


# Confining Spacewalk with SELinux

Jan Pazdziora
Principal Software Engineer
Satellite Engineering, Red Hat

Open Source Conference 2011
19th April 2011
Prague, Czech Republic

# What is Spacewalk?

- System management system.
  - With WebUI and XMLRPC API.
- The upstream for Red Hat Network Satellite and SuSE Manager.
- Written in Java, Python, and Perl.
- Tomcat, httpd with mod_perl and mod_python/mod_wsgi, monitoring, cobbler, database (Oracle, PostgreSQL), jabberd, osa-dispatcher.
- 700+ thousand lines of code.
- `http://spacewalk.redhat.com/`
- `https://fedorahosted.org/spacewalk/`

# What is SELinux?

- Yet another access control mechanism.
- Orthogonal to Un*x users, groups, access rights.

```
$ ps uZ | grep bash
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
                    adelton 32115 0.0 0.0 5248 1744 pts/63
                                          Ss 12:52   0:00 bash
$ ps uZ | grep 1380
system_u:system_r:hald_t:s0
                    root    1380  0.0 0.0 4008 1044 ?
                                      S 10:03   0:00 hald-runner
```

- We use targeted policy, only care about types (**_t**).
- What is not explicitly allowed is denied.

# The goal for confining

- All daemons run in their special domains or use domains native in the OS.
  - `httpd_t` (for Apache), `java_t` (tomcat)
  - `spacewalk_monitoring_t`
  - `oracle_db_t`, `oracle_tnslsnr_t`
  - `jabberd_t`, `osa_dispatcher_t`
- No unconfined_t, nor initrc_t.
- And there should be no AVC denials
  - Yes, we will see one in a minute.
- You get to love **Z**. You get to love iterative work.

# Creating SELinux policy modules

- Type enforcement

```
policy_module(swmon,1.0)
# Type for the processes
type swmon_t;
domain_type(swmon_t);
# Type for the startup script
type swmon_exec_t;
files_type(swmon_exec_t);
# Various macros that bring many allows
init_daemon_domain(swmon_t, swmon_exec_t)
```

- File contexts

```
# Define mapping of labels
/etc/rc\.d/np\.d/step                                    ⏎
            gen_context(system_u:object_r:swmon_exec_t,s0)
```

# Building and loading the module

```
# ln -s /usr/share/selinux/devel/Makefile
# make
Compiling targeted swmon module
/usr/bin/checkmodule:  loading policy configuration from tmp/
/usr/bin/checkmodule:  policy configuration loaded
/usr/bin/checkmodule:  writing binary representation (versior
Creating targeted swmon.pp policy package
rm tmp/swmon.mod tmp/swmon.mod.fc
# semodule -i swmon.pp
```

# Check the transition

```
# cp /bin/sleep .
# chcon -t swmon_exec_t sleep
# cat > init.sh
#!/bin/bash
./sleep 10 &
ps --no-headers -Zp $! | awk '{print $1}'
Ctrl+D
# chmod a+x init.sh
# chcon -t initrc_exec_t init.sh
# ./init.sh
unconfined_u:system_r:swmon_t:s0
```

- Here we forced the context with chcon.
- For production we use restorecon which sets the context based on file context definitions.

# Make it possible to restart

- Type enforcement

```
require {
    type java_t;
    type initrc_t;
}

type sw_initrc_exec_t;
domain_entry_file(initrc_t, sw_initrc_exec_t)
domain_auto_trans(java_t, sw_initrc_exec_t, initrc_t)
```

- File contexts

```
/sbin/rhn-sat-restart-silent                                    ⏎
      gen_context(system_u:object_r:sw_initrc_exec_t,s0)
```

# Oracle RDBMS

- Source not available, prime target for confining.
- Oracle SELinux policy module written by Rob Myers.
- We ended up splitting the file contexts (`.fc`) to separate module:

```
# semodule -l | grep oracle
oracle-nofcontext 1.1.2
oracle-port 1.1.2
oracle-xe 10.2.0.19.1
```

- The `-nofcontext` has all the allows and no paths (file contexts), the `oracle-xe` has only the paths.
- We ship `oracle-rhnsat` with RHN Satellite for the layout of the embedded Oracle installation.

# Example of AVC denial

- In `/var/log/audit/audit.log`.
- Process wants to read its own pid file:

```
avc:  denied  { read } for pid=3169
comm="osa-dispatcher" name="osa-dispatcher.pid"
dev=dm-0 ino=516358
scontext=unconfined_u:system_r:osa_dispatcher_t:s0
tcontext=unconfined_u:object_r:osa_dispatcher_var_run_t:s0
tclass=file
```

- Should we `allow` it?

```
allow osa_dispatcher_t osa_dispatcher_var_run_t:file read;
```

# Example of AVC denial (cont'd)

- Actually, the process does not need to read the file, change the application:

```
- fd = os.open(pid_file, os.O_RDWR | os.O_CREAT, 0644)
+ fd = os.open(pid_file, \
+              os.O_WRONLY | os.O_APPEND | os.O_CREAT, 0644)
```

- To `allow` is not always the best course of action.
- Often, SELinux helps us to find issues in our code.
- Or in someone else's code.

# Example: TCP socket connect

```
avc: denied { name_connect }
for pid=12935 comm="osa-dispatcher" dest=5432
scontext=system_u:system_r:osa_dispatcher_t:s0
tcontext=system_u:object_r:postgresql_port_t:s0
tclass=tcp_socket
```

- The `audit2allow` command prints `allows` needed:

```
allow osa_dispatcher_t postgresql_port_t:tcp_socket ⏎
                                              name_connect;
```

- With `-R` option, it tries to use existing interfaces instead of raw `allows`:

```
corenet_tcp_connect_postgresql_port(osa_dispatcher_t)
```

# Example: directory searched

```
avc:  denied  { search } for pid=20967 comm="oracle"
name="log" dev=dm-0 ino=657566
scontext=system_u:system_r:oracle_db_t:s0
tcontext=user_u:object_r:oracle_tnslsnr_log_t:s0
tclass=dir
```

- Setting SELinux to permissive reveals other actions that were stopped by the first failed one:

```
{ search } for pid=20967 comm="oracle" name="log" dev=dm-0 ir
{ write } for pid=20967 comm="oracle" name="log" dev=dm-0 inc
{ add_name } for pid=20967 comm="oracle" name="sqlnet.log" sc
{ create } for pid=20967 comm="oracle" name="sqlnet.log" scor
{ append open } for pid=20967 comm="oracle" name="sqlnet.log"
{ getattr } for pid=20967 comm="oracle" path="/usr/lib/oracle
```

# Example: directory searched (cont'd)

- Allow Oracle database process to create its error log in listener's directory:

```
filetrans_pattern(oracle_db_t, oracle_tnslsnr_log_t,    ⏎
                                 oracle_db_log_t, { file })
create_files_pattern(oracle_db_t, oracle_tnslsnr_log_t, ⏎
                                        oracle_db_log_t)
```

```
/usr/lib/oracle/xe/(.*/)?network/log/sqlnet\.log(.*)?   ⏎
          gen_context(user_u:object_r:oracle_db_log_t,s0)
```

- Many AVC denials are about logging errors.
- Which only happen from time to time.
- Clues for this one were found in database's alert log, based on timestamps.

# Example: reading from cron's pipe

```
avc: denied { ioctl } for pid=13527 comm="Monitoring"
path="pipe:[4553708]" dev=pipefs ino=4553708
scontext=user_u:system_r:spacewalk_monitoring_t:s0
tcontext=system_u:system_r:crond_t:s0-s0:c0.c1023
tclass=fifo_file
```

- Our programs reads from crond. How could it happen?
- The reason was that crond run logrotate which then restarted our program.
- And it inherited stdin from crond.
- The actual fix in logrotate's config file:

```
-  /sbin/service Monitoring restart > /dev/null 2>/dev/null |
+  /sbin/service Monitoring restart < /dev/null >/dev/null 2>
```

# Example: Apache and rpm database

- Apache httpd deamon attempts to read the rpm database:

```
avc:  denied  { search } for  pid=10689 comm="httpd"
name="rpm" dev=dm-0 ino=1179651
scontext=unconfined_u:system_r:httpd_t:s0
tcontext=system_u:object_r:rpm_var_lib_t:s0 tclass=dir
avc:  denied  { getattr } for  pid=10689 comm="httpd"
path="/var/lib/rpm" dev=dm-0 ino=1179651
scontext=unconfined_u:system_r:httpd_t:s0
tcontext=system_u:object_r:rpm_var_lib_t:s0 tclass=dir
avc:  denied  { open } for  pid=10689 comm="httpd"
name="Packages" dev=dm-0 ino=1179656
scontext=unconfined_u:system_r:httpd_t:s0
tcontext=system_u:object_r:rpm_var_lib_t:s0 tclass=file
```

# Example: Apache and rpm db (cont'd)

- Luckily we knew that it was happening while a mod_python process was processing uploaded rpm.
- And it did not need to look at the database after all:

```
dontaudit httpd_t rpm_var_lib_t:dir list_dir_perms;
dontaudit httpd_t rpm_var_lib_t:file read_file_perms;
```

# Conclusion

- `https://fedorahosted.org/spacewalk/wiki/Features/SELinux`
- We put the exact AVC denial to commit message which addresses the problem in Spacewalk git repo.
  - For our reference.
  - It also makes it easy for you to search through our fixes.
  - Comments and patches most welcome.
- Thank you for your attention.