Oracle Linux 8

Auditing the System With Auditd and Rsyslogd

G24427-01
May 2025

Oracle Linux 8 Auditing the System With Auditd and Rsyslogd,

G24427-01

Contents

Preface

Documentation License iv
Conventions iv
Documentation Accessibility iv
Access to Oracle Support for Accessibility iv
Diversity and Inclusion iv

1 Configuring and Using Auditing

Creating Audit Rules 1-1
Generating Audit Reports 1-2

2 Working With System Log files

About rsyslog 2-1
Logging Configuration Reference 2-2
Configuring rsyslog 2-3
Configuring Logwatch 2-5

3 Using Process Accounting

Preface

Oracle Linux 8: Auditing the System With Auditd and Rsyslogd describes how to generate and analyze system logs to audit kernel level processes and detect unauthorized activity.

Documentation License

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at https://www.oracle.com/corporate/accessibility/.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and

the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

1

Configuring and Using Auditing

Auditing collects data at the kernel level for analysis to identify unauthorized activity. Auditing also often collects data in greater detail than system logging does. The process of examining audit trails to find events of interest can be challenging, but it can be automated.

Some definitions in the audit configuration file, `/etc/audit/auditd.conf`, include the following:

- Data retention policy
- Maximum size of the audit volume
- Action to take if the capacity of the audit volume is exceeded
- Locations of local and remote audit trail volumes

The default audit trail volume is the `/var/log/audit/audit.log` file. For more information, see the `auditd.conf(5)` manual page.

For a hands-on tutorial on using the auditing tools on Oracle Linux, see Audit Oracle Linux with Auditd.

Creating Audit Rules

Create audit rules in the `/etc/audit/audit.rules` configuration file to collect more relevant data for analysis.

By default, auditing captures specific events such as system logins, modifications to accounts, and `sudo` actions. You can configure auditing to capture detailed system call activity and modifications to certain files. The kernel audit daemon (`auditd`) records the events that you configure, including the event type, a timestamp, the associated user ID, and whether a system call succeeded or failed.

The entries in the audit rules file, `/etc/audit/audit.rules`, configures which events are audited. Each rule is a command line option that's passed to the `auditctl` command. Configure this file to match organization's security policy.

The following are examples of rules that can be set in the `/etc/audit/audit.rules` file:

To record all unsuccessful exits from `open` and `truncate` system calls for files and store the information in the `/etc` directory hierarchy, add the following line:

```
-a exit,always -S open -S truncate -F /etc -F success=0
```

To record all files opened by a user with a `UID` value of `10`, add the following line:

```
-a exit,always -S open -F uid=10
```

To record all files that have been revised or whose attributes were changed by any user who originally signed in with a `UID` value of `500` or greater, add the following line:

```
-a exit,always -S open -F auid>=500 -F perm=wa
```

To record requests for write or for file attribute change access, you can store the records in the `/etc/sudoers` file and tag such a record with the string `sudoers-change`:

```
-w /etc/sudoers -p wa -k sudoers-change
```

To record requests for write and for file attribute change access and store records in the `/etc` directory hierarchy, add the following line:

```
-w /etc/ -p wa
```

To require a reboot after changing the audit configuration, add the following line:

```
-e 2
```

> **Note:**
>
> Defining a rule to reboot at the end of the `/etc/audit/audit.rules` file is considered good security practice.

For more examples of audit rules, see also the `auditctl(8)` and `audit.rules(7)` manual pages.

Generating Audit Reports

Search audit data and and generate audit reports with the `ausearch` and `aureport` commands.

Stringent auditing requirements generate large amounts of audit data and can impose a significant performance overhead. Some site security policies stipulate that a system must shut down if events can't be recorded because the audit volumes have exceeded their capacity. Sending auditing data to separate file systems in rotation to prevent overspill and to ease backups is considered good practice.

Tagging audit records can make it more straightforward to search an audit volume by running the `ausearch` command and referring to those tags. For example, to examine records that are tagged with the string `sudoers-change`, run the following command:

```
sudo ausearch -k sudoers-change
```

The `aureport` command generates summaries of audit data. For example, the following command generates a report that shows every sign-in event from 1 second after midnight on the previous day until the current time:

```
sudo aureport -l -i -ts yesterday -te now
```

For more information, see the `ausearch(8)` and `aureport(8)` manual pages.

You can also set up `cron` jobs or `systemd` timers that run `aureport` periodically to generate reports of interest. For more information about scheduling those tasks, see Oracle Linux 8: Automating System Tasks With cron and Oracle Linux 8: Managing the System With systemd.

2
Working With System Log files

System log files contain messages about the system, kernel, services, and applications. The `journald` logging daemon, which is part of `systemd`, records system messages in nonpersistent journal files in memory. The `journald` daemon forwards messages to the system logging daemon, `rsyslog`.

Files in `/run` are volatile, so the log data is lost after a reboot unless you create the `/var/log/journal` directory. The `journalctl` command can be used to query the journal logs.

For more information, see the `journalctl(1)` and `systemd-journald.service(8)` manual pages.

About `rsyslog`

The `rsyslog` utility can be used to manage log messages on Oracle Linux 8 systems. The `/etc/rsyslog.conf` file stores configuration settings for the `rsyslogd` daemon.

The `/etc/rsyslog.conf` file contains global directives, module directives, and rules for the daemon or service. By default, `rsyslog` processes and archives only `syslog` messages, but `rsyslog` can also be configured to archive any other messages that `journald` forwards, such as kernel, boot, `initrd`, `stdout`, and `stderr` messages.

For more information about `rsyslog`, visit https://www.rsyslog.com/doc/ or run the following command install the `rsyslog-doc` package for offline documentation:

```
sudo dnf install -y rsyslog-doc
```

Offline documentation is installed in the `/usr/share/doc/rsyslog/html` directory.

Logging Configuration Reference

The `/etc/rsyslog.conf` file is divided into three main sections.

> **Important:**
>
> The format to configure parameters in `/etc/rsyslog.conf` can change between major versions. The following formats enable backward compatibility with previous configurations:
>
> - Basic or `sysklogd` format. This has been used since the beginning of system logging.
> - Legacy format, where directives are defined on their own specific lines in the file, with each directive being preceded by the dollar (`$`) sign, such as `$MainMsgQueueSize`.
> - Advanced format, which uses the `RainerScript` scripting language for configuring `rsyslog`.
>
> For more information about these formats, see the relevant sections in https://www.rsyslog.com/doc/.

Modules
Modules contain configuration parameters for processing messages. The processed or transformed messages can then be transmitted to various targets as required. Modules are classified into different categories, such as output, input, parser, library, and so on. For a complete list of these module classes, see the appropriate section in https://www.rsyslog.com/doc/. For a list of the modules, see the `rsyslog.conf(5)` manual page. Modules enable different `rsyslog` functionalities to become operative, so long as those modules are loaded. Modules are loaded through the `module load` directive as follows:

```
module(load="module-name")
```

> **Note:**
>
> The directive uses the advanced format for loading a module and replaces the `$ModLoad module-name` legacy format.

Global Directives
Global directives specify configuration options that apply to the `rsyslogd` daemon. A directive might specify the location of auxiliary files. A directive can also be a `module(load" ")` statement that applies global settings, such as the timestamp format to use for all messages, as shown in the following example:

```
module(load="builtin:omfile" Template=RSYSLOG_TraditionalFileFormat")
```

Because the module applies to all messages, the directive is specified under the Global Directives section.

Rules
Rules or rule sets configure how logged messages are managed.
A rule consists of two fields: a selector field and an action field. The two fields are separated by one more spaces or tabs.
The selector field has two parts, separated by a period, which are a facility keyword and a priority keyword. Facility keywords include `auth`, `authpriv`, `cron`, `daemon`, `kern`, and so on. Priority keywords include `debug`, `info`, `notice`, `warning`, and so on. Therefore `kern.*` selects kernel messages of all priority levels, but `kern.emerg` selects emergency kernel messages only.
For a list of both facility and priority selectors, see the `rsyslog.conf(5)` manual page.
The action field typically indicates to which log file the message content is written. For example, the following rule indicates that `cron` messages are stored in log files in the `/var/log/cron` directory:

```
cron.*   /var/log/cron
```

Configuring `rsyslog`

Configure `rsyslog` to include custom organization-specific behaviors.

You can customize `rsyslog` configuration in two ways:

- Edit the `/etc/rsyslog.conf` file.
- Create a configuration file and store it in the `/etc/rsyslog.d` directory. You can select this option to prevent custom configurations from being overwritten when system packages are updated.

Some changes are straightforward to implement within the `/etc/rsyslog.conf` file, such as configuring the log for a specific selector. For example, to change the log for `cron` messages to `cron_new`, add the following line:

```
cron.*     /var/log/cron_new
```

Restart the `rsyslog` service for the change to take effect.

Other changes often require more parameter definitions and steps.

For example, you can create a rule that uses TCP to forward messages to another server where system messages are logged. The following steps implement this sample rule:

1. Create a separate file, for example, `/etc/rsyslog.d/forwarding`, and set the parameters for TCP forwarding in that file, similar to the following:

   ```
   *.* action(type="omfwd"
     queue.filename="fwdRule1"
     queue.maxdiskspace="1g"
     queue.saveOnShutdown="on"
     queue.type="linkedlist"
     action.resumeRetryCount="-1"
     target="example.com" port="30514" protocol="tcp"
   )
   ```

   The following list explains the purpose and allowed values for each setting:

queue.filename
This is the prefix to be attached to the names of each backup file. The prefixed backup files are created in the location as specified by the `workDir` global directive, for example, `global(workDirectory="/var/log")`.

queue.maxdiskspace
Sets the space limit for log files.

queue.saveOnShutdown
Sets whether data is saved in memory if `rsyslog` shuts down.

queue.type
Allows a LinkedList in-memory queue.

action.resumeRetryCount
Sets number of retries. A setting of `-1` retries indefinitely if the remote host is unavailable.

target
This can be a remote host name or an IP address.

Based on the sample configuration, `rsyslog` forwards messages to the remote server `remote-host.com`. The `rsyslog` service also keeps the message in memory in case the remote server is unavailable. If `rsyslog` shuts down or has exhausted allotted memory, then `rsyslog` creates files on disk with the appropriate prefix to the file names.

2. Open the `/etc/rsyslog.conf` configuration file and ensure that the module for TCP syslog reception is loaded. Verify that the comment marks are removed from the following lines:

```
module(load="imtcp")
input(type="imtcp" port="514")
```

3. Also ensure that the global directive to include `/etc/rsyslog.d` files in `rsyslog` configuration is enabled. Verify that the following line isn't commented out:

```
include(file="/etc/rsyslog.d/*.conf" mode="optional")
```

4. Save the `/etc/rsyslog.conf` configuration file and exit.
5. Restart the `rsyslog` service by running the following command:

```
sudo systemctl restart rsyslog
```

A new system logging rule has been configured, and now forwards system messages to another server.

To manage the rotation and archival of the correct logs, edit the `/etc/logrotate.d/syslog` configuration file so that it references each of the log files that are defined in the `RULES` section of the `/etc/rsyslog.conf` configuration file.

To configure how often the logs are rotated and how many past copies of the logs are archived, edit the `/etc/logrotate.conf` configuration file.

For more information about log rotation, see the `logrotate(8)`, `logwatch(8)`, `rsyslogd(8)` and `rsyslog.conf(5)` manual pages.

Configuring Logwatch

Monitor areas of interest in the system logs with Logwatch.

After you install the `logwatch` package, the `/etc/cron.daily/0logwatch` script runs every night and sends an email report to the `root` user.

You can set local configuration options in the `/etc/logwatch/conf/logwatch.conf` file, and those settings override any in the main configuration file `/usr/share/logwatch/default.conf/logwatch.conf`, including the following:

- Log files to monitor, including log files that are stored for other hosts.
- Names of the services to monitor, or services to be excluded from monitoring.
- Level of detail to report.
- User that's sent emailed reports.

Configuring Logwatch on a log server to monitor the system logs for suspicious messages, and disabling Logwatch on individual log clients, is considered good practice.

You can disable high precision timestamps to improve readibility by adding the following entry to the `GLOBAL DIRECTIVES` section of the `/etc/rsyslog.conf` file on each system:

```
module(load="builtin:omfile" Template=RSYSLOG_TraditionalFileFormat")
```

You can also run `logwatch` directly from the command line.

For more information, see the `logwatch(8)` manual page.

3
Using Process Accounting

The tools contained within the `psacct` package implements the process accounting service.

The `psacct` package also includes the following utilities to monitor process activities:

`ac`
Displays connection times in hours for a user as recorded in the `wtmp` file (by default, `/var/log/wtmp`).

`accton`
Turns on process accounting to the specified file. If you don't specify a file name argument, process accounting is stopped. The default system accounting file is `/var/account/pacct`.

`lastcomm`
Displays information about command history as recorded in the system accounting file.

`sa`
Summarizes information about command history as recorded in the system accounting file.

> **Note:**
>
> As for any logging activity, ensure that the file system has enough space to store the system accounting and `wtmp` files. Monitor the size of the files and truncate them as needed.

For more information, see the `ac(1)`, `accton(8)`, `lastcomm(1)`, and `sa(8)` manual pages.