying a program to run when there is a log switch:

  defadm audit AUDIT_LOGFULL=SWITCH
  defadm audit AUDIT_PGM=/absolute/pathname
  auditlog -n /absolute/pathname_to_program

- Specifying the maximum size of the log file:

  auditlog -x *size*

- Specifying the value of the high water mark:

  auditlog -v *value*

- Setting the action to be taken if there is an auditing subsystem error:

  defadm audit AUDIT_LOGERR=[SHUTDOWN|DISABLE]

- Displaying log file characteristics:

  auditlog

- Setting system wide audit criteria:

  auditset -s [operator]*event***,...**

- Setting the extended system wide audit criteria:

  auditset -x [operator]*event***,...**

- Setting audit criteria dynamically for all active users:

auditset -a -e [operator]*event***,...**

- Setting the extended auditing criteria for all active users

  auditset -a -z [operator]*event***,...**

- Setting audit criteria dynamically for specific active users:

  auditset -u *user***,... -e [operator]***event***,...**

- Setting extended auditing criteria for specific active users:

  auditset -u *user***,... -z [operator]***event***,...**

- Setting audit criteria when adding a user to the system:

  useradd... -a *event***,...** *user*

- Setting audit criteria permanently for specific users:

  usermod... -a [operator]*event***,...** *user*
  auditset

- Displaying system wide and user audit criteria:

  auditset

- Displaying system wide audit criteria:

  auditset -d

- Displaying audit criteria for all users:

  auditset -d -a

- Displaying audit criteria for specific users:

  auditset -d -u *user***,...**

- Displaying object level audit criteria:

  auditset -d -m

- Displaying the extended system audit criteria:

  auditset -Z

- Displaying the extended audit criteria for all users:

  auditset -Z -a

- Displaying the extended audit criteria for specific users:

  auditset -Z -u *user***,...**

- Displaying object level extended audit criteria:

  auditset -Z -m

- Starting auditing:

  auditon

- Stopping auditing:

`auditoff`

- File for starting/stopping auditing when the system changes states:

  `/etc/init.d/audit`

5
Displaying Audit Trail Information

5 Displaying Audit Trail Information

Introduction

This chapter provides information on the **auditrpt, auditmap** and **auditfltr** commands. The **auditrpt** command displays the information contained in an audit event log file. The administrator may chose to display the entire contents of a log file or selected portions of it. In addition, audit information may be retrieved from either the current log file or one or more previous log files. The **auditmap** command generates the audit map files. The audit map files contains system dependent information and is used by the **auditrpt** command to interpret and translate numeric data contained in the log file. The **auditfltr** command provides for the portability of audit event log files among different machine architectures.

Format of auditrpt Output

The output of the **auditrpt** command consists of three sections. The first section is the command line entered by the administrator. The remaining two sections are repeated for each audit event log file that is being processed. The second section contains log file and system identification information. This information includes the internal identification of the log file, the audit version that generated the log file, and the identification of the machine that generated the log file. The third section contains the audit record(s) that match the selection criteria specified on the command line. One audit record is displayed per line and consists of a series of fields, separated by commas. The format of an audit record is as follows:

*time,event,pid(lwpid),outcome,user,group(s),session,"subj_lvl", \
(obj_id:obj_type:"obj_lvl":device:maj:min:inode:fsid)(...)[,pgm_prm]*

where

time	The time of the event. The format is *hour:minute:second:day:month:year.*
event	The event type. See Appendix A for a complete list of events.
pid (lwp_id)	The process id preceded by the letter *P*. The `lwp_id` displayed in parentheses. For events that occur in interrupt or system process context, -1 is printed to indicate that a process or LWD ID is not available.
outcome	The outcome of the event: *s* for success or *f(exit code)* for failure.

user	The real and effective user names separated by a colon (for example, `real_user_name:effective_user_name)`.
group	Real and effective groups are displayed, followed by a list of supplementary groups, if any. If group names cannot be found, the group id will be displayed. Groups are separated by a colon (that is, *real_grp:effective_grp:suppl_grp:suppl_grp*:...).
session_id	The session ID number, preceded by the letter **S.**

(obj_id:obj_type:"obj_lvl":device:maj:min:inode:fsid)

This field contains object identification information, enclosed in parentheses.

obj_id	The name of a regular file, special file, directory, named pipe or the id of an IPC object. If the full pathname of a file system object cannot be determined, the partial pathname will be printed with an asterisk (*) as a prefix
obj_type	The object type which may be either: *f* (regular file), *c* (character special file), *b* (block special file), *l* (link), *d* (directory), *p* (named pipe or unnamed pipe), *s* (semaphores), *h* (shared memory), *m* (messages).
"obj_lvl"	The level or level range of the object that was accessed during the event. The level name may contain colons and/or commas; and the level range will be displayed as `"range high"-"range low"`. For readability, each level name will be enclosed in double quotes (for example,`"system:private,audit"`.
device	The object's device number
maj	The major number component of the object's device.
min	The minor number component of the object's device.
inode	The object's inode number.
fsid	The object's file system ID number.

pgm_prm	This field is specific to each event and may be composed of several subfields. The *pgm_prm* field for each event is described fully in the **`auditrpt(1M)`** manual page in the online man pages.

Commas in the display of an audit record serve either to separate fields or act as place holders if the field is not appropriate for the specific event. For example, the *date* event has no objects related to it, therefore the *(obj_id:obj_type:"obj_lvl":device:maj:min:inode:fsid)* field will be replaced with a comma.

If a field is appropriate for an event but its value is "invalid," a *?* will be displayed. For example, if a **`login`** event fails because the logname used is unknown to the system (cannot be translated into a UID), the *user* will be flagged as "invalid" and a *?* will be displayed.

The following is an example of an audit record:

```
14:32:00:18:05:91,open_rd,P4556(2),f(13),boris:boris,irs:staff:proj43,
S328,“Restricted:Proj43”,
(/etc/shadow:f:“system:private”:0x440000:17:2:148:0x440000),12
```

`14:32:00:18:05:91`	The time when the event occurred: 2:32PM on May 18, 1991.
`open_rd`	The event. See Appendix A for a complete list of events.
`P4556(2)`	The process ID number of the process that triggered the event, preceded by the letter *P*. The ID of the LWP that triggered the event is in parentheses.
`f(13)`	The event failed with an exit code of 13.
`boris:boris`	The real user and the effective user separated by a colon.
`irs:staff:proj43`	The real group and the effective group followed by a supplementary group. Each subfield is separated by a colon.
`S328`	The session ID number preceded by the letter *S*.
`“Restricted: Proj 43”`	The security level of the process enclosed in double quotes.
`(/etc/shadow:f:`	The object identification information which includes the following subfields:

`/etc/shadow`	The name of the object.
`f`	The object type which is a regular file.
`“system: private”`	The security level of the object enclosed in double quotes.
`0x440000`	The device number.
`17`	The major number of the object’s device.
`2`	The minor number of the object’s device.
`148`	The object’s inode number.
`0x440000`	The object’s file system ID.
`12`	The file descriptor.

Displaying Information from the Audit Log

The **`auditrpt`** command allows the administrator to display either the entire contents of a log file or selected portions of it. In addition, audit information can be retrieved from either the current log file or one or more previous log files. If no options or arguments are specified, the entire current audit event log file will be displayed in the order in which events were recorded. Auditing must be enabled to view a current log file.

Selected portions of an audit event log file may be displayed based on one or more of the following criteria:

- event type (**`-e`** option, **`-z`** option, or **`-n`** option)
- user id (**`-u`** option)
- object id (**`-f`** option)
- object type (**`-t`** option)
- time interval (**`-s`** and/or **`-h`** options)
- event status: failure or success (**`-a`** option)
- privileges used (**`-p`** option)
- miscellaneous event subtype (**`-v`** option)
- events that satisfy at least one of the specific auditing criteria (**`-o`** option)

When two or more criteria selection options are specified on the same command line, **`auditrpt`** reports only those events that meet all the criteria listed by the options. In other words, it displays the intersection of the criteria. In the example shown in Screen 5-1, **`auditrpt`** displays only those records that have both `boris` in the user field (as either the real or effective user) and **`/etc/passwd`** in the object_id field.

When you use the **`-o`** option, **`auditrpt`** the events that satisfy at least one of the specified auditing criteria are displayed. When this option is not specified a record must satisfy all criteria in order to be displayed.

```
auditrpt -u boris -f /etc/passwd
Command Line Entered: auditrpt -u boris -f /etc/passwd


DATE: 0518, LOG NUMBER: 001, AUDIT VERSION: 5.0

MACHINE ID: UNIX_SV sfadf 4ES/MP 1 i386
14:32:00:18:05:91,open_rd,P5456,f(13),boris:boris,irs:staff:proj43,S337,
,(/etc/passwd:f::0x440000:148:0x440000),12


14:32:00:18:05:91,open_rd,P4556,f(13),boris:boris,irs:staff:proj43,S328,
,(/etc/passwd:f::0x440000:17:2:148:0x440000),12
```

Screen 5-1. Use of auditrpt to Find Specific User Access to a File

If there are no audit records to match a selection criteria the following warning message is displayed:

```
UX:auditrpt: WARNING: no match found in event log file(s)
```

If at least one audit record matches a selection criteria, the command will be silent about the portion of the selection criteria that did not result in a match.

To display audit information from a log file other than the current one, specify the log file(s) as a command line argument. It is not necessary for auditing to be enabled to process previous log files. If both valid and invalid log files are specified, the valid log file(s) will be processed and the following warning message will be displayed for the invalid logs:

```
UX:auditrpt: WARNING: event log file log does not exist
```

Additionally, the **`-i`** option may be used to specify that the log file is to be taken from standard in.

To further assist the administrator, the **`auditrpt`** command has the ability to:

- display the audit records in reverse chronological order (**`-b`** option)
- display audit records as they are being written to the audit event log file (**`-w`** option)
- specify a directory containing the audit map files (**`-m`** option)
- display of the `lwp_id` (**`-x`** option)

The following subsections will describe in detail each of the **`auditrpt`** options.

Displaying Information by Event

The **`-e`** option of the **`auditrpt`** command is used to display audit information for specific events. The argument to the **`-e`** option may consist of one or more events or event classes. Each event or event class must be separated by a comma. If the wrong event type is specified, the following message is displayed and the command terminates:

```
UX:auditrpt: ERROR: event type or class "event" does not exist
```

A space will be interpreted as the end of the event list. For example, to display any events where the system date was changed, enter the following command:

```
auditrpt -e date
```

The operator `!` may be used to signify all the events except those listed. For example, to exclude information about the `access, open_rd,` and `status` events and to display information about all other events, enter the following command:

```
auditrpt -e !access,open_rd,status
```

If an invalid event is given as input to the **`-e`** or -**`n`** option, **`auditrpt`** will display the following message and terminates processing

```
UX:auditrpt: ERROR:event type or class event does not
exist
```

Events are validated against the information contained in the audit map files. Refer to the **`auditmap(1M)`** manual page (see the online man pages) for further information on the audit map files.

Displaying Information About Users

The **`-u`** option of the **`auditrpt`** command is used to display audit information for specific users. The argument to the **`-u`** option may consist of one or more users. A user may be specified by either a logname or numeric uid. Each logname or uid must be separated by a comma. A space will be interpreted as the end of the user list. For example, to display all information related to the user `gailg` enter the following command:

```
auditrpt -u gailg
```

The following command displays audit information for the user whose uid is `210` and the user `wts:`

```
auditrpt -u 210,wts
```

User names and IDs are validated against the information contained in the audit map files. Refer to the **`auditmap`** manual page (see the online man pages) for further information. If an invalid user uid or logname is given as input to the **`-u`** option, **`auditrpt`** will display the following warning message and continue processing:

```
UX:auditrpt: WARNING: user id user does not exist in audit
map
```

If the user exists but there are no audit records for the user in the audit file, **`auditrpt`** will display the following warning message and continue processing:

```
UX:auditrpt: WARNING: no match found in event log file(s)
```

Displaying Information by Object Identity

The **`-f`** option of the **`auditrpt`** command is used to display audit information for specific objects. The argument to **`-f`** may consist of one or more of the following objects: regular file, special character file, special block file, named pipe, IPC id, loadable module id, or directory. Each object must be separated by a comma. A space will be interpreted as the end of the object list. An object (except for an IPC id or loadable module id) must be specified by its full pathname. For example, to display audit information related to the object **`/etc/passwd`** enter the following command:

```
auditrpt -f /etc/passwd
```

For example, assume that your system has two files, **`/proj/mgmt/schedule`** and **`/proj/mgmt/staff,`** that contain sensitive information about a new product that your company is developing. The following command displays all the audit information related to each file:

```
auditrpt -f /proj/mgmt/schedule,/proj/mgmt/staff
```

If the object specified is not a full pathname, the following error message is displayed and processing is terminated:

```
UX:auditrpt: ERROR: full pathname must be specified for
```
obj_id

Displaying Information by Object Type

The **`-t`** option of the **`auditrpt`** command is used to display audit information for specific object types. Object types are identified by single letter codes, as shown in Table 5-1.:

Table 5-1. Object Type Codes

Character	Object Type
`f`	regular files
`c`	character special files
`b`	block special files
`l`	links
`d`	directories
`p`	named pipes or unnamed pipes
`s`	semaphores
`h`	shared memory
`m`	messages
`e`	transport endpoints.

The argument to the **`-t`** option may consist of one or more object types. Each object type must be separated by a comma. A space will be interpreted as the end of the list of object types. For example, to display all audit records for semaphores, enter the following command:

```
auditrpt -t s
```

For example, the following command would display all audit records for both semaphores and pipes:

```
auditrpt -t s,p
```

If the object type is invalid, **`auditrpt`** displays the following error message and terminates processing:

`UX:auditrpt: ERROR: invalid object type specified:` *object_type*

Displaying Information About Privileges

The **-p** option of the **auditrpt** command displays information about events that involved privileged operations. The argument to the **-p** option may consist of one or more privilege names or the keyword `all`. Each privilege name must be separated by a comma. A space will be interpreted as the end of the privilege list. If you specify the keyword `all`, **auditrpt** will display all audit records for all privileges. If you specify a privilege name or names after the **-p** option, **auditrpt** will display only the audit records that involve the specified privilege(s).

For example, most audit user level commands and system calls require the `P_AUDIT` privilege. An exception is the **auditdmp** system call, which requires the `P_AUDITWR` privilege to write miscellaneous audit records to the audit event log file. If you want to see all events that involve the `P_AUDIT` privilege, enter the following command:

```
auditrpt -p audit
```

The `P_DACREAD` and `P_MACREAD` privileges are needed to override Discretionary Access Control (DAC) and Mandatory Access Control (MAC) protections for objects. If any user who is not a system administrator acquires these privileges, there has been a serious breach of system security. If you want to see all uses of these privileges, use the following command:

```
auditrpt -p dacread,macread
```

For a complete list of privileges, see the **intro(2)** manual page in the online man pages.

Displaying Information About a Time Interval

Two different options control how information is displayed about time intervals:

-s	displays all events that started on or after the time listed with the option.
-h	displays all events whose start times are earlier than or equal to the time listed with the option.

You specify the time in the same format used by the **date(1M)** command. The time format is *[mmdd]HHMM* or *mmddHHMM[[cc] yy]*, where *mm* is the month number, *dd* is the day number in the month, *HH* is the hour number (24 hour system), *MM* is the minute number, *cc* is the century minus one, and *yy* is the last two digits of the year number.

For example, the following command reports on all events that started on or after 8:35AM on February 17 of the current year.

```
auditrpt -s 02170835
```

By specifying a starting time with **-s** and an ending time with **-h**, you can report on events that occurred in that span of time. For example, assume that want to know about all password changes that occurred between 1AM and 8AM on February 4, you would enter the following command:

```
auditrpt -e passwd -s 02040100 -h 02040800
```

If you do not use the **`-o`** option, the end time specified by the **`-h`** option must be later than the start time specified by **`-s`**. If not, **`auditrpt`** displays the following error message and terminates processing:

```
UX:auditrpt: ERROR: start time must be earlier than the
end time
```

If you use the **`-o`** option, you can specify a start time later than the end time. This allows you to report on all events in the log file except those occurring during a specified time period. For example, assume that you want to know about all password changes except those that occurred between noon and 5PM on February 3. To report this information, you would type the following command:

```
auditrpt -o -e passwd -s 02031700 -h 02031200
```

Displaying Information by Event Outcome

An event can have one of two outcomes: success `(s)` or failure `(f)`. The **`-a`** option of the **`auditrpt`** command is used to display audit information for a specific outcome. For example, the following command displays information about all successful events:

```
auditrpt -a s
```

If the argument to **`-a`** is neither a `s` or `f`, **`auditrpt`** displays the following error message and terminates processing:

```
UX:auditrpt: ERROR: invalid outcome specified
```

Including LWP Information in an Audit Report

Normally, only the process ID is displayed for each record. However, the **`-x`** option allows you to include the ID of the light weight process (LWP) that is associated with each event. For example, the following command will display information about all successful events, and include the LWP ID in each record:

```
auditrpt -a s -x
```

Additional Options

By default, audit records are displayed in the order in which events were recorded. The **`-b`** option of the **`auditrpt`** command allows the log file to be displayed “backwards.” In other words, the most recent records are displayed first followed by the older records. This option is useful if you think that the event(s) of interest occurred recently.

NOTE

This option requires additional processing and will therefore affect the response time of the command.

The **`-w`** option of the **`auditrpt`** command allows you to display the contents of the log file as it is being written. Its functionality is similar to the **`-f`** option of the **`tail(1M)`** command. This will allow the administrator to monitor system activity as it occurs. The **`-w`** option requires that auditing be enabled and that the audit buffer high water mark be set to zero. A high water mark of zero, will cause the auditing subsystem to bypass the audit buffers and write directly to the log file. Enter the following command to set the high water mark to zero:

```
auditlog -v 0
```

If the high water mark is not set to zero **`auditrpt`** will display the following warning message and continue processing:

```
UX:auditrpt: WARNING: data in audit buffer will not be
immediately displayed
```

If a log file is specified with the **`-w`** option the following warning message will be displayed and **`auditrpt`** will process the current log file.

`log file` *filename* **`ignored`**

Note that the **`-w`** option can not be used if the current log file is a special character device (for example, tape drive). In addition, the **`-b`** and the **`-w`** options cannot be specified on the same command line.

Processing Miscellaneous Records

The **`auditdmp`** system call can be used by privileged user level processes to write audit records to the event log file. When it creates a miscellaneous record, the **`auditdmp`** system call adds a string supplied by the user to the final field of the audit record structure. All audit records created by application programs are of type `misc`, for miscellaneous records. If you want to see all the miscellaneous records in the log file, enter the following command:

```
auditrpt -e misc
```

For more information on programming applications to generate miscellaneous records, see the **`auditdmp(2)`** manual page.

Displaying Information from Multiple Logs

As previously mentioned, **`auditrpt`** will retrieve audit information from the current log file if auditing is enabled and no log files are specified on the command line. To retrieve audit information from one or more previous log files specify the log file names as command line arguments.

For example, to display all audit information for the user `boris` in the log files, **`/var/audit/0215001`** and **`/var/audit/0216001,`** enter the following command:

```
auditrpt -u boris /var/audit/0214001 /var/audit/0215001
```

It is not necessary for auditing to be enabled to process previous log files.

The auditing subsystem keeps sequence information in each log file. If you specify a series of log files, **`auditrpt`** will check this sequence information to ensure that all log files are in the correct order and that no log files in a sequence are missing. If there are any problems, **`auditrpt`** displays the following warning message and continues processing:

```
UX:auditrpt: WARNING: event log file(s) are not in
sequence or missing
```

To minimize the size of the audit event log file, the auditing subsystem records process context information for new processes whenever the information changes, or when an audit log full SWITCH condition occurs. For example, a process can be audited for more than one event, so it would be redundant to repeat all the credential information in all the audit records related to this process. The **`auditrpt`** command reconstructs the credential information for each audit record that is displayed. If log files are not in sequence or are missing, **`auditrpt`** may not find all the necessary information and the following warning message is displayed:

```
UX:auditrpt: WARNING: credential information for Ppid is
incomplete
```

The Audit Map Files

The **`auditmap`** command generates the audit map files. The audit map files contain system dependent information and are used by the **`auditrpt`** command to translate numeric data contained in the log file. Numeric data is recorded in the log file to minimize its size and to reduce processing overhead at recording time.

The **`auditrpt`** command will use the audit map files to translate users, groups, security levels, privileges, events and system calls from numbers to names. If the audit map files are not available or the information contained within does not allow for a translation, **`auditrpt`** will display the ASCII representation of the numeric data. For example, if the audit map files do not contain information for user ID 9424, **`auditrpt`** displays the number `9424` instead of the user name in its output. Without the audit map files the output of **`auditrpt`** is hard to read and interpret.

By default, the audit map files reside in the directory **`/var/audit/auditmap.`** The audit map files are as follows:

`auditmap`	The **`auditmap`** file is an ASCII file. It contains file identification information, which includes the audit software version, timezone information, privilege mechanism information, the system name, machine node name, operating system release and version, and the

machine type. It also contains information on all login names and their corresponding user IDs, all group names and their group IDs, all events and their corresponding event numbers, all event classes and their corresponding events, all privilege names and their corresponding numbers, and all system call names and their corresponding numbers.

The **`auditmap`** command is automatically invoked whenever auditing is enabled. If the audit map file(s) already exist they will be renamed. The auditmap file and each of the LTDB (Level Translation Database) files will be prefixed with an "o". The new audit map files will then be created.

The **`-m`** option of the **`auditmap`** command allows the administrator to specify a directory where the audit map files will reside. For example, if you want to create the audit map files in the directory **`/etc/audit/auditmap,`** enter the following command:

```
auditmap -m /etc/audit/auditmap
```

Specifying the auditmap Directory

By default, the **`auditrpt`** command will access the audit map files in the **`/var/audit/auditmap`** directory. The **`-m`** option of the **`auditrpt`** command allows the administrator to specify the directory which contains the audit map files. For example, if the audit map file is in the directory **`/etc/audit/auditmap,`** enter the following command:

```
auditrpt -m /etc/audit/auditmap
```

It is recommended that the audit map files be archived along with the audit event log files. This will allow for the accurate translation of the numeric data contained in the archived log files. In the scenario of processing archived log files, the **`-m`** option can be used to specify the directory that contains the archived map file.

The auditfltr Command

The **`auditfltr`** command provides for the portability of audit event log files among different machine architectures. **`Auditfltr`** relies on XDR (External Data Representation) for the description and encoding of the data contained in the audit event log file. An audit event log file may be translated from either native machine format to XDR format or from XDR format to native machine format.

It will be necessary to use the **`auditfltr`** command if you are planning to process audit event log files on a machine with a different architecture then the one in which they were created on. If audit event log files are moved among machines of the same architecture no additional processing is required. The **`auditfltr`** command is intended to be executed while the system is in maintenance mode.

The following options are available:

`-i` *type*	Specifies the type of the input file. The input file is always standard in.
`-o` *type*	Specifies the type of the output file. The output file is always standard out. The output file should be redirected, for example to a file or pipe, due to its data format.

The values for *type* may be *N*, for native machine format, or *X*, for XDR format. If an invalid conversion *type* is requested the following error message is displayed and processing is terminated:

```
UX:auditrpt: ERROR: conversion type type is not supported
```

If an invalid combination of conversion *types* is requested the following error message is displayed and processing is terminated:

```
UX:auditrpt: ERROR: invalid combination of conversion
types
```

If no options are specified it is assumed the input file is in native machine format and the output file is in XDR format.

The procedure for transferring an audit event log file has basically three steps. First, the audit log is converted from native machine format to the portable XDR format, using a command like the following:

```
cat /var/audit/0125054 | auditfltr -iN -oX > \
/var/audit/0125054.xfer
```

Second, the file is transferred to another machine. This can be done by transferring the file to magnetic media on one with **`cpio(1M)`** and then restoring it with the same command on the other. Third, the file is converted back to machine format with a command like the following:

```
cat /var/audit/0125054.xfer | auditfltr -iX -oN >
\/var/audit/0125054
```

The **`auditfltr`** command accepts only audit log files as input. Except for the **`lid.internal`** file, all audit map files are in ASCII format and do not require conversion. The **`lid.internal`** binary file is not portable. If you have several machines and want to transport audit files between machines for analysis, you should ensure that all machines have identical security levels and level identifiers (LIDs).

A Quick Reference to Reporting Audit Data

- Reporting audit data by user:

  `auditrpt -u` *user*,**`...`**

- Reporting audit data by object:

  `auditrpt -f` *object_id*,**`...`**

- Reporting audit data by object type:

auditrpt -t *object_type*,**...**

- Reporting audit data by event or event class:

  auditrpt -e *event*,**...**

- Reporting audit data by miscellaneous event subtype:

  auditrpt -v *subtype*,**...**

- Reporting audit data by outcome:

  auditrpt -a [s | f]

- Reporting audit data starting from *time*:

  auditrpt -s *time*

- Reporting audit data involving use of specific privileges:

  auditrpt -p *priv*,**...**

  (Valid only if the Enhanced Security Utilities are installed.)

- Reporting audit data prior to and including *time*:

  auditrpt -h *time*

- Including LWP ID information in an audit report:

  auditrpt -x...

- Reporting audit data “backwards,” with most recent information displayed first:

  auditrpt -b

- Reporting audit data as it is written to the log file:

  ```
  auditlog -v 0
  auditrpt -w
  ```

- Reporting audit events that satisfy at least one of the specified auditing criteria.

  auditrpt -o

- Specifying a directory containing the audit map files:

  auditrpt -m *mapdir*

- Specifying input audit records from standard input

  auditrpt -i

- Specifying the selected event types/classes for the Extended Trusted Applications.

  auditrpt -z[!] event[, . . .]

- Specifying the absolute or relative pathname of the audit log(s) to use

  auditrpt log[. . .]

6
Maintaining the Auditing System

6
Maintaining the Auditing System

Introduction

This chapter provides information on maintaining the auditing subsystem. The major maintenance activity is the archiving of audit information. Since breaches in security are not always detected while they occur, it is recommended that all audit event log files be archived. An archived log file may prove to be valuable in analyzing a security problem. Audit event log files should not be deleted from archival media (for example, tape). If you must delete old log files to provide room for archiving newer log files, you should examine the old audit data very carefully before deleting it, making sure there are no unusual patterns of activity.

At times, you may need to recover audit data from the audit buffer(s) in main memory. You can use the `abuf` function of the **`crash`** command to recover audit data from the buffer(s). This chapter also provides instructions on using the **`crash`** command.

Archiving Audit Information

This section describes how to archive audit information. The audit information consists of two basic sets of data:

- the audit event log files
- the audit map files

It is necessary to archive both sets of data, otherwise you may get misleading information if you process a log file with audit map files that do not reflect the system that generated the log file.

You can archive audit information with the **`cpio(1M)`** command.

The **`cpio`** command archives a file when you use the **`-o`** (copy out) option. The standard input is assumed to be a list of path names of files that will be archived. The output of the command is copied to standard out unless the **`-O`** option is used to specify an output file.

In the following example, the **`ls`** command generates a list of the audit event log files contained in **`/var/audit`** directory. It is assumed that all the log files have the node name `beowulf`, so that the command **`ls /var/audit/*beowulf`** will list the names of all the log files. The following command copies all the log files to the device **`/dev/rmt/c0s0`.**

```
/bin/ls /var/audit/*beowulf | /bin/cpio -o -O \
/dev/rmt/c0s0
```

In the next example, the following command will copy the audit map files to the device **`/dev/rmt/c0s0.`**

```
/bin/ls /var/audit/auditmap | /bin/cpio -o -O \
/dev/rmt/c0s0
```

NOTE

The example above would also archive the old audit map files (audit map files that have the prefix "`o`"). If you do not want the old audit map files on your archive tape, you should either remove them before creating the archive or use a file list as input to the **`cpio`** command.

For further information on the use of the **`cpio(1M)`** command, refer to the respective manual pages in the online man pages.

Prior to using a device, it is necessary to allocate the device to the appropriate user. The audit administrator role, `AUD,` does not allow for this capability. Refer to the "Managing Storage Devices" chapter of the *System Administration* for detailed information on allocating devices.

You should label the archival medium so that you will know the dates covered by the archived audit event log files.

Recovering Audit Data from System Dumps

In the case of a system crash, the data contained in the audit buffers can be recovered and subsequently examined. The `abuf` function of the **`crash`** command can be used to locate and write the audit data to a file. The file can then be processed by the **`auditrpt`** command.

This section assumes that you already know how to use the **`crash`** command. If you do not, read the **`crash(1M)`** manual page in the online man pages before reading this section.

NOTE

On systems with the Enhanced Security Utilities installed, the **`crash`** command is available only in maintenance mode.

The `abuf` function of the **`crash`** command can take two options. The **`-w`** option will redirect the output to a specified file and a format option to control the way the data is printed. The following format options are available:

-b	print data in binary format
-c	print data in character format
-d	print data in decimal format
-o	print data in octal format
-x	print data in hexadecimal format

The default format is hexadecimal (**-x**).

In most cases, you will want to write the audit data to a file for subsequent processing by the **auditrpt** command. The **auditrpt** command requires the data to be in binary format. Therefore, the **-w** option must be used to specify a file along with the **-b** option to indicate the data is to be written in binary format. After you are inside the **crash** program itself, you would enter the following command to save the audit buffer data to a file:

```
abuf -w/var/audit/abuf.info -b
q
```

You can now use the following command to display all the audit data written to the **/var/audit/abuf.info** file:

```
auditrpt /var/audit/abuf.info
```

You can also use the **crash** command to display the audit buffer data on a running system. If you do not specify a dump file with the **-d** option, **crash** by default uses the file **/dev/mem.** Screen 6-1 is an example of the current audit buffer information generated by the `abuf` function of the **crash** command.

```
> abuf -d -w/home/adf/audit_dmp
AUDIT BUFFER CONTROL STRUCTURE
a_vhigh  = 20480
a_bsize  = 20480
a_curbuf = 0
a_wptr   = d10324f4
a_bp     = d1032000
a_cp     = d1020f90
a_ap     = 0
a_asize  = 0


NUMBER OF AUDIT BUFFERS = 2


BUFFER     ADDR        INUSE     FLAGS
0          d1032000    1268      0x0
1          d1037000    0         0x0


Audit Buffer Size: 20480 bytes
Amount of Data: 1268 bytes
d1032000:  0000000026   0000000056   0016777515   0000001264
d1032010:  0709486268   0000000000   0000000000   0000000000
d1032020:  0000000003   0000000000   0000000003   0000000579
d1032030:  0000000056   0000000026   0000000027   0000000092
d1032040:  0016777216   0000001264   0709486268   0000000000
d1032050:  0000000014   0000000003   0000000000   0000000001
d1032060:  0000000002   0000000004   0000000005   0000000006
d1032070:  0000000007   0000000008   0000000010   0000000012
d1032080:  0000000023   0000000047   0000000009   0000000092
d1032090:  0000000027   0000720919   0000000036   0016777515
d10320a0:  0000001264   0709486268   0000000000   0000000001
d10320b0:  0000000036   0000720919   0000851994   0000000056
d10320c0:  0016777516   0000001265   0709486268   0000000000
   .          .            .            .            .
   .          .            .            .            .
   .          .            .            .            .
   .          .            .            .            .
   .          .            .            .            .
   .          .            .            .            .
   .          .            .            .            .
   .          .            .            .            .
d1033390:  0000851995   0000851994   0000000056   0016777271
d10333a0:  0000000409   0709406400   0000000000   0000000000
d10333b0:  0000000000   0000000003   0000000000   0000000000
d10333c0:  0000000240   0000000056   0000851994   0000851995
```

Screen 6-1. abuf Function of crash Command

In the case of an auditing subsystem error, **`crash`** can be used to recover the data contained within the audit buffers of main memory. Simply execute **`crash`** with no options, then use the `abuf` **`-w`** *`filename`* **`-b`** *`function`* to write the audit data to a file.

CAUTION

When you save audit data using the **`crash`** command, you should ensure that the resulting file has the correct ownership and access control permissions. Otherwise, unauthorized users could gain access to audit data.

The **`proc`** function of the **`crash`** command has also been modified to display additional information related to auditing. If auditing was enabled when the system dump was created, the **`proc -f`** function of the **`crash`** command displays the information shown in Screen 6-2 for each specified process at the end of its report for the process:

```
Process Credentials: uid: 0, gid: 3, real uid: 0, real gid: 3 lid: 0
       Process Privileges: working: 07ffffff   maximum: 07ffffff
       Process Audit Structure:
               Current Working Directory: /
               Flags: 0
               Process Event Mask: 07ff0000 1a010018 00000000 00000000
                                   00000000 00000000 00000000 00000000
               User    Event Mask: 00000000 00000000 00000000 00000000
                                   00000000 00000000 00000000 00000000
       LWP Audit Structure:
               Flags: 0
               Event Number: 0
               Record Sequence Number: 36
               Starting time of Event: 29
               Bufp:   0x42a1f690
               Obufp:  0x0
               Frecp:  0x44344520
               Cmpcnt:          0
```

Screen 6-2. proc -f Output with Auditing Enabled

If auditing was disabled when the system dump was created, the **`proc -f`** function of the **`crash`** command displays the information shown in Screen 6-3 at the end of its report for each specified process:

```
proc -f
     .
     .
     .
   proc: auditing disabled - no process audit structure
```

Screen 6-3. proc -f Output with Auditing Disabled

The **`proc -fn`** function prints the audit related information in an expanded format. If auditing was enabled when the system dump was created, the **`proc -fn`** function of the **`crash`** command displays the information shown in Screen 6-4 for each specified process at the end of its report for the process:

If auditing was disabled when the system dump was created, the **`proc -fn`** function of the **`crash`** command displays the information shown in Screen 6-5 at the end of its report for each specified process:

```
SLOT ST  PID  PPID   SID   UID   LWP  PRI ENG  EVENT       NAME        FLAGS
 17  s   243     1   243     0     0   80   0  e-40e42490  rpc.rusersd
        Process Credentials: uid: 0, gid: 0, real uid: 0, real gid: 0 lid: 0
        Process Privileges:
                working: allprivs
                maximum: allprivs
        Process Audit Structure:
                Current Working Directory: ?
                Flags: NONE
                Process Event Mask:
add_grp,add_usr,add_usr_grp,assign_lid,assign_nm,audit_buf,audit_ctl,audit_dmp
,audit_evt,audit_log,audit_map,date,deactivate_lid,del_nm,init,mod_grp,mod_usr
,
                User    Event Mask:     none
        LWP Audit Structure:
                Flags: NONE
        Event Number: NULL
        Record Sequence Number: 0
        Starting time of Event: 0
                Bufp:   0x40e42bd0
                Obufp:  0x0
                Frecp:  0x44c52160
                Cmpcnt:          0
```

Screen 6-4. proc -fn Output Using Default Symbol Table, Auditing Enabled

```
proc -fn
    .
    .
    .
     proc: auditing disabled - no process audit structure
```

Screen 6-5. proc -fn Output Using Default Symbol Table, Auditing Disabled

A
Summary of Auditable Events and Classes

Introduction

This appendix contains an Auditable Event Table and an Event Class Table. These tables provide a quick reference; detailed information is contained in the “Auditable Events” chapter of this guide.

Table of Auditable Events

In the following table each entry consists of:

- the event
- a brief description of the event
- the system call or user level command that triggers the event

NOTE

An event is applicable only if the system software configuration supports the stated system call or command. However, non-applicable events may be reserved to allow release compatibility.

The Auditable Events are either all selected (all) or not selected (none). The fixed events are listed in Table A-1 and the selectable events are listed in Table A-2.

Table A-1. Table of Fixed Events

Event	Event Description	System Call or Command
`add_grp`	adding groups	**`groupadd(1M)`**
`add_usr`	adding user attributes	**`useradd(1M)`**
`add_usr_grp`	add group members	**`addgrpmem(1M)`**, **`useradd(1M)`**, **`usermod(1M)`**

Table A-1. Table of Fixed Events (Cont.)

Event	Event Description	System Call or Command
audit_buf	set audit buffer attributes	**auditbuf(2)**
audit_ctl	enable/disable auditing	**auditoff(1M), auditon(1M), auditctl(2)**
audit_dmp	record audit_dmp failures	**auditdmp(2)**
audit_evt	set auditable events	**auditset(1M), auditevt(2)**
audit_log	set log file attributes	**auditlog(1M), auditlog(2)**
audit_map	create audit map files	**auditmap(1M)**
date	change the date	**adjtime(2), settimeofday(2), stime(2), posix_clocks(2)**
init	change **init** states	**init(1M)**
mod_grp	modify group information	**groupmod(1M)**
mod_usr	modify user information	**usermod(1M)**

Table A-2. Table of Selectable Events

Event	Event Description	System Call or Command
access	determine accessibility of a file	**access(2)**
acct_off	disable accounting	**acct(2)**
acct_on	enable accounting	**acct(2)**
acct_sw	switch accounting files	**acct(2)**
bad_auth	bad logname or password	**login(1)**
bad_lvl	bad login level	**login(1)**
cancel_job	cancel a print job	**cancel(1), lpsched(1M)**
chg_dir	change working directory	**chdir(2), fchdir(2)**
chg_root	change root directory	**chroot(2)**
chg_times	change file access and modification times	**utime(2)**
create	create a new file system object	**creat(2)**

Table A-2. Table of Selectable Events (Cont.)

Event	Event Description	System Call or Command
cron	cron job	**cron(1M)**
dac_mode	change mode of an object	**chmod(2), fchmod(2)**
dac_own_grp	change group of an object	**chown(2), fchown(2), lchown(2)**
disp_attr	display device attributes	**devstat(2), fdevstat(2)**
exec	execute an object	**exec(2), exece(2)**
exit	terminate a process	**exit(2)**
fcntl	file control operations	**fcntl(2)**
file_priv	change privileges on a file	**filepriv(2)**
fork	create a new process	**fork(2), fork1(2), forkall(2), vfork(2)**
iocntl	control I/O operations	**ioctl(2)**
ipc_acl	change IPC access control lists	**aclipc(2)**
keyctl	allow processors to be used	**keyctl(2)**
kill	post a signal	**kill(2), sigsendset(2)**
link	create a link to an object	**link(2)**
login	use of a login scheme	**login(1)**
logoff	terminate a login session	**ttymon(1)**
lp_admin	administrative use of lp system	**lpadmin(1M), lpfilter(1M), lpforms(1M), lpmove(1M), lpsched(1M), lpshut(1M), lpusers(1M)**
lp_misc	miscellaneous use of lp system	**lpsched(1M)**
lwp_create	create a new process	**_lwp_create(2)**
lwp_exit	terminate lwp	**_lwp_exit(2)**
lwp_kill	post a signal to lwp	**_lwp_kill(2)**
lwp_setbias	change processor for lwp	**cpu_bias(2)**
lwp_setrun	change processor for lwp	**cpu_bias(2)**

Table A-2. Table of Selectable Events (Cont.)

Event	Event Description	System Call or Command
`mk_dir`	make a directory	**mkdir(2)**
`mk_node`	make a special file	**mknod(2), xmknod(2)**
`modadm`	register a module	**modadm(2)**
`modload`	load a module	**modload(2)**
`moduload`	unload a module	**moduload(2)**
`modpath`	modify module search path	**modpath(2)**
`mount`	mount a device or file system	**mount(2)**
`msg_ctl`	message control operations	**msgctl(2)**
`msg_get`	get message queue	**msgget(2)**
`msg_op`	message operations	**msgop(2)**
`online`	take processor on/offline	**online(2)**
`open_rd`	open an object for reading	**open(2)**
`open_wr`	open an object for writing	**open(2)**
`passwd`	change password	**passwd(1)**
`pipe`	create an unnamed pipe	**pipe(2)**
`pm_denied`	failed use of privilege	**NA**
`prt_job`	start/end of printer job	**lp(1)**
`prt_lvl`	override output level	**lp(1)**
`recvfd`	receive file descriptor	NA
`rm_dir`	remove a directory	**rmdir(2)**
`sched_fc`	fixed class scheduler operations	**priocntl(2)**
`sched_fp`	fixed priority scheduler operations	**priocntl(2)**
`sched_lk`	lock a process into memory	**plock(2), memcntl(2)**
`sched_ts`	time-sharing scheduler operations	**priocntl(2)**
`sem_ctl`	semaphore control operations	**semctl(2)**
`sem_get`	get the set of semaphores	**semget(2)**
`sem_op`	semaphore operations	**semop(2)**
`set_attr`	set device attributes	**devstat(2), fdevstat(2)**
`set_gid`	change group ID	**setgid(2)**
`set_grps`	set multiple groups	**setgroups(2)**

Table A-2. Table of Selectable Events (Cont.)

Event	Event Description	System Call or Command
`set_pgrps`	set process groups	**`setpgrp(2)`**, **`setpgid(2)`**
`set_sid`	assign a session ID	**`setsid(2)`**
`set_uid`	change user ID	**`setuid(2)`**
`setrlimit`	set resource limits	**`setrlimit(2)`**
`shm_bind`	a shared memory binding	**`shmbind(2)`**
`shm_ctl`	shared memory control operations	**`shmctl(2)`**
`shm_get`	get shared memory identifier	**`shmget(2)`**
`shm_op`	shared memory operations	**`shmop(2)`**
`status`	get file status	**`stat(2)`**, **`xstat(2)`**, **`fstat(2)`**, **`fxstat(2)`**
`sym_create`	create a symbolic link	**`symlink(2)`**
`sym_status`	get status of symbolic link	**`lstat(2)`**, **`lxstat(2)`**, **`symlink(2)`**
`tfadmin`	administrative commands	**`tfadmin(1M)`**
`trunc_lvl`	truncate a printed level	**`lp(1)`**
`ulimit`	resource limits	**`ulimit(2)`**
`umount`	unmount a device or file system	**`umount(2)`**
`unlink`	unlink an object	**`unlink(2)`**

Auditable Event Classes

Table A-3 lists the predefined event classes defined in the **`/etc/security/audit/classes`** file.

Table A-3. Auditable Event Classes

Event Class	Events in Class
acct	acct_off acct_sw acct_on
audit	audit_buf audit_ctl audit_dmp audit_evt audit_log audit_map
cov_chan	cov_chan1 cov_chan2 cov_chan3 cov_chan4 cov_chan5 cov_chan6 cov_chan7 cov_chan8
dac	dac_mode dac_own_grp file_acl ipc_acl
device	disp_attr mount set_attr set_lvl_rng umount
dir_access	access chg_dir chg_root chg_times status sym_status
dir_make	link mk_dir mk_mld rm_dir sym_create unlink
file_access	access chg_times open_rd open_wr status sym_status
file_attr	add_grp add_usr add_usr_grp mod_grp mod_usr
file_make	create link mk_node sym_create unlink
id_auth	bad_auth bad_lvl cron def_lvl login passwd
io_cntl	fcntl iocntl
module	modadm modload modpath moduload
msg	msg_ctl msg_get msg_op
path	chg_dir chg_root
printer	cancel_job lp_admin lp_misc page_lvl prt_job prt_lvl trunc_lvl
priv	file_priv pm_denied
process	exec exit fork kill set_gid set_grps set_pgrps set_sid set_uid
res_limit	setrlimit ulimit
sched	sched_lk sched_ts sched_fc sched_fp
sem	sem_ctl sem_get sem_op
shm	shm_ctl shm_get shm_op
sym_link	sym_create sym_status
use_lwp	lwp_create lwmp_exit lwp_kill

Index

Spine for 1/2” Binder

Product Name: 0.5" from top of spine, Helvetica, 36 pt, Bold

PowerMAX OS

Volume Number (if any): Helvetica, 24 pt, Bold

Volume Name (if any): Helvetica, 18 pt, Bold

Manual Title(s): Helvetica, 10 pt, Bold, centered vertically within space above bar, double space between each title

User/
Admin

Audit Trail
Admin
0890431

Bar: 1" x 1/8" beginning 1/4" in from either side

Part Number: Helvetica, 6 pt, centered, 1/8" up