# Chapter 19. Security Event Auditing

## Table of Contents

## 19.1. Synopsis

The FreeBSD operating system includes support for security event auditing. Event auditing supports reliable, fine-grained, and configurable logging of a variety of security-relevant system events, including logins, configuration changes, and file and network access. These log records can be invaluable for live system monitoring, intrusion detection, and postmortem analysis. FreeBSD implements Sun™'s published Basic Security Module (BSM) Application Programming Interface (API) and file format, and is interoperable with the Solaris™ and Mac OS® X audit implementations.

This chapter focuses on the installation and configuration of event auditing. It explains audit policies and provides an example audit configuration.

After reading this chapter, you will know:

- What event auditing is and how it works.
- How to configure event auditing on FreeBSD for users and processes.
- How to review the audit trail using the audit reduction and review tools.

Before reading this chapter, you should:

- Understand UNIX® and FreeBSD basics (FreeBSD Basics).

- Be familiar with the basics of kernel configuration/compilation (Configuring the FreeBSD Kernel).
- Have some familiarity with security and how it pertains to FreeBSD (Security).

**Warning**

The audit facility has some known limitations. Not all security-relevant system events are auditable and some login mechanisms, such as Xorg-based display managers and third-party daemons, do not properly configure auditing for user login sessions.

The security event auditing facility is able to generate very detailed logs of system activity. On a busy system, trail file data can be very large when configured for high detail, exceeding gigabytes a week in some configurations. Administrators should take into account the disk space requirements associated with high volume audit configurations. For example, it may be desirable to dedicate a file system to **/var/audit** so that other file systems are not affected if the audit file system becomes full.

## 19.2. Key Terms

The following terms are related to security event auditing:

- *event*: an auditable event is any event that can be logged using the audit subsystem. Examples of security-relevant events include the creation of a file, the building of a network connection, or a user logging in. Events are either "attributable", meaning that they can be traced to an authenticated user, or "non-attributable". Examples of non-attributable events are any events that occur before authentication in the login process, such as bad password attempts.
- *class*: a named set of related events which are used in selection expressions. Commonly used classes of events include "file creation" (fc), "exec" (ex), and "login_logout" (lo).
- *record*: an audit log entry describing a security event. Records contain a record event type, information on the subject (user) performing the action, date and time information, information on any objects or arguments, and a success or failure condition.
- *trail*: a log file consisting of a series of audit records describing security events.

Trails are in roughly chronological order with respect to the time events completed. Only authorized processes are allowed to commit records to the audit trail.

- *selection expression*: a string containing a list of prefixes and audit event class names used to match events.
- *preselection*: the process by which the system identifies which events are of interest to the administrator. The preselection configuration uses a series of selection expressions to identify which classes of events to audit for which users, as well as global settings that apply to both authenticated and unauthenticated processes.
- *reduction*: the process by which records from existing audit trails are selected for preservation, printing, or analysis. Likewise, the process by which undesired audit records are removed from the audit trail. Using reduction, administrators can implement policies for the preservation of audit data. For example, detailed audit trails might be kept for one month, but after that, trails might be reduced in order to preserve only login information for archival purposes.

## 19.3. Audit Configuration

User space support for event auditing is installed as part of the base FreeBSD operating system. Kernel support is available in the **GENERIC** kernel by default, and auditd(8) can be enabled by adding the following line to **/etc/rc.conf**:

```
auditd_enable="YES"
```

Then, start the audit daemon:

```
# service auditd start
```

Users who prefer to compile a custom kernel must include the following line in their custom kernel configuration file:

```
options AUDIT
```

### 19.3.1. Event Selection Expressions

Selection expressions are used in a number of places in the audit configuration to determine which events should be audited. Expressions contain a list of event classes to match. Selection expressions are evaluated from left to right, and two expressions are combined by appending one onto the other.

Default Audit Event Classes summarizes the default audit event classes:

**Table 1. Default Audit Event Classes**

| Class Name | Description | Action |
|---|---|---|
| all | all | Match all event classes. |
| aa | authentication and authorization | |
| ad | administrative | Administrative actions performed on the system as a whole. |
| ap | application | Application defined action. |
| cl | file close | Audit calls to the `close` system call. |
| ex | exec | Audit program execution. Auditing of command line arguments and environmental variables is controlled via audit_control(5) using the `argv` and `envv` parameters to the `policy` setting. |

| Class Name | Description | Action |
|---|---|---|
| fa | file attribute access | Audit the access of object attributes such as stat(1) and pathconf(2). |
| fc | file create | Audit events where a file is created as a result. |
| fd | file delete | Audit events where file deletion occurs. |
| fm | file attribute modify | Audit events where file attribute modification occurs, such as by chown(8), chflags(1), and flock(2). |
| fr | file read | Audit events in which data is read or files are opened for reading. |
| fw | file write | Audit events in which data is written or files are written or modified. |
| io | ioctl | Audit use of the `ioctl` system call. |
| ip | ipc | Audit various forms of Inter-Process Communication, including |

| Class Name | Description | Action |
|---|---|---|
| | | POSIX pipes and System V IPC operations. |
| lo | login_logout | Audit login(1) and logout(1) events. |
| na | non attributable | Audit non-attributable events. |
| no | invalid class | Match no audit events. |
| nt | network | Audit events related to network actions such as connect(2) and accept(2). |
| ot | other | Audit miscellaneous events. |
| pc | process | Audit process operations such as exec(3) and exit(3). |

These audit event classes may be customized by modifying the **audit_class** and **audit_event** configuration files.

Each audit event class may be combined with a prefix indicating whether successful/ failed operations are matched, and whether the entry is adding or removing matching for the class and type. Prefixes for Audit Event Classes summarizes the available prefixes:

**Table 2. Prefixes for Audit Event Classes**

| Prefix | Action |
|---|---|
| + | Audit successful events in this class. |
| - | Audit failed events in this class. |
| ^ | Audit neither successful nor failed events in this class. |
| ^+ | Do not audit successful events in this class. |
| ^- | Do not audit failed events in this class. |

If no prefix is present, both successful and failed instances of the event will be audited.

The following example selection string selects both successful and failed login/logout events, but only successful execution events:

```
lo,+ex
```

## 19.3.2. Configuration Files

The following configuration files for security event auditing are found in **/etc/security**:

- **audit_class**: contains the definitions of the audit classes.
- **audit_control**: controls aspects of the audit subsystem, such as default audit classes, minimum disk space to leave on the audit log volume, and maximum audit trail size.
- **audit_event**: textual names and descriptions of system audit events and a list of which classes each event is in.

- **audit_user**: user-specific audit requirements to be combined with the global defaults at login.
- **audit_warn**: a customizable shell script used by auditd(8) to generate warning messages in exceptional situations, such as when space for audit records is running low or when the audit trail file has been rotated.

> ⚠ **Warning**
>
> Audit configuration files should be edited and maintained carefully, as errors in configuration may result in improper logging of events.

In most cases, administrators will only need to modify **audit_control** and **audit_user**. The first file controls system-wide audit properties and policies and the second file may be used to fine-tune auditing by user.

### 19.3.2.1. The **audit_control** File

A number of defaults for the audit subsystem are specified in **audit_control**:

```
dir:/var/audit
dist:off
flags:lo,aa
minfree:5
naflags:lo,aa
policy:cnt,argv
filesz:2M
expire-after:10M
```

The `dir` entry is used to set one or more directories where audit logs will be stored. If more than one directory entry appears, they will be used in order as they fill. It is common to configure audit so that audit logs are stored on a dedicated file system, in order to prevent interference between the audit subsystem and other subsystems if the file system fills.

If the `dist` field is set to `on` or `yes`, hard links will be created to all trail files in **/var/audit/dist**.

The `flags` field sets the system-wide default preselection mask for attributable events. In the example above, successful and failed login/logout events as well as authentication and authorization are audited for all users.

The `minfree` entry defines the minimum percentage of free space for the file system where the audit trail is stored.

The naflags entry specifies audit classes to be audited for non-attributed events, such as the login/logout process and authentication and authorization.

The policy entry specifies a comma-separated list of policy flags controlling various aspects of audit behavior. The cnt indicates that the system should continue running despite an auditing failure (this flag is highly recommended). The other flag, argv, causes command line arguments to the execve(2) system call to be audited as part of command execution.

The filesz entry specifies the maximum size for an audit trail before automatically terminating and rotating the trail file. A value of 0 disables automatic log rotation. If the requested file size is below the minimum of 512k, it will be ignored and a log message will be generated.

The expire-after field specifies when audit log files will expire and be removed.

### 19.3.2.2. The **audit_user** File

The administrator can specify further audit requirements for specific users in **audit_user**. Each line configures auditing for a user via two fields: the alwaysaudit field specifies a set of events that should always be audited for the user, and the neveraudit field specifies a set of events that should never be audited for the user.

The following example entries audit login/logout events and successful command execution for root and file creation and successful command execution for www. If used with the default **audit_control**, the lo entry for root is redundant, and login/logout events will also be audited for www.

```
root:lo,+ex:no
www:fc,+ex:no
```

## 19.4. Working with Audit Trails

Since audit trails are stored in the BSM binary format, several built-in tools are available to modify or convert these trails to text. To convert trail files to a simple text format, use praudit. To reduce the audit trail file for analysis, archiving, or printing purposes, use auditreduce. This utility supports a variety of selection parameters, including event type, event class, user, date or time of the event, and the file path or object acted on.

For example, to dump the entire contents of a specified audit log in plain text:

```
# praudit /var/audit/AUDITFILE
```

Where *AUDITFILE* is the audit log to dump.

Audit trails consist of a series of audit records made up of tokens, which `praudit` prints sequentially, one per line. Each token is of a specific type, such as `header` (an audit record header) or `path` (a file path from a name lookup). The following is an example of an `execve` event:

```
header,133,10,execve(2),0,Mon Sep 25 15:58:03 2006, + 384 msec
exec arg,finger,doug
path,/usr/bin/finger
attribute,555,root,wheel,90,24918,104944
subject,robert,root,wheel,root,wheel,38439,38032,42086,128.232.9.1
return,success,0
trailer,133
```

This audit represents a successful `execve` call, in which the command `finger doug` has been run. The `exec arg` token contains the processed command line presented by the shell to the kernel. The `path` token holds the path to the executable as looked up by the kernel. The `attribute` token describes the binary and includes the file mode. The `subject` token stores the audit user ID, effective user ID and group ID, real user ID and group ID, process ID, session ID, port ID, and login address. Notice that the audit user ID and real user ID differ as the user `robert` switched to the `root` account before running this command, but it is audited using the original authenticated user. The `return` token indicates the successful execution and the `trailer` concludes the record.

XML output format is also supported and can be selected by including `-x`.

Since audit logs may be very large, a subset of records can be selected using `auditreduce`. This example selects all audit records produced for the user `trhodes` stored in **AUDITFILE**:

```
# auditreduce -u trhodes /var/audit/AUDITFILE | praudit
```

Members of the `audit` group have permission to read audit trails in **/var/audit**. By default, this group is empty, so only the `root` user can read audit trails. Users may be added to the `audit` group in order to delegate audit review rights. As the ability to track audit log contents provides significant insight into the behavior of users and processes, it is recommended that the delegation of audit review rights be performed with caution.

## 19.4.1. Live Monitoring Using Audit Pipes

Audit pipes are cloning pseudo-devices which allow applications to tap the live audit record stream. This is primarily of interest to authors of intrusion detection and system monitoring applications. However, the audit pipe device is a convenient way for the administrator to allow live monitoring without running into problems with audit trail file ownership or log rotation interrupting the event stream. To track the live audit event stream:

```
# praudit /dev/auditpipe
```

By default, audit pipe device nodes are accessible only to the `root` user. To make them accessible to the members of the `audit` group, add a `devfs` rule to **/etc/devfs.rules**:

```
add path 'auditpipe*' mode 0440 group audit
```

See devfs.rules(5) for more information on configuring the devfs file system.

**Warning**

It is easy to produce audit event feedback cycles, in which the viewing of each audit event results in the generation of more audit events. For example, if all network I/O is audited, and `praudit` is run from an SSH session, a continuous stream of audit events will be generated at a high rate, as each event being printed will generate another event. For this reason, it is advisable to run `praudit` on an audit pipe device from sessions without fine-grained I/O auditing.

## 19.4.2. Rotating and Compressing Audit Trail Files

Audit trails are written to by the kernel and managed by the audit daemon, auditd(8). Administrators should not attempt to use newsyslog.conf(5) or other tools to directly rotate audit logs. Instead, `audit` should be used to shut down auditing, reconfigur the audit system, and perform log rotation. The following command causes the audit daemon to create a new audit log and signal the kernel to switch to using the new log. The old log will be terminated and renamed, at which point it may then be manipulated by the administrator:

```
# audit -n
```

If auditd(8) is not currently running, this command will fail and an error message will be produced.

Adding the following line to **/etc/crontab** will schedule this rotation every twelve hours:

```
0     */12       *       *       *       root    /usr/sbin/audit -
```

The change will take effect once **/etc/crontab** is saved.

Automatic rotation of the audit trail file based on file size is possible using `filesz` in **audit_control** as described in The audit_control File.

As audit trail files can become very large, it is often desirable to compress or otherwise archive trails once they have been closed by the audit daemon. The **audit_warn** script can be used to perform customized operations for a variety of audit-related events, including the clean termination of audit trails when they are rotated. For example, the following may be added to **/etc/security/audit_warn** to compress audit trails on close:

```
#
# Compress audit trail files on close.
#
if [ "$1" = closefile ]; then
        gzip -9 $2
fi
```

Other archiving activities might include copying trail files to a centralized server, deleting old trail files, or reducing the audit trail to remove unneeded records. This script will be run only when audit trail files are cleanly terminated. It will not be run on trails left unterminated following an improper shutdown.

**Last modified on**: February 18, 2025 by Fernando Apesteguía

‹ Prev Home Next ›

FreeBSD

English System

**About**

FreeBSD

FreeBSD Foundation

Get FreeBSD

Code of Conduct

Security Advisories

**Documentation**

Documentation portal

Manual pages

Presentations and papers

Previous versions

4.4BSD Documents

Wiki

**Community**

Get involved

Community forum

Mailing lists

IRC Channels

Bug Tracker

**Legal**

Donations

Licensing

Privacy Policy

Legal notices

© 1994-2025 The FreeBSD Project. All rights reserved

Made with ♥ by the FreeBSD Community