Rocky Enterprise Linux 9.2 Manual Pages on command 'audit2why.1'

$ man audit2why.1

AUDIT2ALLOW(1) NSA AUDIT2ALLOW(1)

NAME

audit2allow - generate SELinux policy allow/dontaudit rules from logs of denied operations

audit2why - translates SELinux audit messages into a description of why the access was denied (audit2allow -w)

SYNOPSIS

audit2allow [options]

OPTIONS

-a | --all

Read input from audit and message log, conflicts with -i

-b | --boot

Read input from audit messages since last boot conflicts with -i

-d | --dmesg

Read input from output of /bin/dmesg. Note that all audit mes? sages are not available via dmesg when auditd is running; use "ausearch -m avc | audit2allow" or "-a" instead.

-D | --dontaudit

Generate dontaudit rules (Default: allow)

-h | --help

Print a short usage message

-i <inputfile> | --input <inputfile>

read input from <inputfile>

-l | --lastreload

read input only after last policy reload

-m <modulename> | --module <modulename>

Generate module/require output <modulename>

-M <modulename>

Generate loadable module package, conflicts with -o

-p <policyfile> | --policy <policyfile>

Policy file to use for analysis

-o <outputfile> | --output <outputfile>

append output to <outputfile>

-r | --requires

Generate require output syntax for loadable modules.

-N | --noreference

Do not generate reference policy, traditional style allow rules.

This is the default behavior.

-R | --reference

Generate reference policy using installed macros. This attempts

to match denials against interfaces and may be inaccurate.

-x | --xperms

Generate extended permission access vector rules

-w | --why

Translates SELinux audit messages into a description of why the

access was denied

-v | --verbose

Turn on verbose output

DESCRIPTION

This utility scans the logs for messages logged when the system denied

permission for operations, and generates a snippet of policy rules

which, if loaded into policy, might have allowed those operations to succeed. However, this utility only generates Type Enforcement (TE) al? low rules. Certain permission denials may require other kinds of pol? icy changes, e.g. adding an attribute to a type declaration to satisfy an existing constraint, adding a role allow rule, or modifying a con? straint. The audit2why(8) utility may be used to diagnose the reason when it is unclear.

Care must be exercised while acting on the output of this utility to ensure that the operations being permitted do not pose a security threat. Often it is better to define new domains and/or types, or make other structural changes to narrowly allow an optimal set of operations to succeed, as opposed to blindly implementing the sometimes broad changes recommended by this utility. Certain permission denials are not fatal to the application, in which case it may be preferable to simply suppress logging of the denial via a 'dontaudit' rule rather than an 'allow' rule.

EXAMPLE

NOTE: These examples are for systems using the audit package. If you do not use the audit package, the AVC messages will be in /var/log/messages. Please substitute /var/log/messages for /var/log/audit/audit.log in the examples.

Using audit2allow to generate module policy

```
$ cat /var/log/audit/audit.log | audit2allow -m local > local.te
$ cat local.te
module local 1.0;
require {
        class file {  getattr open read };
        type myapp_t;
        type etc_t;
 };
allow myapp_t etc_t:file { getattr open read };
<review local.te and customize as desired>
```

Using audit2allow to generate module policy using reference policy

```
$ cat /var/log/audit/audit.log | audit2allow -R -m local > local.te
$ cat local.te
policy_module(local, 1.0)
gen_require(`
        type myapp_t;
        type etc_t;
')
files_read_etc_files(myapp_t)
<review local.te and customize as desired>
```

Building module policy using Makefile

```
# SELinux provides a policy devel environment under
# /usr/share/selinux/devel including all of the shipped
# interface files.
# You can create a te file and compile it by executing
$ make -f /usr/share/selinux/devel/Makefile local.pp
# This make command will compile a local.te file in the current
# directory. If you did not specify a "pp" file, the make file
# will compile all "te" files in the current directory.  After
# you compile your te file into a "pp" file, you need to install
# it using the semodule command.
$ semodule -i local.pp
```

Building module policy manually

```
# Compile the module
$ checkmodule -M -m -o local.mod local.te
# Create the package
$ semodule_package -o local.pp -m local.mod
# Load the module into the kernel
$ semodule -i local.pp
```

Using audit2allow to generate and build module policy

```
$ cat /var/log/audit/audit.log | audit2allow -M local
Generating type enforcement file: local.te
Compiling policy: checkmodule -M -m -o local.mod local.te
Building package: semodule_package -o local.pp -m local.mod
```

********** IMPORTANT **********

In order to load this newly created policy package into the kernel,

you are required to execute

semodule -i local.pp

Using audit2allow to generate monolithic (non-module) policy

$ cd /etc/selinux/$SELINUXTYPE/src/policy

$ cat /var/log/audit/audit.log | audit2allow >> domains/misc/local.te

$ cat domains/misc/local.te

allow cupsd_config_t unconfined_t:fifo_file { getattr ioctl };

<review domains/misc/local.te and customize as desired>

$ make load

AUTHOR

This manual page was written by Manoj Srivastava <srivasta@debian.org>,

for the Debian GNU/Linux system. It was updated by Dan Walsh <dwalsh@redhat.com>

The audit2allow utility has contributions from several people, includ?

ing Justin R. Smith and Yuichi Nakamura. and Dan Walsh