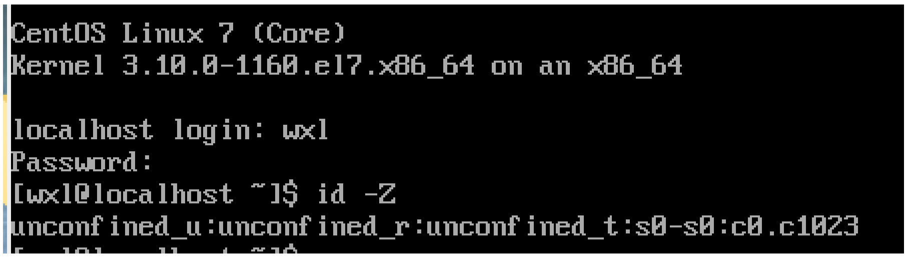
selinux 实验报告

一. 第一部分实验

1. 改变用户绑定的security context

- 首先检查当前用户的security context，显示**操作前用户`wxl`的security context** 为`unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023`

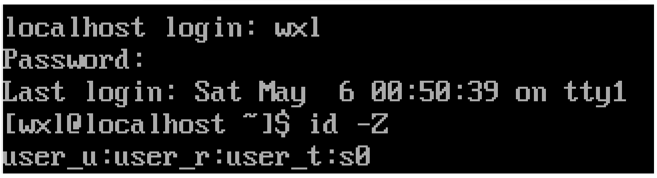


```
CentOS Linux 7 (Core)
Kernel 3.10.0-1160.el7.x86_64 on an x86_64

localhost login: wxl
Password:
[wxl@localhost ~]$ id -Z
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

- 加载新配置，使用命令：

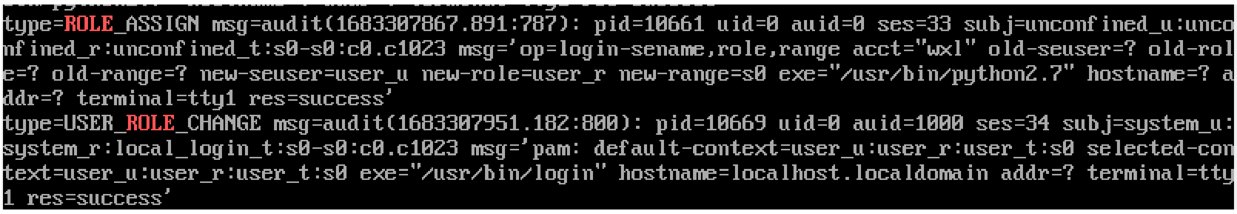
```
semanage login -a -s user_u wxl
```

- 最后重新登录用户`wxl`，使用`id -Z`命令检查，**操作后用户`wxl`的security context** 为`user_u:user_r:user_t:s0`



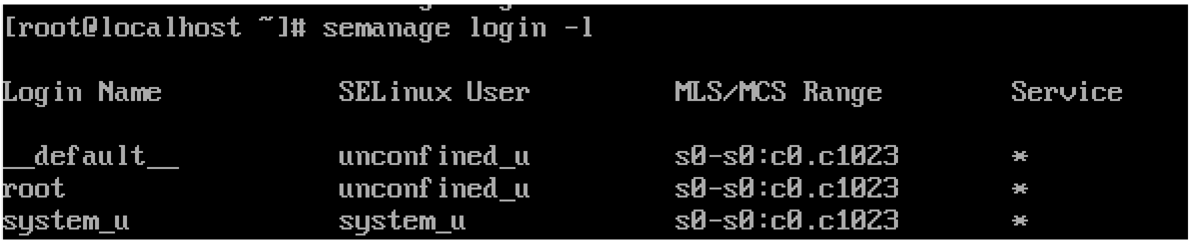
```
localhost login: wxl
Password:
Last login: Sat May  6 00:50:39 on tty1
[wxl@localhost ~]$ id -Z
user_u:user_r:user_t:s0
```

- 在`/var/log/audit/audit.log`文件中查看相关审计日志



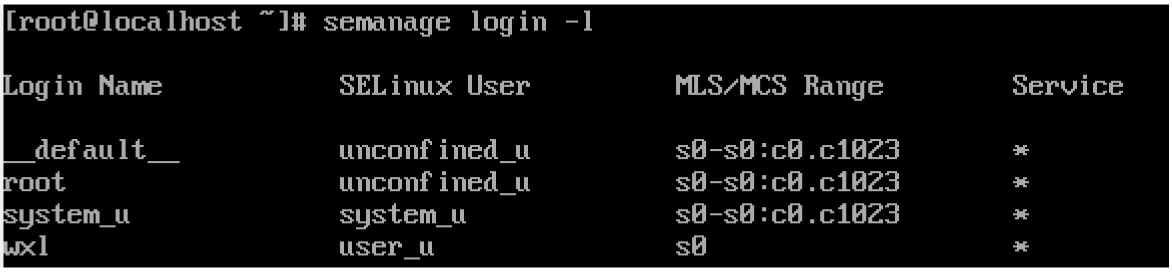
```
type=ROLE_ASSIGN msg=audit(1683307867.891:787): pid=10661 uid=0 auid=0 ses=33 subj=unconfined_u:unco
nfined_r:unconfined_t:s0-s0:c0.c1023 msg='op=login-sename,role,range acct="wxl" old-seuser=? old-rol
e=? old-range=? new-seuser=user_u new-role=user_r new-range=s0 exe="/usr/bin/python2.7" hostname=? a
ddr=? terminal=tty1 res=success'
type=USER_ROLE_CHANGE msg=audit(1683307951.182:800): pid=10669 uid=0 auid=1000 ses=34 subj=system_u:
system_r:local_login_t:s0-s0:c0.c1023 msg='pam: default-context=user_u:user_r:user_t:s0 selected-con
text=user_u:user_r:user_t:s0 exe="/usr/bin/login" hostname=localhost.localdomain addr=? terminal=tty
1 res=success'
```

操作前：



```
[root@localhost ~]# semanage login -l

Login Name           SELinux User         MLS/MCS Range        Service

__default__          unconfined_u         s0-s0:c0.c1023       *
root                 unconfined_u         s0-s0:c0.c1023       *
system_u             system_u             s0-s0:c0.c1023       *
```

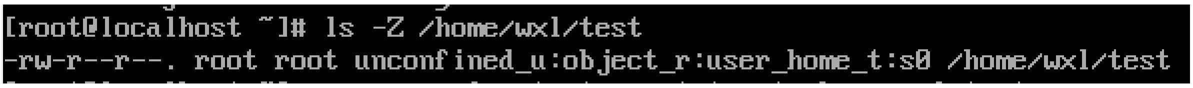
操作后：



```
[root@localhost ~]# semanage login -l

Login Name           SELinux User         MLS/MCS Range        Service

__default__          unconfined_u         s0-s0:c0.c1023       *
root                 unconfined_u         s0-s0:c0.c1023       *
system_u             system_u             s0-s0:c0.c1023       *
wxl                  user_u               s0                   *
```

2. 改变文件的security context

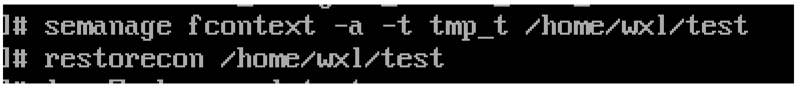
- 操作前文件`/home/wxl/test`上下文：



```
[root@localhost ~]# ls -Z /home/wxl/test
-rw-r--r--. root root unconfined_u:object_r:user_home_t:s0 /home/wxl/test
```

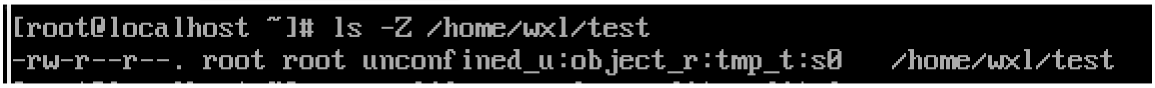
- 配置文件上下文&生效上下文：

```
semanage fcontext -a -t tmp_t /home/wxl/test
//将test文件的type改为tmp_t
restorecon /home/wxl/test
```



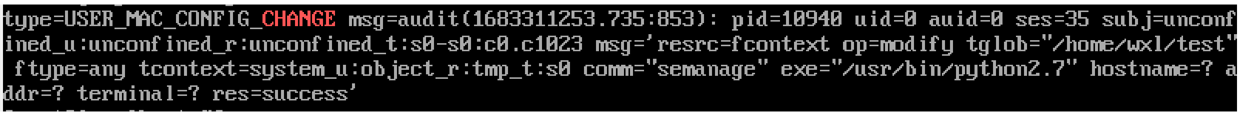
```
]# semanage fcontext -a -t tmp_t /home/wxl/test
]# restorecon /home/wxl/test
```

- 操作后文件`/home/wxl/test`上下文：



```
[root@localhost ~]# ls -Z /home/wxl/test
-rw-r--r--. root root unconfined_u:object_r:tmp_t:s0   /home/wxl/test
```

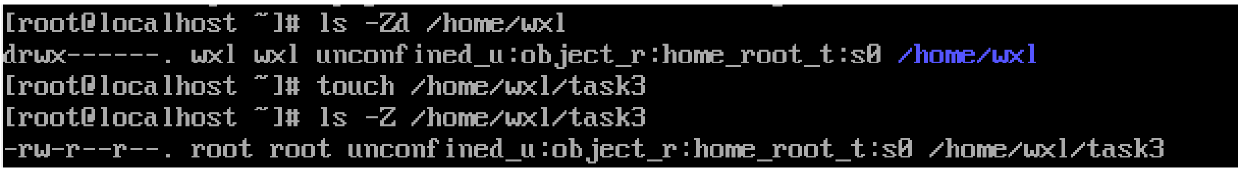
- 在`/var/log/audit/audit.log`文件中查看相关审计日志



```
type=USER_MAC_CONFIG_CHANGE msg=audit(1683311253.735:853): pid=10940 uid=0 auid=0 ses=35 subj=unconf
ined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 msg='resrc=fcontext op=modify tglob="/home/wxl/test"
 ftype=any tcontext=system_u:object_r:tmp_t:s0 comm="semanage" exe="/usr/bin/python2.7" hostname=? a
ddr=? terminal=? res=success'
```

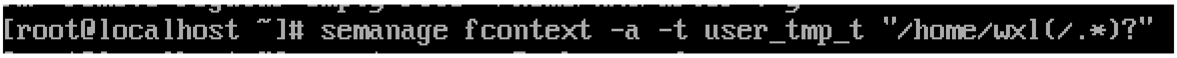
3. 自定义新创建文件的security context（默认是继承父目录的context）

- 操作前在/home/wxl目录下创建文件，新创建文件的context：

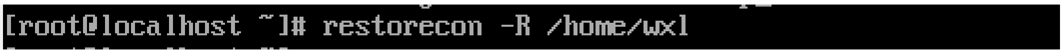


```
[root@localhost ~]# ls -Zd /home/wxl
drwx------. wxl wxl unconfined_u:object_r:home_root_t:s0 /home/wxl
[root@localhost ~]# touch /home/wxl/task3
[root@localhost ~]# ls -Z /home/wxl/task3
-rw-r--r--. root root unconfined_u:object_r:home_root_t:s0 /home/wxl/task3
```

- 配置

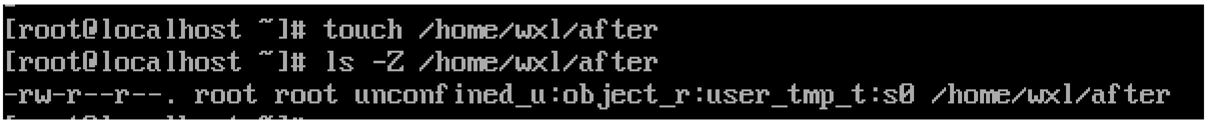


```
[root@localhost ~]# semanage fcontext -a -t user_tmp_t "/home/wxl(/.*)?"
```



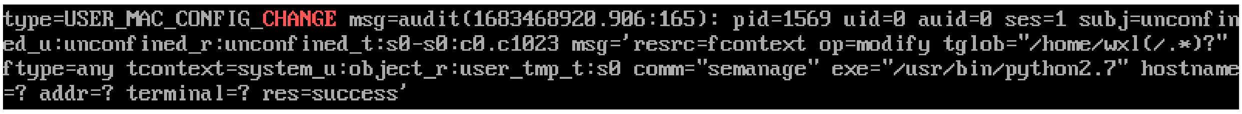
```
[root@localhost ~]# restorecon -R /home/wxl
```

- 操作后在/home/wxl目录下创建文件，新创建文件的context：



```
[root@localhost ~]# touch /home/wxl/after
[root@localhost ~]# ls -Z /home/wxl/after
-rw-r--r--. root root unconfined_u:object_r:user_tmp_t:s0 /home/wxl/after
```

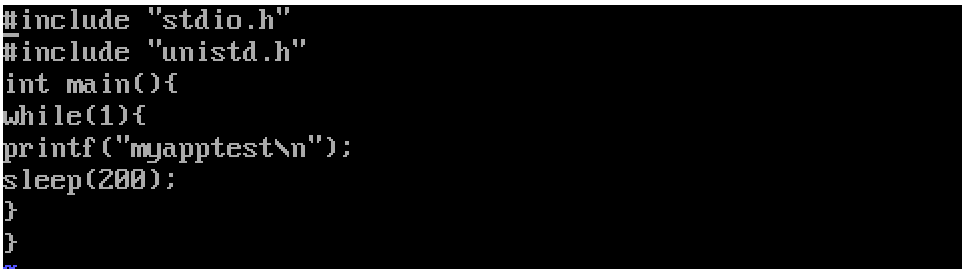
- 查看日志信息



```
type=USER_MAC_CONFIG_CHANGE msg=audit(1683468920.906:165): pid=1569 uid=0 auid=0 ses=1 subj=unconfin
ed_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 msg='resrc=fcontext op=modify tglob="/home/wxl(/.*)?"
ftype=any tcontext=system_u:object_r:user_tmp_t:s0 comm="semanage" exe="/usr/bin/python2.7" hostname
=? addr=? terminal=? res=success'
```

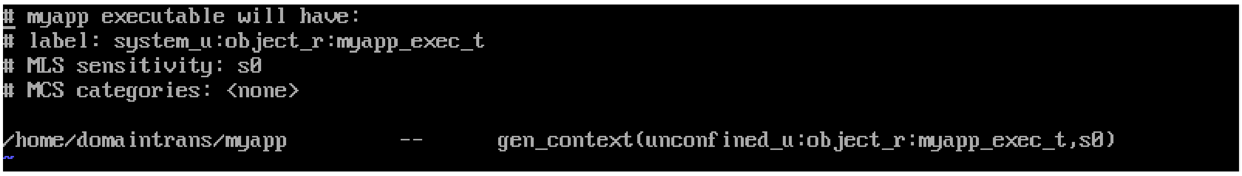
4. 配置SELinux，实现进程domain的type transition

- myapp.c gcc -o myapp myapp.c



```
#include "stdio.h"
#include "unistd.h"
int main(){
while(1){
printf("myapptest\n");
sleep(200);
}
}
~
```

- 给myapp增加上下文 myapp.fc restorecon myapp

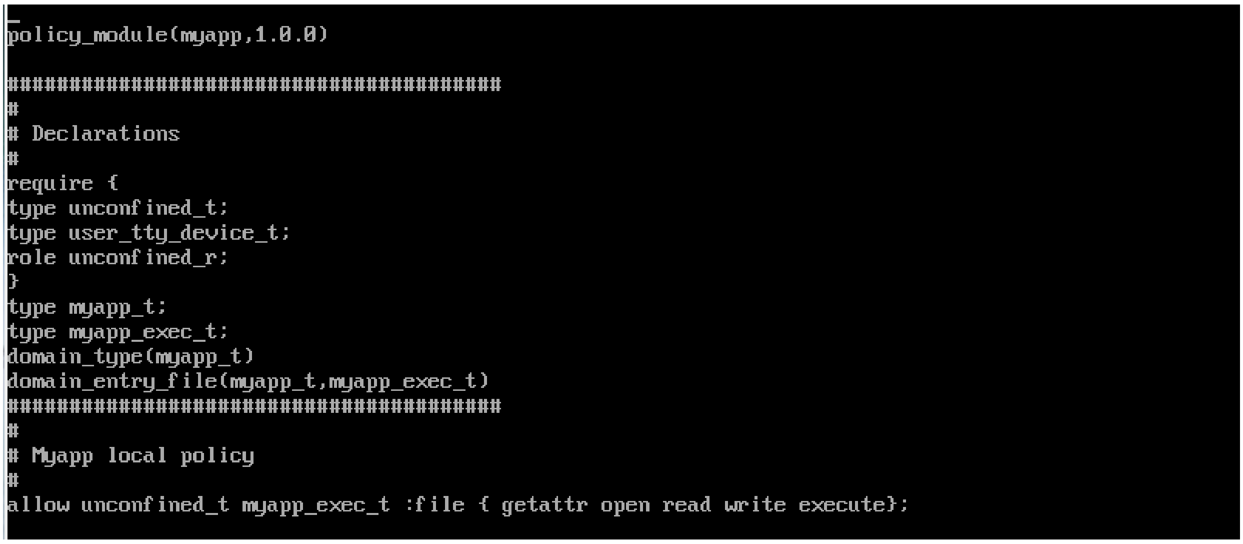


```
# myapp executable will have:
# label: system_u:object_r:myapp_exec_t
# MLS sensitivity: s0
# MCS categories: <none>

/home/domaintrans/myapp         --      gen_context(unconfined_u:object_r:myapp_exec_t,s0)
~
```

- 需要满足的条件myapp.te

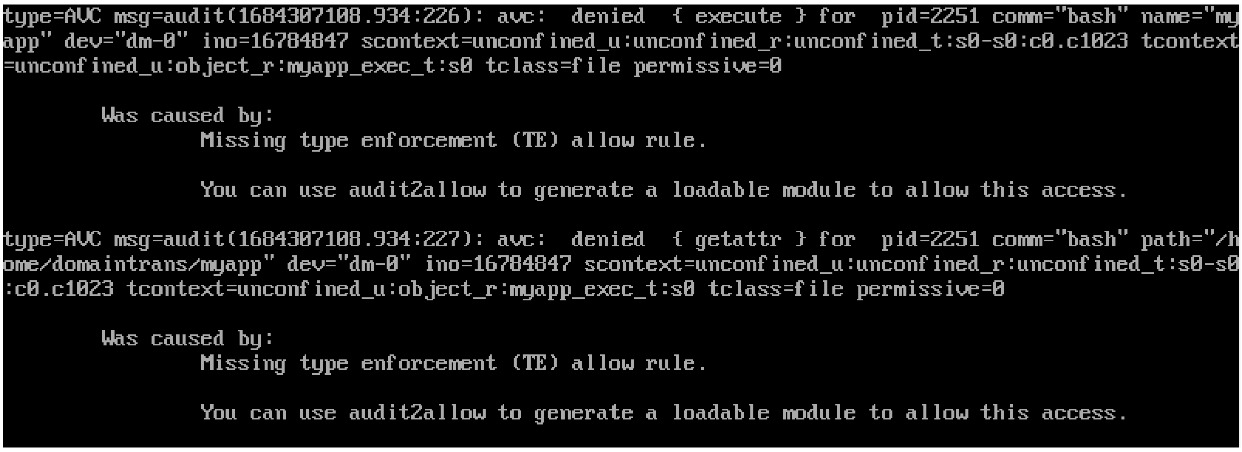
```
role unconfined_r types myapp_t;
allow unconfined_t myapp_exec_t : file { getattr open read write execute};
allow myapp_t myapp_exec_t : file { entrypoint read write map execute};
allow unconfined_t myapp_t :process transition;
allow myapp_t unconfined_t :process sigchld;
allow myapp_t user_tty_device_t :chr_file { ioctl read write getattr open execute};
```

- myapp.te文件

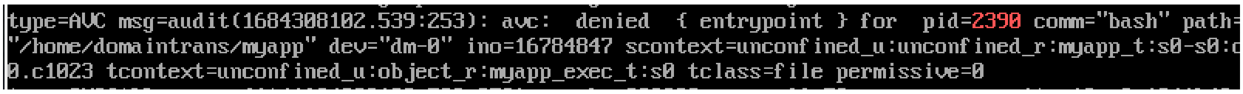


```
policy_module(myapp,1.0.0)

########################################
#
# Declarations
#
require {
type unconfined_t;
type user_tty_device_t;
role unconfined_r;
}
type myapp_t;
type myapp_exec_t;
domain_type(myapp_t)
domain_entry_file(myapp_t,myapp_exec_t)
########################################
#
# Myapp local policy
#
allow unconfined_t myapp_exec_t :file { getattr open read write execute};
```

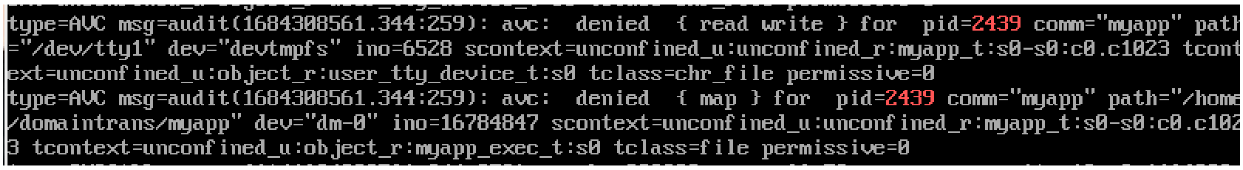
- 不满足条件时的日志信息



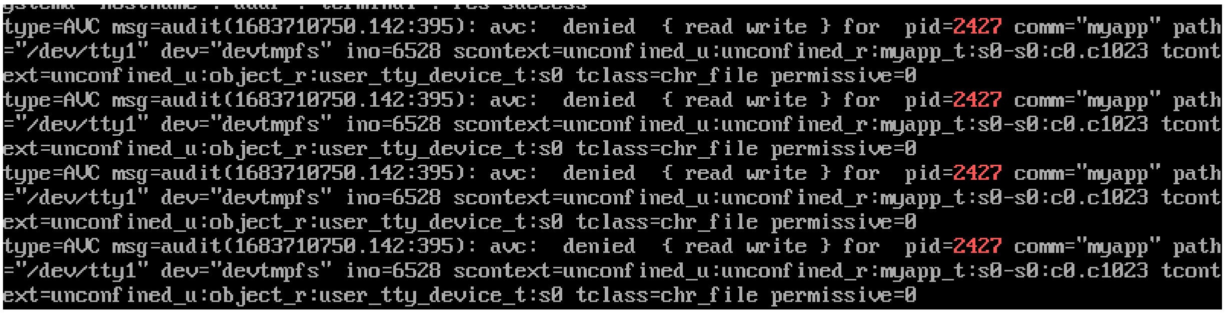
```
type=AVC msg=audit(1684307108.934:226): avc:  denied  { execute } for  pid=2251 comm="bash" name="my
app" dev="dm-0" ino=16784847 scontext=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 tcontext
=unconfined_u:object_r:myapp_exec_t:s0 tclass=file permissive=0

        Was caused by:
                Missing type enforcement (TE) allow rule.

                You can use audit2allow to generate a loadable module to allow this access.

type=AVC msg=audit(1684307108.934:227): avc:  denied  { getattr } for  pid=2251 comm="bash" path="/h
ome/domaintrans/myapp" dev="dm-0" ino=16784847 scontext=unconfined_u:unconfined_r:unconfined_t:s0-s0
:c0.c1023 tcontext=unconfined_u:object_r:myapp_exec_t:s0 tclass=file permissive=0

        Was caused by:
                Missing type enforcement (TE) allow rule.

                You can use audit2allow to generate a loadable module to allow this access.
```



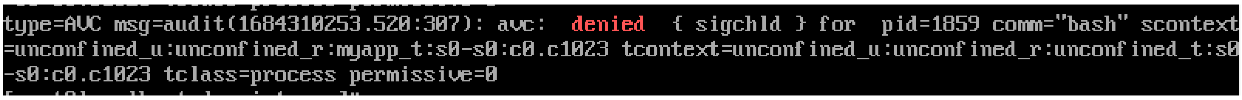
```
type=AVC msg=audit(1684308102.539:253): avc:  denied  { entrypoint } for  pid=2390 comm="bash" path=
"/home/domaintrans/myapp" dev="dm-0" ino=16784847 scontext=unconfined_u:unconfined_r:myapp_t:s0-s0:c
0.c1023 tcontext=unconfined_u:object_r:myapp_exec_t:s0 tclass=file permissive=0
```



```
type=AVC msg=audit(1684308561.344:259): avc:  denied  { read write } for  pid=2439 comm="myapp" path
="/dev/tty1" dev="devtmpfs" ino=6528 scontext=unconfined_u:unconfined_r:myapp_t:s0-s0:c0.c1023 tcont
ext=unconfined_u:object_r:user_tty_device_t:s0 tclass=chr_file permissive=0
type=AVC msg=audit(1684308561.344:259): avc:  denied  { map } for  pid=2439 comm="myapp" path="/home
/domaintrans/myapp" dev="dm-0" ino=16784847 scontext=unconfined_u:unconfined_r:myapp_t:s0-s0:c0.c102
3 tcontext=unconfined_u:object_r:myapp_exec_t:s0 tclass=file permissive=0
```



```
type=AVC msg=audit(1683710750.142:395): avc:  denied  { read write } for  pid=2427 comm="myapp" path
="/dev/tty1" dev="devtmpfs" ino=6528 scontext=unconfined_u:unconfined_r:myapp_t:s0-s0:c0.c1023 tcont
ext=unconfined_u:object_r:user_tty_device_t:s0 tclass=chr_file permissive=0
type=AVC msg=audit(1683710750.142:395): avc:  denied  { read write } for  pid=2427 comm="myapp" path
="/dev/tty1" dev="devtmpfs" ino=6528 scontext=unconfined_u:unconfined_r:myapp_t:s0-s0:c0.c1023 tcont
ext=unconfined_u:object_r:user_tty_device_t:s0 tclass=chr_file permissive=0
type=AVC msg=audit(1683710750.142:395): avc:  denied  { read write } for  pid=2427 comm="myapp" path
="/dev/tty1" dev="devtmpfs" ino=6528 scontext=unconfined_u:unconfined_r:myapp_t:s0-s0:c0.c1023 tcont
ext=unconfined_u:object_r:user_tty_device_t:s0 tclass=chr_file permissive=0
type=AVC msg=audit(1683710750.142:395): avc:  denied  { read write } for  pid=2427 comm="myapp" path
="/dev/tty1" dev="devtmpfs" ino=6528 scontext=unconfined_u:unconfined_r:myapp_t:s0-s0:c0.c1023 tcont
ext=unconfined_u:object_r:user_tty_device_t:s0 tclass=chr_file permissive=0
```

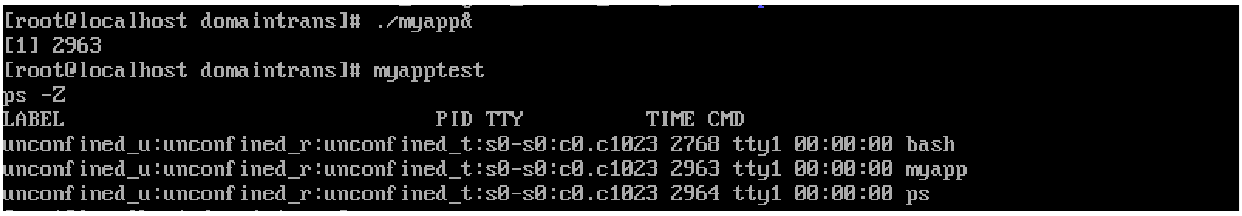


```
type=AVC msg=audit(1684310253.520:307): avc:  denied  { sigchld } for  pid=1859 comm="bash" scontext
=unconfined_u:unconfined_r:myapp_t:s0-s0:c0.c1023 tcontext=unconfined_u:unconfined_r:unconfined_t:s0
-s0:c0.c1023 tclass=process permissive=0
```

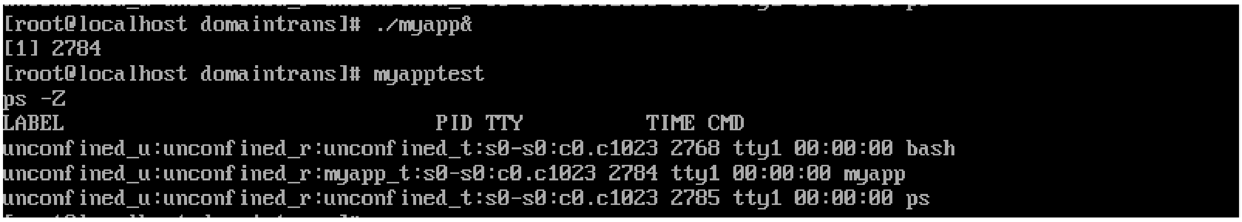
- 编译，加载策略

```
make
semodule -i myapp.pp
```

- 未配置transition规则时进程被执行后的context和配置transition 规则后进程执行时的context:



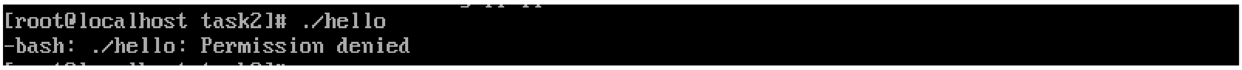
```
[root@localhost domaintrans]# ./myapp&
[1] 2963
[root@localhost domaintrans]# myapptest
ps -Z
LABEL                             PID TTY          TIME CMD
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 2768 tty1 00:00:00 bash
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 2963 tty1 00:00:00 myapp
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 2964 tty1 00:00:00 ps
```



```
[root@localhost domaintrans]# ./myapp&
[1] 2784
[root@localhost domaintrans]# myapptest
ps -Z
LABEL                             PID TTY          TIME CMD
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 2768 tty1 00:00:00 bash
unconfined_u:unconfined_r:myapp_t:s0-s0:c0.c1023 2784 tty1 00:00:00 myapp
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 2785 tty1 00:00:00 ps
```

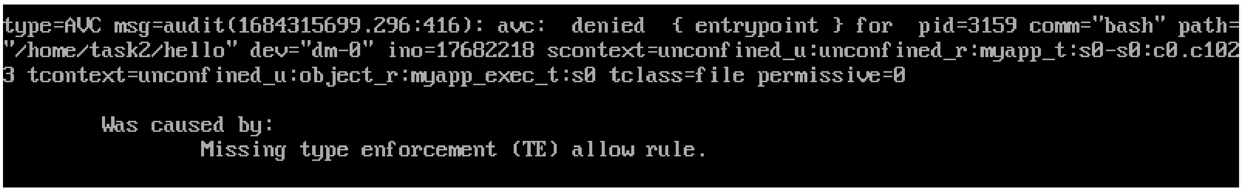
二. 第二部分实验

- 在myapp.te中添加`files_type(myapp_exec_t)`，编译并加载策略，然后`chcon -t myapp_exec_t hello`
- 执行hello

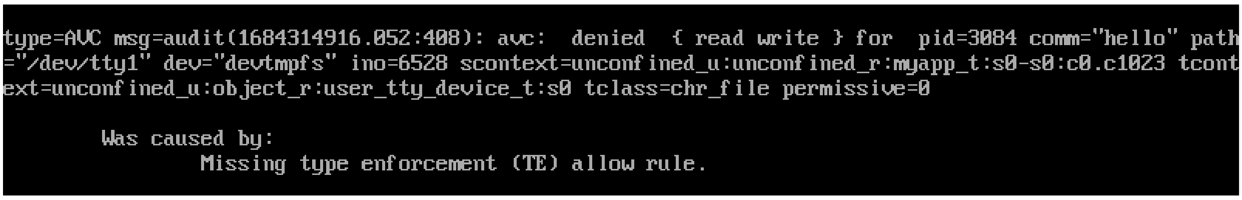


```
[root@localhost task2]# ./hello
-bash: ./hello: Permission denied
```

- 原因：缺少TE规则

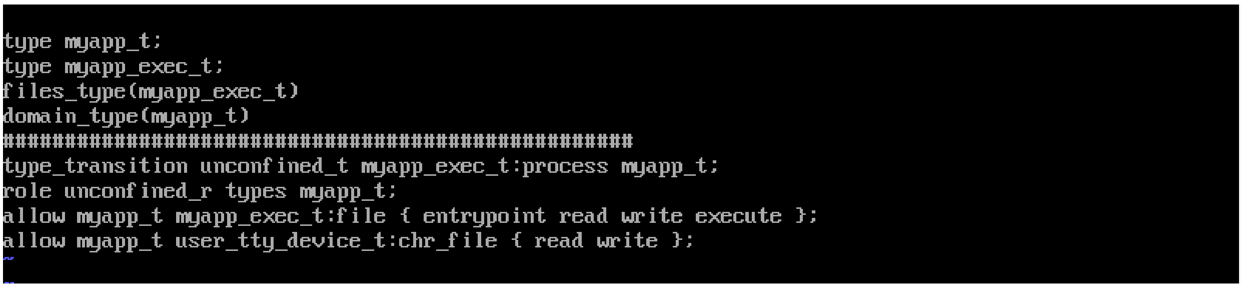


```
type=AVC msg=audit(1684315699.296:416): avc:  denied  { entrypoint } for  pid=3159 comm="bash" path=
"/home/task2/hello" dev="dm-0" ino=17682218 scontext=unconfined_u:unconfined_r:myapp_t:s0-s0:c0.c102
3 tcontext=unconfined_u:object_r:myapp_exec_t:s0 tclass=file permissive=0

        Was caused by:
                Missing type enforcement (TE) allow rule.
```



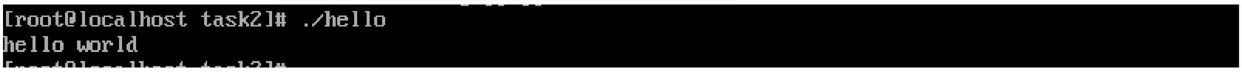
```
type=AVC msg=audit(1684314916.052:408): avc:  denied  { read write } for  pid=3084 comm="hello" path
="/dev/tty1" dev="devtmpfs" ino=6528 scontext=unconfined_u:unconfined_r:myapp_t:s0-s0:c0.c1023 tcont
ext=unconfined_u:object_r:user_tty_device_t:s0 tclass=chr_file permissive=0

        Was caused by:
                Missing type enforcement (TE) allow rule.
```

- 新的myapp.te文件



```
type myapp_t;
type myapp_exec_t;
files_type(myapp_exec_t)
domain_type(myapp_t)
####################################################
type_transition unconfined_t myapp_exec_t:process myapp_t;
role unconfined_r types myapp_t;
allow myapp_t myapp_exec_t:file { entrypoint read write execute };
allow myapp_t user_tty_device_t:chr_file { read write };
~
```

- 编译并加载策略`make semodule -i myapp.pp`，执行hello成功



```
[root@localhost task2]# ./hello
hello world
```