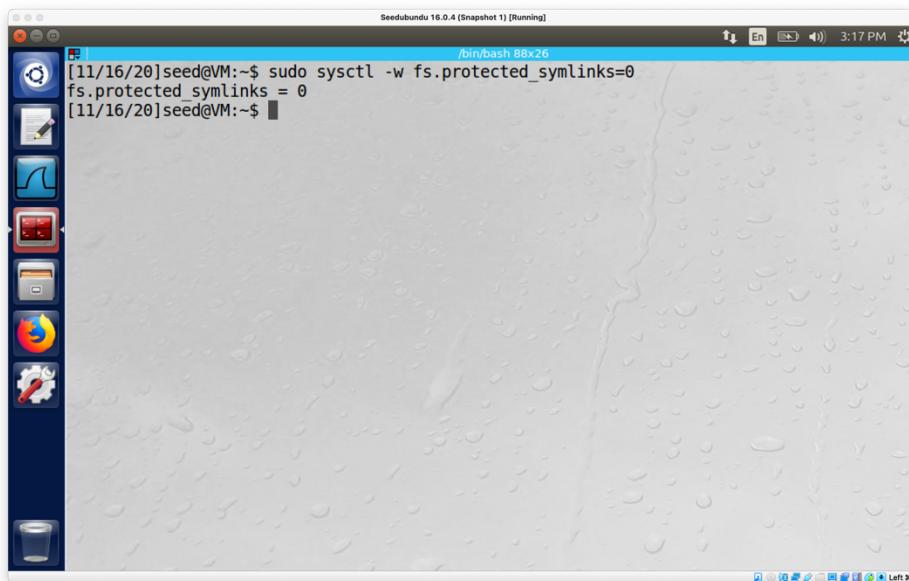


Man Tik Li
CS 377 - 001
November 16, 2020

Lab 4

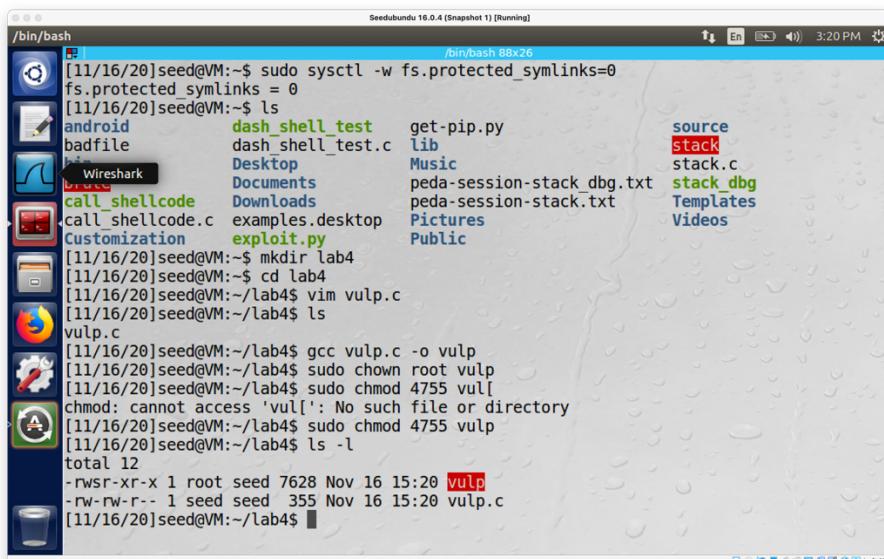
Initial Setup

On Ubuntu 16.04, use the following: \$ sudo sysctl -w fs.protected_symlinks=0



Created vulp.c in “lab4” folder. Set vulp to Set-UID program:

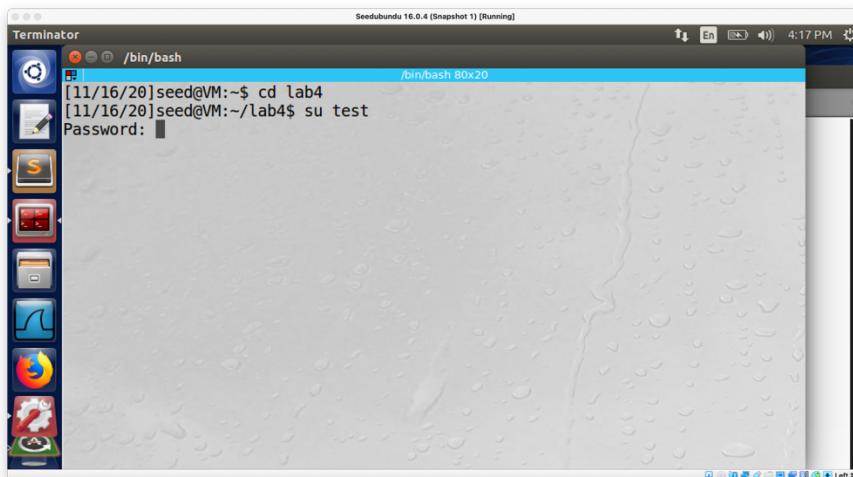
```
$ gcc vulp.c -o vulp
$ sudo chown root vulp
$ sudo chmod 4755 vulp
```



Task 1

After edited the passwd file, the test user appear in passwd.

```
root@VM:/etc# cat /etc/passwd | grep test
test:U6aMy0wojraho:0:0:test:/root:/bin/bash
root@VM:/etc#
```



After I hit enter, I can enter to the root without password.

```
[11/16/20]seed@VM:~/lab4$ su test
Password:
root@VM:/home/seed/lab4#
```

Task 2

```
[11/16/20]seed@VM:~/lab4$ vim attack_process.c
[11/16/20]seed@VM:~/lab4$ gcc -o attack_process attack_process.c
[11/16/20]seed@VM:~/lab4$ ./attack_process
bash: ./attack_process: No such file or directory
[11/16/20]seed@VM:~/lab4$ ./attack_process
```

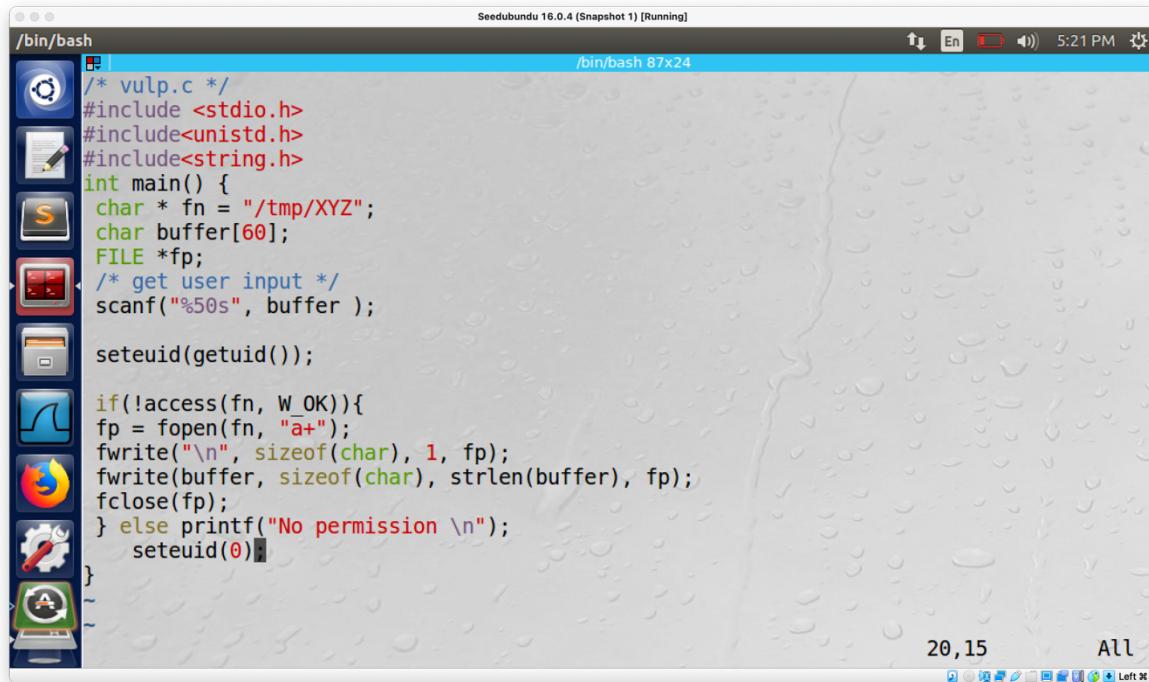
The screenshot shows a terminal window titled "Terminator" running on "Seedubuntu 16.0.4 (Snapshot 1) [Running]". The terminal window has a blue header bar with the title and some system icons. The main area of the terminal shows a series of failed login attempts. The user "attack" is attempting to log in with various passwords, all of which fail with the message "No permission". The terminal ends with a message: "STOP... The passwd file has been changed". The terminal window is set to a resolution of 80x20.

```
[11/16]No permission  
[11/16]No permission  
^C No permission  
[11/16]No permission  
attack No permission  
[11/16]No permission  
[11/16]No permission  
[11/16]No permission  
.attac No permission  
[11/16]No permission  
 No permission  
 STOP... The passwd file has been changed  
[11/16/20]seed@VM:~/lab4$
```

After running the attack_process and target_process.sh. The program successfully changed the passwd file.

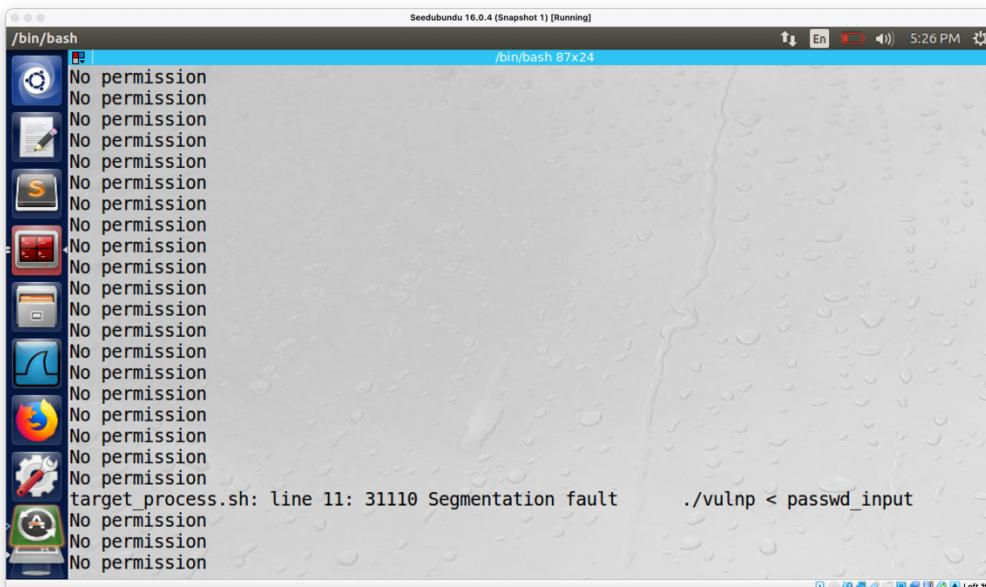
```
[11/16/20]seed@VM:~/lab4$ cat /etc/passwd | grep test  
rambo:U6aMy0wojraho:0:0:test:/root:/bin/bash
```

Task 3



```
/* 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 );
    seteuid(getuid());
    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(0);
}
```

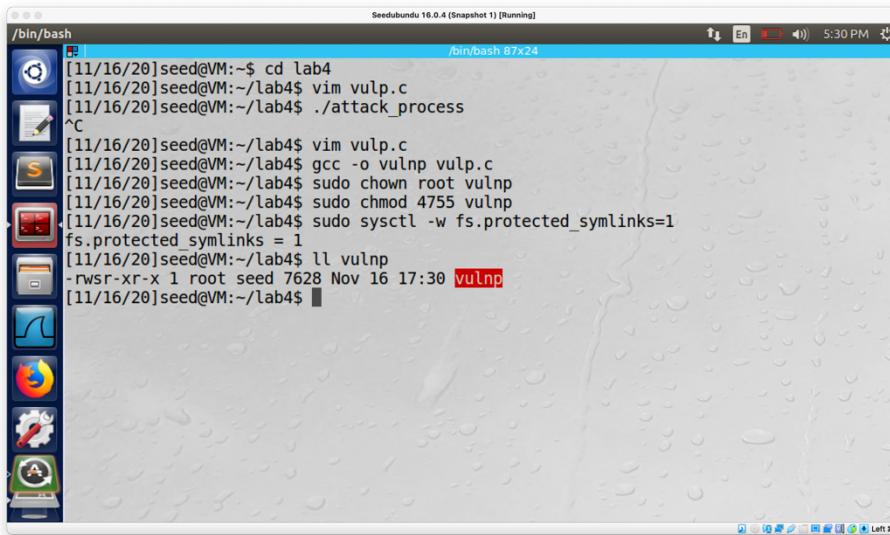
Edited the vulnerable program by use the real UID and effective UID of the program. We set the effective UID to be the same as real UID. Then it drops the privileges and at the end of the program we change the effective UID to be the same as before to gain the privileges back.



```
No permission
target_process.sh: line 11: 31110 Segmentation fault      ./vulnp < passwd_input
No permission
No permission
No permission
```

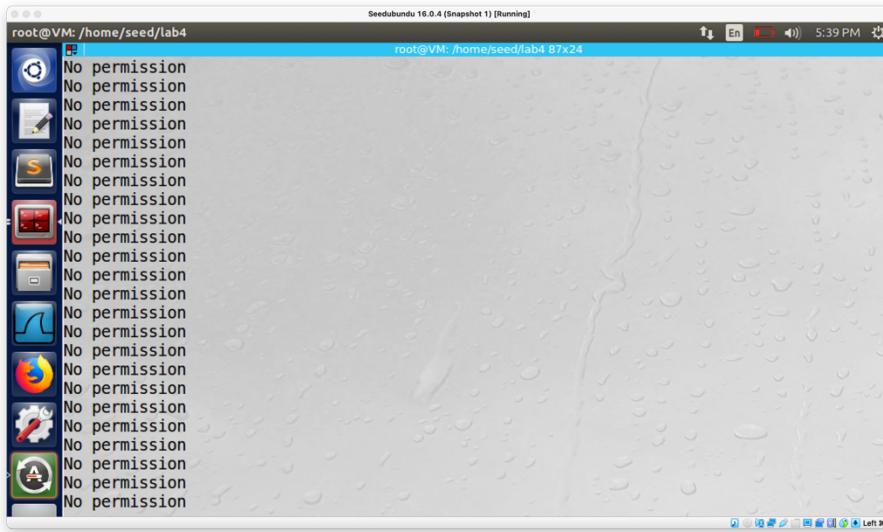
Result: The attack is not successful. I received segmentation fault, because the seteuid to currently limit the privilege. I can't use normal user to edit root permission files, which results in segment fault.

Task 4



A screenshot of a terminal window titled "Seedubuntu 16.0.4 (Snapshot 1) [Running] /bin/bash". The terminal shows a user session with the following commands and output:

```
[11/16/20]seed@VM:~$ cd lab4
[11/16/20]seed@VM:~/lab4$ vim vulp.c
[11/16/20]seed@VM:~/lab4$ ./attack_process
^C
[11/16/20]seed@VM:~/lab4$ vim vulp.c
[11/16/20]seed@VM:~/lab4$ gcc -o vulnp vulp.c
[11/16/20]seed@VM:~/lab4$ sudo chown root vulnp
[11/16/20]seed@VM:~/lab4$ sudo chmod 4755 vulnp
[11/16/20]seed@VM:~/lab4$ sudo sysctl -w fs.protected_symlinks=1
fs.protected_symlinks = 1
[11/16/20]seed@VM:~/lab4$ ll vulnp
-rwsr-xr-x 1 root seed 7628 Nov 16 17:30 vulnp
[11/16/20]seed@VM:~/lab4$
```



A screenshot of a terminal window titled "Seedubuntu 16.0.4 (Snapshot 1) [Running] root@VM: /home/seed/lab4 /bin/bash". The terminal shows a root session with the following command and output:

```
No permission
```

The attack is not successful, I received no permission in the output and the program keeps on running in a loop for a while.

1. Here does this protection scheme work?

The race condition vulnerability exploited involved symbolic link inside the '/tmp' directory. Therefore, preventing programs from following symbolic links under specific conditions might overcome this vulnerability. In this lab, since the program ran with root privilege and the /tmp directory is also owned by the root, the program will not be allowed to follow any symbolic link that is not created by the root. Also, the symbolic link was created by the attacker which was actually the seed account. Hence, this link will not be followed, and that lead to a crash.

2. What are the limitations of this scheme?

The scheme does not stop the race condition from taking place but hinders it from causing damages. Therefore, it is a good mechanism for access control that allowed only the intended users to access the file but did not really pause the race condition from taking place.