Debugging SEPolicy

Chris Simmonds, 2net

Debugging SELinux

- Some tips and tricks to help debug SEPolicy problems
  - sesearch
  - sepolicy-check
  - decoding inode numbers

Sample policy

Here is sepolicy for a simple http daemon:

`httpd.te`

```
type httpd, domain;
type httpd_exec, vendor_file_type, exec_type, file_type;
type httpd_data_file, file_type, vendor_file_type;
init_daemon_domain(httpd)

allow httpd httpd_data_file:dir search;
allow httpd httpd_data_file:file { getattr open read };
allow httpd node:tcp_socket node_bind;
allow httpd port:tcp_socket name_bind;
allow httpd self:capability net_bind_service;
allow httpd self:tcp_socket { accept bind create listen read setopt shutdown write };
```

`file_contexts` - note that the httpd is implemented by busybox

```
/vendor/bin/busybox     u:object_r:httpd_exec:s0
/vendor/etc/html(/.*)?  u:object_r:httpd_data_file:s0
```

Run-time

You can see the context of the busybox process

```
# ps -AZ | grep busybox
u:r:httpd:s0      root      452     1    2812   1348 0       0 S busybox
```

The context of the executable

```
# ls -Z /vendor/bin/busybox
u:object_r:httpd_exec:s0 /vendor/bin/busybox
```

The context of the html files

```
# ls -Z /vendor/etc/html
u:object_r:httpd_data_file:s0 index.html
```

1. sesearch

`sesearch` is the SELinux policy rule search tool

Usage (somewhat simplified)

```
sesearch RULE -s SOURCE -t TARGET -tc TCLASS -p PERMS

RULE = --allow --dontaudit --neverallow

-ds    A matching rule must have the specified source attribute/type/role explicitly,
       instead of matching by attribute contents.
-dt    Same for specified target
```

Installing

There is an old version of `sesearch` already in AOSP, but it does not work:

```
$ which sesearch
/home/chris/aosp/external/selinux/prebuilts/bin/sesearch
$ sesearch --help
Traceback (most recent call last):
[...]
    import networkx as nx
ImportError: No module named networkx
```

Instead, install the setools package

```
$ sudo apt install setools
```

sesearch query

Now we can query the policy using sesearch

Note: does not require access to the source `*.te` and `*_context` files

Example, list the allow rules for `httpd` when accessing target `httpd_data_file`:

```
1  $ adb pull /sys/fs/selinux/policy
2  $ /usr/bin/sesearch --allow -s httpd -t httpd_data_file policy
3  allow domain vendor_file_type:lnk_file { getattr open read };
4  allow httpd httpd_data_file:dir search;
5  allow httpd httpd_data_file:file { getattr open read };
6  allow httpd vendor_file_type:dir { getattr ioctl lock open read search watch watch_reads };
7  allow httpd vendor_file_type:file { execute getattr map open read };
8  allow httpd vendor_file_type:lnk_file { getattr read };
```

Line 3: `httpd` is derived from `domain`
Lines 4, 5: defined by me in the httpd.te file
Lines 6, 7, 8: `httpd_data_file` is derived from `vendor_file_type`

sesearch query

List *all* the allow rules for `httpd`

```
$ /usr/bin/sesearch --allow -s httpd policy
allow domain aaudio_config_prop:file { getattr map open read };
allow domain apex_mnt_dir:dir { getattr search };
allow domain apex_mnt_dir:lnk_file { getattr ioctl lock map open read watch watch_reads };
allow domain arm64_memtag_prop:file { getattr map open read };
allow domain ashmem_device:chr_file { append getattr ioctl lock map read write };
[...]
allow httpd vendor_file_type:file { execute getattr map open read };
allow httpd vendor_file_type:lnk_file { getattr read };
```

Total of 185

2. sepolicy-check

`sepolicy-check` is a useful tool for testing if a domain has a given permission

Build it:

```
$ croot
$ mmm system/sepolicy/tools
```

Usage:

```
sepolicy-check -s <source> -t <target> -c <class> -p <perm> -P <policy file>
```

Example: check that domain `httpd` can read an `httpd_data_file`

```
$ adb pull /sys/fs/selinux/policy
$ sepolicy-check -s httpd -t httpd_data_file -c file -p read -P policy
Match found!
```

But, there is no match if we check for write access

```
$ sepolicy-check -s httpd -t httpd_data_file -c file -p write -P policy
```

3. Hunt the inode

In this AVC, what file is the ino referring to?

```
avc: denied { search } for name="data" dev="dm-8" ino=97 scontext=u:r:simpleservice_app:s0
tcontext=u:object_r:system_data_file:s0:c512,c768 tclass=dir permissive=0
```

Find the mount point for device `dm-8`

```
# df -h | grep dm-8
/dev/block/dm-8                5.9G 548M  5.4G   9% /data
```

Use `find` to locate the inode

```
# find /data -inum 97
/data/data
/data/user/0
```

Answer: the AVC is caused when `simpleservice_app` tries to do a directory search in `/data/data` (or `/data/user/0` which is symlinked to `/data/data`)

audit2allow

- `audit2allow` is an SELinux tool that converts AVC audit logs into allow policy rules
- Quick (and lazy) way to generate SELinux policy

```
$ adb pull /sys/fs/selinux/policy
$ adb logcat -b all -d | audit2allow -p policy
#============= simpleservice_app ==============
allow simpleservice_app system_data_file:dir search;
```