LAB 3: SELinux

Initial Set-up:

I ran the sestatus command to check if the selinux is in enforcing mode.

```
[sowmyashree@localhost ~]$ sestatus
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   enforcing
Mode from config file:          enforcing
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Max kernel policy version:      31
[sowmyashree@localhost ~]$
```

Task 1: Running testprog:

I ran binary from the shell using the given command. The binary was executed successfully which is not expected since we are running the selinux in enforcing mode. Enforcing mode denies applications from running if there is no particular security policy/incorrect policy written for them. Also noticed that the testprog is running in an unconfined space which is not highly restricted which could be the reason why we were able to run it.

```
[sowmyashree@localhost shared]$ sudo /usr/bin/testprog /etc/testprog.conf /var/run/testprog.pid &
[1] 1685
[sowmyashree@localhost shared]$ Using configuration file: /etc/testprog.conf
Wrote PID to /var/run/testprog.pid
Writing output to: /var/testprog/testprg.txt
Iteration count: -1
```

Task 2:

I ran the testprog as a service for the second time. I was still able to run it. The default behavior of selinux is that it inherits context from the parent context unless there exists a policy rule that specifies otherwise. But here, when I ran the testprog it didn't do so inferring that there is some selinux policy rule behind this.

```
[sowmyashree@localhost shared]$ fg sudo /usr/bin/testprog /etc/testprog.conf /var/run/testprog.pid
sudo /usr/bin/testprog /etc/testprog.conf /var/run/testprog.pid
^C
[sowmyashree@localhost shared]$ sudo systemctl start testprog
[sudo] password for sowmyashree:
[sowmyashree@localhost shared]$ sudo systemctl status testprog
● testprog.service - SELinux Test Program
   Loaded: loaded (/etc/systemd/system/testprog.service; enabled; vendor preset: disabled)
   Active: active (running) since Thu 2021-03-04 17:20:54 EST; 13s ago
 Main PID: 11894 (testprog)
   CGroup: /system.slice/testprog.service
           └─11894 /usr/bin/testprog /etc/testprog.conf /var/run/testprog.pid

Mar 04 17:20:54 localhost.localdomain systemd[1]: Started SELinux Test Program.
Mar 04 17:20:54 localhost.localdomain testprog[11894]: Using configuration file: /etc/testprog.conf
Mar 04 17:20:54 localhost.localdomain testprog[11894]: Wrote PID to /var/run/testprog.pid
Mar 04 17:20:54 localhost.localdomain testprog[11894]: Writing output to: /var/testprog/testprg.txt
Mar 04 17:20:54 localhost.localdomain testprog[11894]: Iteration count: -1
Hint: Some lines were ellipsized, use -l to show in full.
```

```
[sowmyashree@localhost shared]$ ps -efZ | grep $(cat /var/run/testprog.pid)
system_u:system_r:unconfined_service_t:s0 root 11894 1  0 17:20 ?      00:00:00 /usr/bin/testprog /e
tc/testprog.conf /var/run/testprog.pid
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 sowmyas+ 11903 1591  0 17:27 tty4 00:00:00 gre
p --color=auto 11894
[sowmyashree@localhost shared]$ _
```

Task 3:

I added selinux policy that was given to us. Once the policy was loaded, I ran the testprog once again to check its context. The testprog was still running in unconfined space which is not intended.

```
[sowmyashree@localhost shared]$ sudo semodule -i testprog.pp
[sowmyashree@localhost shared]$ sudo semodule -l | grep testprog
testprog        0.1
[sowmyashree@localhost shared]$ _
```

```
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 root 1726 1534  0 12:56 tty1 00:00:00 grep --c
olor=auto 1721
```

Task 4:

Adding policy module was not enough as seen in previous task, so I tried to change the context of the file itself using the 2 suggested methods:

a) chcon : by the running this command we were able to change from unconfined space, however, if the filesystem were to be restored, it would restore to the default type which is bin_t.

```
[sowmyashree@localhost shared]$ sudo chcon -t testprog_exec_t /usr/bin/testprog
[sudo] password for sowmyashree:
[sowmyashree@localhost shared]$ ls -Z /usr/bin/testprog
-rwxr-xr-x. root root unconfined_u:object_r:testprog_exec_t:s0 /usr/bin/testprog
[sowmyashree@localhost shared]$
```

```
restorecon reset /usr/bin/testprog context unconfined_u:object_r:testprog_exec_t:s0->unconfined_u:ob
ject_r:bin_t:s0
```

b) I created a new context file testprog.fc and rebuilt the policy and loaded it. Here, even if I restore the filesystem (/ file configuration), the default context will be set to testprog_exec_t. And when I ran testprog binary from the shell it gave a segmentation fault. This confirms that the context is changed from unconfined space which wasn't restricted to the one that is restricted.

```
[sowmyashree@localhost shared]$ echo "/usr/bin/testprog -- system_u:object_r:testprog_exec_t:s0" > t
estprog.fc
[sowmyashree@localhost shared]$ cat testprog.fc
/usr/bin/testprog -- system_u:object_r:testprog_exec_t:s0
[sowmyashree@localhost shared]$
```

```
[sowmyashree@localhost ~]$ sudo restorecon -v /usr/bin/testprog
[sowmyashree@localhost ~]$ ls -Z /usr/bin/testprog
-rwxr-xr-x. root root unconfined_u:object_r:testprog_exec_t:s0 /usr/bin/testprog
[sowmyashree@localhost ~]$
```

```
[sowmyashree@localhost ~]$ sudo /usr/bin/testprog /etc/testprog.conf /var/run/testprog.pid &
[1] 1574
[sowmyashree@localhost ~]$
[1]+  Segmentation fault      sudo /usr/bin/testprog /etc/testprog.conf /var/run/testprog.pid
[sowmyashree@localhost ~]$
```

Task 5:

I tried to access the audit file with entries associated to only testprog and tried to filter it further. Even with this, the output generated was quite large.

```
xt=system_u:object_r:console_device_t:s0 tclass=chr_file permissive=0
type=SYSCALL msg=audit(1614907602.934:150): arch=c000003e syscall=2 success=no exit=-13 a0=7faf427c5
49d a1=101 a2=0 a3=1ffb items=0 ppid=1574 pid=1576 auid=1000 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=
0 sgid=0 fsgid=0 tty=(none) ses=1 comm="testprog" exe="/usr/bin/testprog" subj=unconfined_u:unconfi
ed_r:testprog_t:s0-s0:c0.c1023 key=(null)
type=AVC msg=audit(1614907602.935:151): avc:  denied  { write } for  pid=1576 comm="testprog" name="
testprog.pid" dev="tmpfs" ino=14663 scontext=unconfined_u:unconfined_r:testprog_t:s0-s0:c0.c1023 tcc
ntext=system_u:object_r:init_var_run_t:s0 tclass=file permissive=0
type=SYSCALL msg=audit(1614907602.935:151): arch=c000003e syscall=2 success=no exit=-13 a0=7ffce69ea
8c2 a1=241 a2=1b6 a3=24 items=0 ppid=1574 pid=1576 auid=1000 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=
0 sgid=0 fsgid=0 tty=(none) ses=1 comm="testprog" exe="/usr/bin/testprog" subj=unconfined_u:unconfi
ed_r:testprog_t:s0-s0:c0.c1023 key=(null)
type=ANOM_ABEND msg=audit(1614907602.935:152): auid=1000 uid=0 gid=0 ses=1 subj=unconfined_u:unconf
ned_r:testprog_t:s0-s0:c0.c1023 pid=1576 comm="testprog" reason="memory violation" sig=11
[sowmyashree@localhost ~]$ _
```

```
type=AVC msg=audit(1614907602.931:147): avc:  denied  { read write } for  pid=1576 comm="testprog"
ath="/dev/tty1" dev="devtmpfs" ino=1043 scontext=unconfined_u:unconfined_r:testprog_t:s0-s0:c0.c102
 tcontext=unconfined_u:object_r:user_tty_device_t:s0 tclass=chr_file permissive=0
type=AVC msg=audit(1614907602.931:147): avc:  denied  { read write } for  pid=1576 comm="testprog"
ath="/dev/tty1" dev="devtmpfs" ino=1043 scontext=unconfined_u:unconfined_r:testprog_t:s0-s0:c0.c102
 tcontext=unconfined_u:object_r:user_tty_device_t:s0 tclass=chr_file permissive=0
type=AVC msg=audit(1614907602.933:148): avc:  denied  { create } for  pid=1576 comm="testprog" scon
ext=unconfined_u:unconfined_r:testprog_t:s0-s0:c0.c1023 tcontext=unconfined_u:unconfined_r:testprog_
t:s0-s0:c0.c1023 tclass=unix_dgram_socket permissive=0
type=AVC msg=audit(1614907602.934:149): avc:  denied  { create } for  pid=1576 comm="testprog" scon
ext=unconfined_u:unconfined_r:testprog_t:s0-s0:c0.c1023 tcontext=unconfined_u:unconfined_r:testprog_
t:s0-s0:c0.c1023 tclass=unix_dgram_socket permissive=0
type=AVC msg=audit(1614907602.934:150): avc:  denied  { write } for  pid=1576 comm="testprog" name="
console" dev="devtmpfs" ino=1037 scontext=unconfined_u:unconfined_r:testprog_t:s0-s0:c0.c1023 tcont
xt=system_u:object_r:console_device_t:s0 tclass=chr_file permissive=0
type=AVC msg=audit(1614907602.935:151): avc:  denied  { write } for  pid=1576 comm="testprog" name=
estprog.pid" dev="tmpfs" ino=14663 scontext=unconfined_u:unconfined_r:testprog_t:s0-s0:c0.c1023 tco
text=system_u:object_r:init_var_run_t:s0 tclass=file permissive=0
[sowmyashree@localhost ~]$ _
```

I ran sealert tool with audit log files. In there, it provided a policy fix (with 100% confidence) which would enable testprog to write into the PID file and also provided as to how generate a new policy module for this.

When I ran the first command it suggested and investigated the my-testprog.te file it generated. Applying that policy would mean that there would be two policies that monitor the testprog file. And it also suggests that the write access must be allowed in var_run_t type which is already existing for the /var/run directory. This would mean that the testprog would have write access to all the files inside this directory. In addition to this we will have to enable read access to the conf

file which resides in the etc directory which holds much sensitive info (like passwd, shadow files storing the hashed passwords of the users). Allowing this would not be a secure option even though it is better than running in unconfined space. Hence the best option would be to go ahead with the policy we wrote on our own.

```
SELinux is preventing /usr/bin/testprog from write access on the file testprog.pid.

*****  Plugin catchall (100. confidence) suggests   **************************

If you believe that testprog should be allowed write access on the testprog.pid file by default.
Then you should report this as a bug.
You can generate a local policy module to allow this access.
Do
allow this access for now by executing:
# ausearch -c 'testprog' --raw | audit2allow -M my-testprog
# semodule -i my-testprog.pp
```

```
[sowmyashree@localhost ~]$ sudo ausearch -c 'testprog' --raw | audit2allow -M my-testprog
[sudo] password for sowmyashree:
******************** IMPORTANT ***********************
To make this policy package active, execute:

semodule -i my-testprog.pp
```

```
[sowmyashree@localhost ~]$ cat my-testprog.te

module my-testprog 1.0;

require {
        type var_run_t;
        type console_device_t;
        type init_var_run_t;
        type user_tty_device_t;
        type testprog_t;
        class unix_dgram_socket create;
        class chr_file { append read write };
        class file write;
}

#============= testprog_t ==============
allow testprog_t console_device_t:chr_file write;
allow testprog_t init_var_run_t:file write;
```