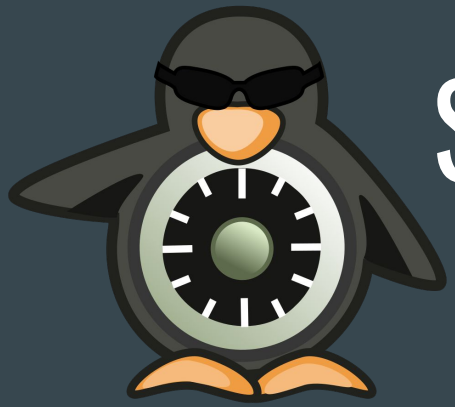

SELinux Policy Development

•••

Jason Zaman

FOSSASIA 2018 March 24

blog.perfinion.com

Overview

1. Whoami
2. What is SELinux?
3. Parts of an SELinux Policy
4. Policy Modules
5. Reference Policy
    a. Perms
    b. Patterns
    c. Filetrans
6. Module Sources
7. Interfaces
8. Booleans
9. Let’s write a policy

Who Am I?

Jason “perfinion” Zaman

Gentoo Linux Developer - SELinux and Hardened Gentoo projects

Upstream SELinux maintainer

jason@perfinion.com GPG keyid: `0x7EF137EC935B0EAF`

Blog: http://blog.perfinion.com/ Twitter: @perfinion Github: github.com/perfinion

What is SELinux?

Security **E**nhanced **Linux**

SELinux ***controls access*** between applications and resources.

By using a ***mandatory*** security policy, SELinux enforces the security goals of the system regardless of whether applications misbehave or users act carelessly.

SELinux is capable of enforcing a wide range of security goals, from simply sandboxing applications to locking down network facing daemons and restricting users to only the resources they need to work.

Parts of an SELinux Policy

/etc/selinux/strict/	base dir for the policy
policy/policy.31	loaded into the kernel
contexts/*	context files for SELinux-aware programs
contexts/files/ file_contexts file_contexts.home_dirs file_contexts.local	All the fcontexts for the entire system

Policy Modules

SELinux policies are made from many separate modules that can be loaded independently

Each module has rules, fcontexts, interfaces.

Rules + fcontexts are in the module

Interfaces are in /usr/share/selinux/strict/include/

Before loading, the policy modules are linked together in /var/lib/selinux/strict/active/modules/

```
# semodule --list=full
500 mycustom                    pp
400 base                        pp
400 chromium                    pp
400 cron                        pp
400 cups                        pp
400 gnome                       pp
400 gpg                         pp
400 java                        pp
```

Reference Policy

The base policy that all distros base their policies on.

Written with lots of interfaces to make things easier than writing raw policy.

Interfaces in m4

make -f /usr/share/selinux/strict/include/Makefile

Split into two main chunks: defines and interfaces.

Refpolicy - perms

SELinux has a lot of perms that are needed together.

eg: Reading a file requires:

```
define(`read_inherited_file_perms',`{ getattr read lock ioctl }')

define(`read_file_perms',`{ read_inherited_file_perms open }')

define(`manage_dir_perms',`{ create open getattr setattr read write link unlink rename search add_name remove_name reparent rmdir lock ioctl }')
```

They always follow a consistent naming scheme: `<perm>_<class>_perms`

Perm: `getattr, setattr, read, append, write, rw, create, delete, manage`
Class: `file, dir, lnk_file, sock_file, blk_file, chr_file`

`read_file_perms, create_lnk_file_perms, rw_sock_file_perms, manage_dir_perms`

Look in policy/support/obj_perm_sets.spt

Refpolicy - patterns

Frequently need to do several things together.

Creating and writing to a file requires searching the dir the file will be in, and creating the file itself.

```
# Parameters:
# 1. domain type
# 2. container (directory) type
# 3. file type
```

```
define(`create_files_pattern',`
        allow $1 $2:dir add_entry_dir_perms;
        allow $1 $3:file create_file_perms;
')

define(`write_files_pattern',`
        allow $1 $2:dir search_dir_perms;
        allow $1 $3:file write_file_perms;
')

write_files_pattern(app_t, var_log_t, app_log_t)

rw_sock_file_pattern(app_t, mysqld_var_run_t,
mysqld_var_run_t)
```

Refpolicy - filetrans

We have many domains and contexts on our system, this is how they get there.

Tells SELinux that if a domain creates an object inside a dir with a label, the created object should have a different label.

```
filetrans_pattern(httpd_t, var_log_t,
httpd_log_t, dir)
```

```
/var/log            var_log_t
/var/log/apache     httpd_log_t
```

```
# Parameters:
# 1. domain type
# 2. container (directory) type
# 3. new object type
# 4. object class(es)
# [optional] 5. filename (c style
strcmp ready)

define(`filetrans_pattern',`
        allow $1 $2:dir rw_dir_perms;
        type_transition $1 $2:$4 $3 $5;
')
```

Refpolicy Module Sources

Made up from 3 files each

- apache.te — type enforcement rules
- apache.fc — fcontexts for the module
- apache.if — interfaces

Some general rules:

- The types names should all start with the module
- Modules should only directly access its own types
- Accessing other types should be done through interfaces

```
make -f /usr/share/selinux/strict/include/Makefile

Compiling strict apache module
/usr/bin/checkmodule:  loading policy
configuration from tmp/apache.tmp
/usr/bin/checkmodule:  policy configuration loaded
/usr/bin/checkmodule:  writing binary
representation (version 19) to tmp/apache.mod
Creating strict apache.pp policy package

semodule -i apache.pp
```

Interfaces

M4 macros to help do complicated things and keep everything readable.

https://github.com/TresysTechnology/refpolicy/wiki/InterfaceNaming

```
modulename[_modifier]_verb_predicate()
```

```
apache_append_log(app_t)
apache_read_config(app_t)
apache_role(staff_r, staff_t)
gpg_domtrans(app_t)
logging_log_filetrans(app_t, app_log_t, file)
```

Finding Interfaces

https://github.com/sjvermeu/small.coding/blob/master/selinux-local/localfuncs

sefinddef sefindif seshowdef seshowif

```
$ sefindif "filetrans.*var_log"
system/logging.if: interface(`logging_log_filetrans',`
system/logging.if:    filetrans_pattern($1, var_log_t, $2, $3, $4)
$ seshowif logging_log_filetrans
interface(`logging_log_filetrans',`
     gen_require(`
          type var_log_t;
     ')

     files_search_var($1)
     filetrans_pattern($1, var_log_t, $2, $3, $4)
     allow $1 var_log_t:lnk_file read_lnk_file_perms;
')
```

Booleans

Groups of rules that can be enabled or disabled easily by the administrator

```
$ setsebool -P httpd_can_network_connect on
```

```
## <desc>
##   Determine whether httpd scripts and
##   modules can connect to the network using TCP
## </desc>
gen_tunable(httpd_can_network_connect, false)

tunable_policy(`httpd_can_network_connect',`
        corenet_sendrecv_all_client_packets(httpd_t)
        corenet_tcp_connect_all_ports(httpd_t)
        corenet_tcp_sendrecv_all_ports(httpd_t)
')
```

Let’s Write a Policy From Scratch

The VM has /usr/local/bin/server.py which needs to bind to port 8080 and logs the hits to /var/log/server/hits.log

server.py
setenforce 0
server.py
ausearch -m avc -ts recent
cd ~/selinux/; make && semodule -X500 -i *.pp
server.py

Learning More

Slides will be on my blog: http://blog.perfinion.com/

SELinux coloring book: http://blog.linuxgrrl.com/2014/04/16/the-selinux-coloring-book/

SELinux project wiki: https://selinuxproject.org/

Gentoo SELinux wiki: https://wiki.gentoo.org/wiki/SELinux

Fedora SELinux wiki: https://fedoraproject.org/wiki/SELinux

SELinux Notebook:
http://freecomputerbooks.com/books/The_SELinux_Notebook-4th_Edition.pdf