Oracle Linux 8

Configuring the File Access Policy Daemon

F88842-01
January 2024

Oracle Linux 8 Configuring the File Access Policy Daemon,

F88842-01

Contents

Preface

Oracle Linux 8: Configuring the File Access Policy Daemon describes how you can install and configure the File Access Policy Daemon, `fapolicyd`, on Oracle Linux to improve system security though policy rules that either allow or block specific applications from running.

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/.

For information about the accessibility of the Oracle Help Center, see the Oracle Accessibility Conformance Report at https://www.oracle.com/corporate/accessibility/templates/t2-11535.html.

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 About the File Access Policy Daemon

The File Access Policy Daemon, fapolicyd, is a service that can be used to help protect a system by limiting which applications have permission to run. The service can be used to complement other security related services, including SELinux. Unlike SELinux, which isn't concerned with how files and applications are installed onto the system and whether they're trusted, fapolicyd implements policy decisions based on whether applications are trusted and how they were installed onto the system.

Fapolicyd uses the `fanotify` kernel API to monitor file system events. When a file is accessed, fapolicyd decides whether the event can continue by checking the file against a trust database and evaluating a set of policy rules. If the file isn't in the trust database and a policy rule denies the action, the event is blocked and an EACCESS 'Permission denied' error is returned to the user.

Fapolicyd automatically adds files that are installed by DNF to the trust database, by using a DNF plugin. This approach helps to make fapolicyd more efficient when evaluating files that have been installed legitimately onto the system. Files can be evaluated based on their SHA-256 hash so that they can't be modified after they're added to the trust database. You can optionally add files to the trust database for files that aren't installed by DNF. You must reload the database after files are added manually or when files are installed by using the `rpm` command outside of the DNF framework.

A cache is used by Fapolicyd to help improve performance and to reduce the amount of time spent processing rules and performing database lookups for frequent events.

Fapolicyd rules define logging options that can be used to audit events. The default policy uses the audit log which can be viewed by using the `ausearch` command. You can change policy rules to log to the system log or to both the audit log and the system log to help with debugging.

For more information about the Fapolicyd see https://github.com/linux-application-whitelisting/fapolicyd.

About the Trust Database

Fapolicyd uses a trust database to efficiently lookup files that are trusted by the system. If a file isn't in the trust database, fapolicyd falls back to processing policy rules to decide whether an event is allowed or not.

The trust database is partitioned into information that fapolicyd gathers about installed RPM packages and information added manually by using configuration files.

The trust database is automatically populated with information about files that are installed onto the system, by copying information from the RPM database. When fapolicyd is started or an update is run, the trust database is updated for all files listed in the RPM database. The trust database is also updated automatically when new packages are installed or packages are updated by using DNF. The tracking of package data is handled by the `fapolicyd-dnf-plugin`, which notifies the fapolicyd daemon about DNF updates or installations.

If you install packages directly by using the `rpm` command, then the fapolicyd RPM database isn't updated and you might experience system freezes. In this case, it can be useful to refresh the trust database. See Refreshing the Trust Database for more information.

Administrators can also add files to the trust database manually when files are installed or added to the system outside of the usual DNF and RPM packaging mechanisms, such as compiling binaries from source, or when using the Python `pip` utility, Ruby's `gem` utility, Node.js's `npm` utility, or Perl's CPAN tooling.

The trust database distinguishes between these two types of trust: files that are trusted because they belong to the RPM database, and files that are trusted because they have been added manually to the trust database. For example, entries that are trusted from the RPM database are of the type `rpmdb`:

```
rpmdb /usr/bin/dnf-3 2092 
0a53d05260ba7ed4573...7ec64816e3ad49a2078c84836aeb7833e
```

Files that are added manually to the trust database are of the type `filedb`, for example:

```
filedb /home/user/demo.bin 140468 
e38cd120c925...46c9cd1aa83e44e697f0f3393d98b305
```

The database also stores the path to the file, the size of the file in bytes and a SHA-256 hash for the file. The file size and SHA-256 hash can be used to perform further integrity checks on files to make the system much more restrictive and robust against malicious activity. Although enabling integrity checking can protect against somebody working around fapolicyd by changing a file at a particular location, we don't recommend configuring fapolicyd for integrity checking because it increases the risk of system deadlock. Integrity checks are disabled in fapolicyd by default. See Checking for Trust Mismatches and Viewing the Content in the Trust Database for more information.

Note that processing of trusted files in the trust database is cached and is much quicker than the processing of individual policy rules. Therefore, if you need fapolicyd to trust particular applications or files that are made available to the system outside of DNF, it's more efficient to add them to the trust file database than to define rules for them. Only add policy rules for individual files if you need custom rules around user or group permissions.

About Policy Rules

Policy rules control how fapolicyd handles files that aren't explicitly listed within the trust database. Policy rules can also be used to further restrict behavior on the system for files that are in the trust database. You can use policy rules to make fapolicyd more or less restrictive. You can define policy rules to create exceptions that allow or deny certain applications for specified scenarios. For example, you can create rules to explicitly allow an application for a particular user or group, but to deny the application for all other users.

The `fapolicyd` package ships with two policy rule sets:

- The **known-libs** policy

The known-libs policy is the default rule set and is designed to protect the system by only allowing known applications or libraries to run. The policy is slightly more permissive because it allows Elf binaries, python programs, and shell scripts to run for trusted applications and libraries.

- The **restrictive** policy
The restrictive policy provides mostly the same rules as the known-libs policy, but includes several more restrictive rules that prevent the running of any application or library that's not within the trust database. This policy blocks any possibility of running any executable file that isn't trusted.

The rules for these policies are shipped in `/usr/share/fapolicyd/sample-rules/`. The rules that apply to the known-libs policy are copied into `/etc/fapolicyd/rules.d/` when the package is installed to make them active. The files that apply to each policy are described in `/usr/share/fapolicyd/sample-rules/README-rules`.

Fapolicyd processes rules based on their natural sort order, so rule files are named according to the following prefix convention:

- `10-`: Definitions of macros that can be used in other rules.
- `20-`: Rules to work around fapolicyd for system critical activity, such as when dracut builds kernel initramfs files or when DNF needs to run particular tools for updates.
- `30-`: Rules that identify access patterns that show how a program might be started. A default rule checks for programs that are started by the runtime linker (ld.so).
- `40-`: Rules for ELF binary files, such as rules to prevent malformed ELF files from running even if trusted, trust rules for ELF libraries and for trusted ELF binary executable files.
- `50-`: Rules that set out which users or groups are trusted to run particular programs or access particular files.
- `60-`: Rules for access to particular applications, where the application isn't in the trust database.
- `70-`: Rules related to different programming languages or scripting languages, such as Python, Perl, PHP, Ruby, or Lua.
- `80-`: Rules for trusted applications that might need advanced access controls.
- `90-`: General catch-all allow and deny rules.

Rules are compiled into a single file in `/etc/fapolicyd/compiled.rules` that fapolicyd reads at runtime.

Rule structure is described in detail in the `fapolicyd.rules(5)` manual page.

2 Installing and Running `fapolicyd`

Use the `dnf` command to install fapolicyd from the Oracle Linux 8 AppStream repository.

1. Verify that the `ol8_appstream` repository is enabled.
2. Use `dnf` to install the package.

   ```
   sudo dnf install fapolicyd
   ```

3. Enable and start the fapolicyd service.

   ```
   sudo systemctl enable --now fapolicyd
   ```

Changing Runtime Configuration

Runtime configuration options are set in `/etc/fapolicyd/fapolicyd.conf`. Options are described in detail in the `fapolicyd.conf(6)` manual page. For more information, see also https://github.com/linux-application-whitelisting/fapolicyd/blob/main/README.md.

Configuring Runtime Statistics Reporting

Fapolicyd can generate a runtime statistics report that provides information about accesses, denials, and cache performance. The runtime statistics report is generated when fapolicyd is stopped and can be useful for gathering information about performance. You can use the information in the statistics report to decide whether to change other configuration options or to perform debug work. The report is written to `/var/log/fapolicyd-access.log` and default output is similar to the following:

```
Permissive: false
q_size: 640
Inter-thread max queue depth 6
Allowed accesses: 668513
Denied accesses: 0
Trust database max pages: 25600
Trust database pages in use: 7567 (29%)

File access attempts from oldest to newest as of Wed Nov 15 16:46:59 2023

        FILE                                                  ATTEMPTS
---------------------------------------------------------------------------
/var/tmp/dracut.kZVhRg/initramfs/usr/lib/kbd/keymaps/xkb/tr-alt.map.gz  1
/var/tmp/dracut.kZVhRg/initramfs/usr/lib/kbd/unimaps/koi8u.uni  1
/var/tmp/dracut.kZVhRg/initramfs/usr/bin/stgTP4DF               1
...
/usr/bin/mandb (?)                                              1
/usr/bin/mandb (?)                                              264
---
```

```
Subject cache size: 1549
Subject slots in use: 1549 (100%)
Subject hits: 666964
Subject misses: 46044
Subject evictions: 44495 (6%)
```

Configuration options that can be set to control this report include:

- **do_stat_report**: Controls whether the statistics report is generated. By default the value is set to 1 to indicate that the option is enabled.
- **detailed_report** Controls whether fapolicyd adds subject and object information to the usage statistics report, indicating the number of times particular subject-object events occur. This content can be useful when debugging but can be disabled to reduce the size of the log. The default value for this option is set to 1 to indicate that the option is enabled.

Logging Controls

Audit logging is mostly handled using fanotify events in the audit log, but you can also configure the policy to log to the system log. See Changing Default Policy Logging, Debugging and Permissive Mode and Auditing Denial Events.

When logging to the system log or running fapolicyd in debugging mode, you can control the content of the log output from access decisions by configuring the **syslog_format** option. The format is a comma separated list of the different values to indicate rule information with subject and object information delineated by use of a colon character. Note that system performance is affected by the amount of content that you log. The default value is:
`rule,dec,perm,auid,pid,exe,:,path,ftype,trust`.

Values available for the **syslog_format** option include:

- `rule`: The rule number from the compiled policy rules. See Listing Policy Rules.
- `dec`: The decision that fapolicyd takes for the rule.
- `perm`: The permission that's applied in the rule.
- Any of the subject options.
- `:`: The separator to delineate between subject and object options.
- Any of the object options.

For more information about subject and object options, see Creating Policy Rules.

Performance Controls

Performance control options can help improve memory usage and processing.

The following options are available to control the size of the caches that are used by fapolicyd to improve performance. For both options, aim to keep the allocated memory as small as feasibly possible, but ensure that enough memory is allocated to the cache to maximize the hits to evictions ratio, that can be calculated from the statistics report. To avoid cache churn resulting from collisions, consider setting that cache size values to prime numbers.

- **subj_cache_size**: Controls how many entries the subject cache holds. The default value is 1549.
- **obj_cache_size**: Controls how many entries the object cache holds. The default value is 8191.

Debugging and Permissive Mode

By default, fapolicyd is configured in enforcing mode. Permissive mode allows events that fapolicyd might have blocked to run but continues to audit the event. Permissive mode, when used with an appropriate logging configuration enabled, can be helpful when debugging. Don't leave permissive mode enabled on a system that you want to protect by using fapolicyd.

You can enable permissive mode by editing `/etc/fapolicyd/fapolicyd.conf` and setting the `permissive` configuration option to 1. You must restart the fapolicyd service for the change to take effect.

You can optionally run fapolicyd as an active process in permissive mode with debugging enabled to get more information about events as they take place.

To run fapolicyd in permissive mode with debugging enabled:

1. Stop the fapolicyd service.

   ```
   sudo systemctl stop fapolicyd
   ```

2. Run fapolicyd from the command line with the `--permissive` and `--debug` options enabled.

   ```
   sudo fapolicyd --permissive --debug
   ```

   Output similar to the following is displayed.

   ```
   Loading rule file:
   ## This file is automatically generated from /etc/fapolicyd/rules.d
   %languages=application/x-bytecode.ocaml,application/x-
   bytecode.python,application/java-archive,text/x-java,application/x-java-
   applet,application/javascript,text/javascript,text/x-awk,text/x-
   gawk,text/x-lisp,application/x-elc,text/x-lua,text/x-m4,text/x-
   nftables,text/x-perl,text/x-php,text/x-python,text/x-R,text/x-ruby,text/x-
   script.guile,text/x-tcl,text/x-luatex,text/x-systemtap
   allow perm=any uid=0 : dir=/var/tmp/
   allow perm=any uid=0 trust=1 : all
   allow perm=open exe=/usr/bin/rpm : all
   allow perm=open exe=/usr/bin/python3.9 comm=dnf : all
   deny_audit perm=any pattern=ld_so : all
   deny_audit perm=any all : ftype=application/x-bad-elf
   allow perm=open all : ftype=application/x-sharedlib trust=1
   deny_audit perm=open all : ftype=application/x-sharedlib
   allow perm=execute all : trust=1
   allow perm=open all : ftype=%languages trust=1
   deny_audit perm=any all : ftype=%languages
   allow perm=any all : ftype=text/x-shellscript
   deny_audit perm=execute all : all
   ```

```
allow perm=open all : all
Loaded 14 rules
Changed to uid 980
Initializing the trust database
fapolicyd integrity is 0
backend rpmdb registered
backend file registered
Loading rpmdb backend
Loading file backend
Checking if the trust database up to date
Importing trust data from rpmdb backend
Importing trust data from file backend
Entries in trust DB: 86032
Loaded trust info from all backends(without duplicates): 86032
Trust database checks OK
added /dev/shm mount point
added / mount point
added /var/oled mount point
added /boot mount point
added /boot/efi mount point
added /run/user/982 mount point
added /run/user/1000 mount point
Starting to listen for events
```

Events that would be denied are tagged with `dec=deny_audit`.

Tip:

Debug output can be verbose. Use the `--debug-deny` option instead of the standard `--debug` option to only output denial events. You don't need to run in debug mode to audit denial events, though. See Auditing Denial Events for more information. If you need to get more information about denial events, you can change rules to log to the system log. See Changing Default Policy Logging for more information.

3. To exit the running daemon, use the Ctrl-C key combination to send a SIGINT to interrupt the process.

Auditing Denial Events

Use the `ausearch` command to view denied events in the audit log, as these are tracked by using fanotify messages.

You need at least one rule defined for auditd to start logging fapolicyd events. If you don't have any rules defined, no events appear in the audit log. You can create any rule for auditing to start working. For example, you can create a rule to audit changes to configuration in `/etc/fapolicyd` as follows:

```
sudo tee /etc/audit/rules.d/40-fapolicyd.rules > /dev/null <<'EOF'
# This policy monitors /etc/fapolicyd/ for changes to configuration
# This rule is generated to ensure that events are logged to the audit
log for fapolicyd tracking
```

```
-w /etc/fapolicyd/ -p wa -k fapolicyd_changes
EOF
```

You need to restart the auditd service or reboot the system for this change to take effect. To restart the auditd service run:

```
sudo service auditd restart
```

Note:

Auditd can't be restarted by using the `systemctl` command.

Denial events are logged to the audit log and can be reviewed by using the `ausearch` command. For example:

```
sudo ausearch --start today -m fanotify
```

Use `aureport` to create easier to read outputs. For example:

```
sudo ausearch --start today -m fanotify --raw | aureport --file
```

3

Managing File Trusts

File trusts are stored in the trust database. Trusts can either be generated based on information in the RPM database or can be manually defined by adding configuration entries on the file system. The contents of the trust database and how trusts work is discussed in more detail in About the Trust Database.

Refreshing the Trust Database

Refresh the trust database if files on the system have been added or updated outside of the DNF framework.

To refresh the fapolicyd trust database manually, run:

```
sudo fapolicyd-cli --update
```

Adding Files to the Trust File Database

You can add any files that aren't installed by using DNF to the file database manually.

To add a file to the trust file database, run:

```
sudo fapolicyd-cli --file add <path_to_file> --trust-file trust_entry
```

If the file isn't already in a trust database, the command adds the file to the trust file configuration by creating an entry at `/etc/fapolicyd/trust.d/<trust_entry>`.

For example, to add `/home/user/demo.bin` to `/etc/fapolicyd/trust.d/demo`, run:

```
sudo fapolicyd-cli --file add /home/user/demo.bin --trust-file demo
```

Tip:

You can use command line tools such as `find` to add many entries to the trust file database at the same time. For example:

```
find /home/user/bin/ -type f -exec fapolicyd-cli --file add {} --
trust-file trusted_user_bin \;
```

All entries in the trust file database are stored as plain text files in `/etc/fapolicyd/trust.d/` and can be edited with a text editor, if required. If you need to update file sizes or hash values, see Updating the Trust File Database.

To remove a file from the trust file database, either edit the text file directly to remove the entry, or run:

```
sudo fapolicyd-cli --file delete <path_to_file>
```

Important:

After you make any changes to the trust file database you must refresh the trust databases for fapolicyd to note any changes. See Refreshing the Trust Database.

Updating the Trust File Database

To update the trust file database for changes to file sizes or hashes on all files in the file trust database, run:

```
fapolicyd-cli -f update
```

If you specify the path to the file, only the values for that file are updated in the database.

After you make any changes to the trust file database you must refresh the trust databases for fapolicyd to note any changes. See Refreshing the Trust Database.

Checking for Trust Mismatches

Trust mismatches occur when the file size or SHA-256 hash value for a file on the file system no longer matches the information stored for the file in the trust database. Changing a file outside of using DNF can cause a trust mismatch. For example, if a file is installed or updated by using the `rpm` command directly or when a user or process has changed the file.

Note:

While you can configure fapolicyd for file integrity checks based on size or on the SHA-256 hash, we don't recommend applying this option globally as it increases the likelihood of system deadlock.

To check for trust mismatches on a system, run:

```
sudo fapolicyd-cli --check-trustdb
```

Output typically describes the files that mismatch and what the mismatch is. For example:

```
/etc/selinux/targeted/contexts/files/file_contexts miscompares: size
sha256
```

```
/etc/selinux/targeted/policy/policy.33 miscompares: size sha256
/opt/rh/gcc-toolset-12/root/usr/bin/ld miscompares: size sha256
/usr/lib64/gconv/gconv-modules.cache miscompares: size sha256
...
```

Note that mismatches are expected because the size or content of some files change from the values in the RPM database after certain commands or services are run. Nonetheless, checking for mismatches can help alert you to files that might be in the trust database but which have changed after they were added to the database.

Viewing the Content in the Trust Database

You can view all the information in the trust database by dumping the data. To dump the data in the trust database, run:

```
sudo fapolicyd-cli -D
```

Output is listed to display the *type of trust*, the *path to the file* that's trusted, the *size of the file* in bytes and the *SHA-256 hash* for the file.

Tip:

You can use command line tools such as `grep` to limit the data returned in the dump output. For example:

```
sudo fapolicyd-cli -D|grep '/usr/bin/dnf-3'
```

Resetting the Trust Database

You can reset the trust database by stopping fapolicyd by deleting the database when debugging issues in fapolicyd.

To reset the trust database, run:

```
sudo systemctl stop fapolicyd
sudo fapolicyd-cli --delete-db
```

The trust database are removed entirely and then created and updated when you next start the fapolicyd service.

Don't ever remove the `/var/lib/fapolicyd/` directory directly as this could prevent fapolicyd from functioning correctly and can cause system lockout.

4

Managing Policies

For more information about policy rules and how they work, see About Policy Rules.

Listing Policy Rules

To list policy rules that are in use by fapolicyd, run:

```
sudo fapolicyd-cli -l
```

Output from this command might be similar to the following:

```
-> %languages=application/x-bytecode.ocaml,application/x-
bytecode.python,application/java-archive,text/x-java,application/x-java-
applet,application/javascript,text/javascript,text/x-awk,text/x-gawk,text/x-
lisp,application/x-elc,text/x-lua,text/x-m4,text/x-nftables,text/x-
perl,text/x-php,text/x-python,text/x-R,text/x-ruby,text/x-
script.guile,text/x-tcl,text/x-luatex,text/x-systemtap
1. allow perm=any uid=0 : dir=/var/tmp/
2. allow perm=any uid=0 trust=1 : all
3. allow perm=open exe=/usr/bin/rpm : all
4. allow perm=open exe=/usr/bin/python3.9 comm=dnf : all
5. deny_audit perm=any pattern=ld_so : all
6. deny_audit perm=any all : ftype=application/x-bad-elf
7. allow perm=open all : ftype=application/x-sharedlib trust=1
8. deny_audit perm=open all : ftype=application/x-sharedlib
9. allow perm=execute all : trust=1
10. allow perm=open all : ftype=%languages trust=1
11. deny_audit perm=any all : ftype=%languages
12. allow perm=any all : ftype=text/x-shellscript
13. deny_audit perm=execute all : all
14. allow perm=open all : all
```

Note that the rules are numbered. When fapolicyd runs in debug mode, the output displays the rule number that's enforced for an event. You can use this information to decide whether you need to insert a new policy rule to change the existing policy and to decide where that rule might need to be in the policy hierarchy. See Debugging and Permissive Mode for information on debugging.

You can compare this list to the compiled rules to check whether you need to reload the rules into fapolicyd.

```
sudo cat /etc/fapolicyd/compiled.rules
```

See Checking and Loading Policy Rules.

Creating Policy Rules

You might need to create custom policy rules if the default policy is too restrictive and is preventing applications from running. You can also create custom policy rules to further restrict access to applications and files. Rules can be added to the `/etc/fapolicyd/rules.d` directory following the conventions described in About Policy Rules and in the `fapolicyd.rules(5)` manual page.

Rules have the following general format:

```
decision perm subject : object
```

- **decision**: whether to allow or deny an event and whether to log that action. Values can be `allow`, `deny`, `allow_audit`, `deny_audit`, `allow_syslog`, `deny_syslog`, `allow_log`, or `deny_log`.
- **perm**: the type of permission applied to the file, such as whether to trigger when the object is opened, run, or for any activity on the object. Values can be `open`, `execute`, or `any`. If no permission is specified, the default value is `open`.
- **subject**: the actor performing the action on the object that the permission and decision applies to. This value can be set to `all` for every actor, but can also be limited to a particular UID or GID or another executable file. Values can be `all`, `auid`, `uid`, `gid`, `sessionid`, `pid,ppid`, `trust`, `comm`, `exe`, `dir`, `device`, `pattern`. See the `fapolicyd.rules(5)` manual page for more detail.
- **object**: the files or applications that the decision applies to. This value can be specified in various ways, such as by providing the path to a particular file, the MIME type of the file or for file matches in the trust database. Values can be `all`, `path`, `dir`, `device`, `ftype`, `trust`, `sha256hash`. See the `fapolicyd.rules(5)` manual page for more detail.

You can specify several subject and object directives for a single rule. For example, you can match the file path, MIME type, and file size as the object for a rule.

Rules can also be created to generate macros, or sets, that consist of a key and a set of values in the form of integers or strings in a comma-separated list. Set rules are prefixed with the percentage (%) symbol, and the set can be referenced when creating a rule by specifying the key prefixed with the percentage (%) symbol. The format of a set rule follows:

```
%key=value1,value2
```

You can see an example of a set rule in `/etc/fapolicyd/rules.d/10-languages.rules`.

The following example rules can be used to help guide you when creating custom rules for an environment.

Example 4-1 Create a rule to allow a group of trusted users to run files matching the defined languages in their home directories

1. Create a system group and add trusted users to that group, or select an existing group of trusted users. In this example, we use the `adm` group for system administrators.

2. Create a policy rule file at `/etc/fapolicyd/rules.d/50-adm-home-trust.rules` and add the following content:

   ```
   allow perm=any gid=adm : ftype=%languages dir=/home
   deny_log perm=any all : ftype=%languages dir=/home
   ```

   Two rules are defined:

   - Grant all users in the group `adm` all permissions on any files, in the `/home` directory, of the type described in the `languages` set in `/etc/fapolicyd/rules.d/10-languages.rules`.
   - Deny all users any permissions on any files, in the `/home` directory, of the type described in the `languages` macro. The `deny_log` rule causes the event to be added to both the audit log and to the system log.

3. Compile and load the new rules.

   ```
   sudo fagenrules --load
   ```

   See Checking and Loading Policy Rules for more information.

4. Verify that the rule is in effect.

   a. As a user in the `adm` group, create a test executable file in `$HOME/test_fapolicyd.py` with the following content:

   ```
   #/usr/bin/python3
   print("Test succeeded")
   ```

   b. Set the file mode to executable:

   ```
   chmod +x $HOME/test_fapolicyd.py
   ```

   c. Run the file:

   ```
   $HOME/test_fapolicyd.py
   ```

   The following output is displayed:

   ```
   Test succeeded
   ```

   d. As a user that isn't in the `adm` group, repeat the same steps. When you run the test script, the following output is displayed:

   ```
   bash: ./test_fapolicyd.py: Operation not permitted
   ```

   If you have auditing configured, you can check for entries in the audit log as follows:

   ```
   sudo ausearch --start today -m fanotify
   ```

Output similar to the following is displayed:

```
...
time->Fri Jan  5 12:05:39 2024
type=PROCTITLE msg=audit(1704456339.181:406): proctitle="bash"
type=PATH msg=audit(1704456339.181:406): item=0 name="./
test_fapolicyd.py" inode=68153551
  dev=fc:00 mode=0100755 ouid=0 ogid=0 rdev=00:00
obj=unconfined_u:object_r:user_home_t:s0
  nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0
cap_frootid=0
type=CWD msg=audit(1704456339.181:406): cwd="/home/guest"
type=SYSCALL msg=audit(1704456339.181:406): arch=c000003e
syscall=257 success=no exit=-1
  a0=ffffff9c a1=555d4b74e050 a2=0 a3=0 items=1 ppid=46725
pid=46765 auid=1000 uid=1001
  gid=1001 euid=1001 suid=1001 fsuid=1001 egid=1001 sgid=1001
fsgid=1001 tty=pts0 ses=3
  comm="bash" exe="/usr/bin/bash"
  subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
key=(null)
type=FANOTIFY msg=audit(1704456339.181:406): resp=2
```

Example 4-2 Create a rule to allow users to run a trusted application

Note that creating rules to allow particular applications to run is less efficient than adding the application to the trust database. See Adding Files to the Trust File Database for more information. Only add rules for specific files if you need to provide more explicit controls over file accesses. In this example, a rule is created controlling the ability to run a particular binary application that can only be run by using the trusted bash shell binary.

1. Create a policy rule file at `/etc/fapolicyd/rules.d/80-trustapp.rules` and add the following content:

   ```
   allow_log perm=execute exe=/usr/bin/bash trust=1 : path=/opt/
   external/app.bin ftype=application/x-executable trust=0
   ```

   The rule states that in the case that the trusted bash shell tries to run the untrusted application at the path `/opt/external/app.bin`, where that file is of MIME type `application/x-executable`, the action is allowed. Note that the decision is set to `allow_log` so that the action is logged for when we verify that the rule is working. After you have verified the rule, you can change the decision to `allow`.

   Tip:

   To check the mime type of a file when writing rules, run the following command: `fapolicyd-cli --ftype` *`/path-to-file`*

   Note that by specifying the MIME type for the file and the way in which the file can be loaded, the rule is reasonably restrictive. You can optionally create a more restrictive rule based on the SHA-256 hash of the file, which would prevent the rule from applying if the file is modified:

- Obtain the SHA-256 hash for the file.

  ```
  sha256sum /opt/external/app.bin|awk '{print $1}'
  ```

- Use the SHA-256 hash to rewrite the rule as follows:

  ```
  allow perm=execute exe=/usr/bin/bash trust=1 :
  sha256hash=340e...74f3cdd3babb
  ```

  Set the value of `sha256hash` to the value that you obtained from running the previous command.

2. Compile and load the new rules.

   ```
   sudo fagenrules --load
   ```

   See Checking and Loading Policy Rules for more information.

3. Verify that the rule is in effect.
   When you try to run `/opt/external/app.bin` as a standard user from the Bash shell, it runs correctly. Because the decision in the rule is to `allow_log`, you can view the fapolicyd log to see the rule in action. For example, run:

   ```
   sudo journalctl -S today -u fapolicyd|tail
   ```

   The output displays that the rule is being enforced, for example:

   ```
   Jan 05 14:22:36 ro-ansible-ol8 fapolicyd[45792]: rule=13 dec=allow_log
     perm=execute auid=1000 pid=50439 exe=/usr/bin/bash : path=/opt/app.bin
     ftype=application/x-executable trust=0
   ```

Checking and Loading Policy Rules

If rules have been added or changed, you must check for consistency with the compiled rule set that fapolicyd uses and load them into fapolicyd.

1. Check for inconsistencies between the rules in `/etc/fapolicyd/rules.d` and the compiled rules in `/etc/fapolicyd/compiled.rules`.

   ```
   sudo fagenrules --check
   ```

   If the rules in `/etc/fapolicyd/rules.d` have been updated and need to be recompiled and loaded, the output appears as follows:

   ```
   /sbin/fagenrules: Rules have changed and should be updated
   ```

2. To compile the rules in `/etc/fapolicyd/rules.d` and load them into fapolicyd, run:

   ```
   sudo fagenrules --load
   ```

   You don't need to restart fapolicyd for the changes to take effect.

3. List the rules to review that the changes have loaded correctly into fapolicyd.

   ```
   sudo fapolicyd-cli  -l
   ```

Changing Default Policy Logging

Default policy rules shipped with fapolicyd are configured to only log to the audit log for denials. This configuration is appropriate for production systems but the information stored in these logs might be limited for debugging purposes. If you need to track which rules are being used to make the final decision on an event, you can either run fapolicyd in debugging mode, or you could change rules to output information to the system log. For more information on running fapolicyd in debugging mode, see Debugging and Permissive Mode.

Policy rule decisions identify whether to log information and how that information must be logged. By default, for denial decisions, the rules that are included in the fapolicyd package use the `deny_audit` decision type. You can change all `deny_audit` decision type to `deny_log` to log information to both the audit log and to the system log.

To enhance logging:

1. Update all rules in `/etc/fapolicyd/rules.d` to replace `deny_audit` with `deny_log`.

   ```
   sudo bash -c 'for i in /etc/fapolicyd/rules.d/*; do sed -i "s/
   deny_audit/deny_log/g" $i; done'
   ```

   You can also enable logging for any allow rules, but doing so can result in verbose output and can have significant performance impact.
2. Check that the rules are updated.

   ```
   sudo fagenrules --check
   ```

3. Load the new rules.

   ```
   sudo fagenrules --load
   ```

4. Review the rules to ensure that the change has loaded correctly into fapolicyd.

   ```
   sudo fapolicyd-cli  -l | grep 'deny_log'
   ```

When an event is denied by fapolicyd and the decision is set to `deny_log`, an entry appears in the system log, similar to the following:

```
fapolicyd[1478]: rule=13 dec=deny_log perm=execute auid=1000 pid=5361
exe=/usr/bin/bash :
   path=/home/user/demo.bin ftype=application/x-executable trust=0
```

Note that the output includes the rule number for the rule that fapolicyd used to make the final decision to deny the event.

To view system log entries for fapolicyd, run:

```
sudo journalctl -S today -u fapolicyd
```