**Computer Security** **Name:** Ameer Hamza
**Assignment#2: Race Condition Vulnerability Lab**

1. **Environment Setup**

   (a) **Turning off countermeasures**: As the initial task, I turned off the Unix built-in protection against race condition attacks. The protection does not allow symbolic links to be followed in the world-writable directories (like `/tmp`), if the symlink owner does not match the follower (of the symlink) and the directory owner. This functionality has been disabled, so it is easy to follow symlinks.

```
[03/01/23]seed@VM:~/lab2_race_condition$ sudo sysctl
-w fs.protected_symlinks=0
fs.protected_symlinks = 0
[03/01/23]seed@VM:~/lab2_race_condition$ sudo sysctl
fs.protected_regular=0
fs.protected_regular = 0
[03/01/23]seed@VM:~/lab2_race_condition$
```

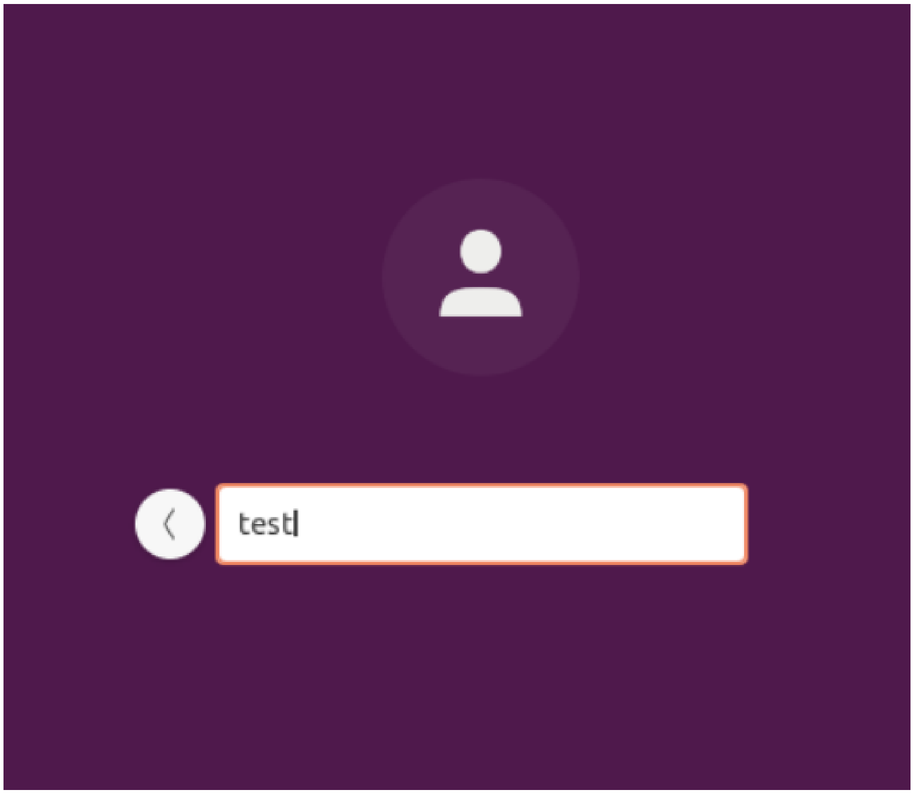
   (b) **Setup the `SET-UID` program**: I compiled the *vulp.c* program, and then it a `SET-UID` program. Now the user (`seed`) can run the program with root privileges.

```
[03/01/23]seed@VM:~/.../Labsetup$ gcc vulp.c -o vulp
[03/01/23]seed@VM:~/.../Labsetup$ ls -al vulp
-rwxrwxr-x 1 seed seed 17104 Mar  1 15:41 vulp
[03/01/23]seed@VM:~/.../Labsetup$ sudo chown root vul
p
[03/01/23]seed@VM:~/.../Labsetup$ ls -al vulp
-rwxrwxr-x 1 root seed 17104 Mar  1 15:41 vulp
[03/01/23]seed@VM:~/.../Labsetup$ sudo chmod 4755 vul
p
[03/01/23]seed@VM:~/.../Labsetup$ ls -al vulp
-rwsr-xr-x 1 root seed 17104 Mar  1 15:41 vulp
[03/01/23]seed@VM:~/.../Labsetup$
```

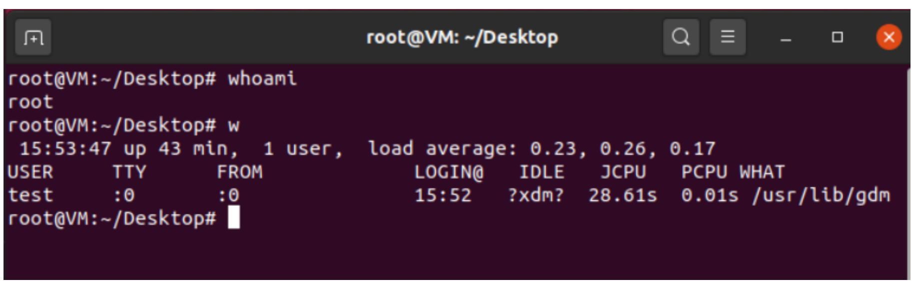
2. **Task 1: Choosing our target** We choose `/etc/passwd` as the target file. In this task, I simply opened the `/etc/passwd` as a superuser and entered a new user `test`, with hash of the password corresponding to the empty password. Below, I show the last line for the password file after modifying.

```
[03/01/23]seed@VM:~/Desktop$ tail -1 /etc/passwd
test:U6aMy0wojraho:0:0:test:/root:/bin/bash
```

   I logout of the `seed` account and try logging in as `test` user. Unfortunately, there is no way to show that I enter an empty password except for a video recording, which is not feasible.

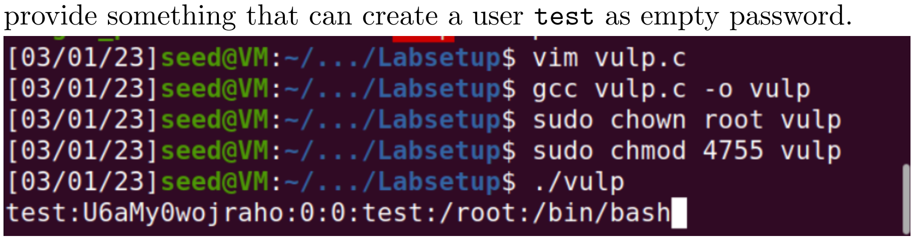



I check the account details. It shows that the `test` user is a root user. I later realized that an easy way was to user command `su test`.

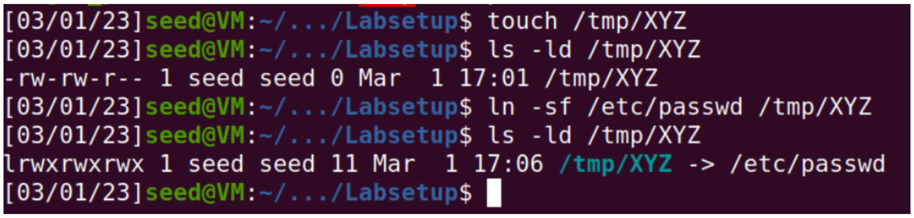



3. **Task 2: Launching the Race Condition Attack**

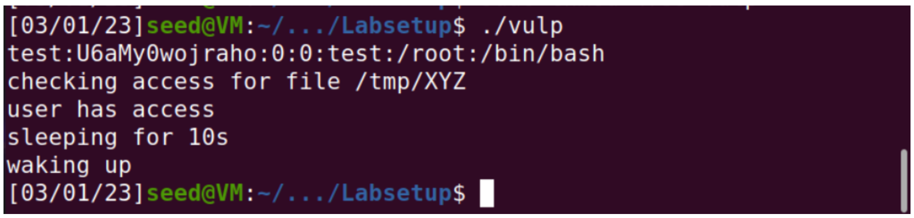
   (a) **Simulating a Slow Machine**: This part simulates a slow machine, where `vulp.c` program does into the sleep for 10s. Please find `4.1_vulp.c`. I compiled, changed it to `SET-UID` program, and ran it. It requires the line to be added to the file `/tmp/XYZ`, so I provide something that can create a user `test` as empty password.



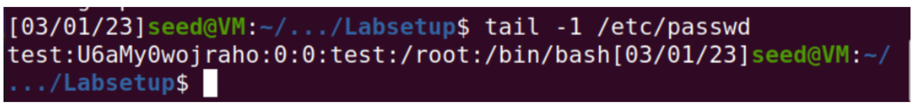

   I had already created a `/tmp/XYZ` file before running `vulp`, which can be seen in the screenshot below. I input the vulnerable string and 10 seconds counter starts. Meanwhile, I create a symbolic link from `/tmp/XYZ` to `/etc/passwd`.



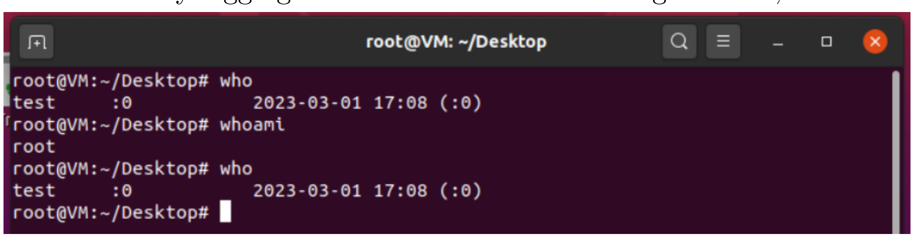

Once 10 seconds time ends, the program ends. I have printed extra info for debugging purposes.




I check the last line of `/etc/passwd` to verify that the attack was successful.

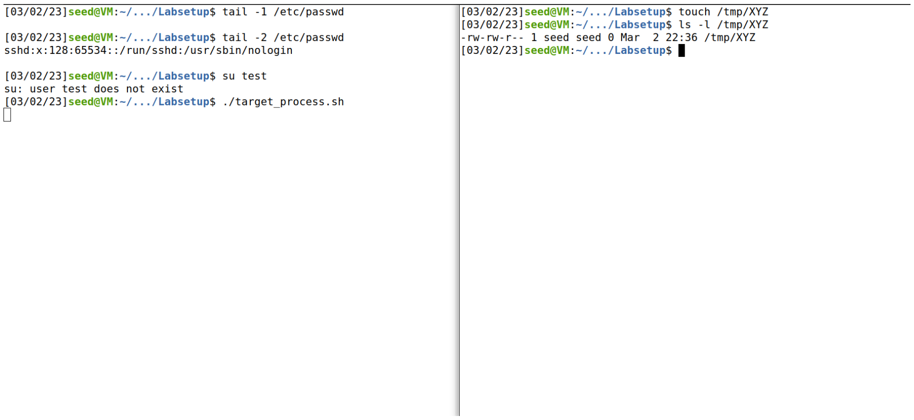



I also check by logging in as `test` user and checking the info, as shown below.




(b) **The Real Attack**:

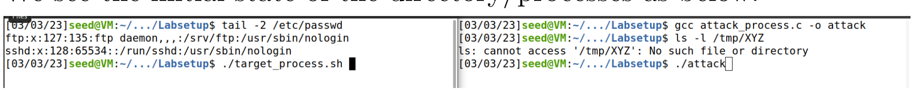
I removed the sleep command (check `4.2_vulp.c`), and created two scripts, one is vulnerable program script (`4.2_target_process.sh` - already given) and other is attack script (`4.2_attack_process.sh`). In the attack script, in the infinite `while` loop, I first remove the `/tmp/XYZ` file (`unlink` also works), and then create `/tmp/XYZ`. Then I unlink it and link again (`ln -sf`). The reason behind first removing and creating again is that `/tmp/XYZ` will be created as a simple file, and hopefully it will pass the check whether the user has `access` to the file. However, then we need to `unlink` it and `link` the file to `/etc/passwd` in order to be able to create a new line in `/etc/passwd`. Below shoes initial state of the directory/processes.




Once we run the attack process, the attack happens and we create a `test` user, as shown.


(c) **An Improved Attack Method**:

We have an improved attack using `renameat2` syscall, which is an atomic operation instead of using `unlink` and `link`. Find the vulnerable program script (`4.3_target_process.sh`), attack program (`4.3_attack_process.c`), and the usual vulnerable program (`4.3_vulp.c`). We see the initial state of the directory/processes as below.




After attack happens, we see that `/etc/passwd` gets changed and we get a user `test` as root user.

```
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
access granted
STOP... The passwd file has been changed
[03/03/23]seed@VM:~/.../Labsetup$ tail -3 /etc/passwd
sshd:x:128:65534::/run/sshd:/usr/sbin/nologin

test:U6aMy0wojraho:0:0:test:/root:/bin/bash[03/03/23]seed@VM:~/.../Lab
setup$
[03/03/23]seed@VM:~/.../Labsetup$
[03/03/23]seed@VM:~/.../Labsetup$
[03/03/23]seed@VM:~/.../Labsetup$
[03/03/23]seed@VM:~/.../Labsetup$ su test
Password:
root@VM:/home/seed/lab2_race_condition/Labsetup#
```

4. **Task 3: Countermeasures**

   (a) **Applying the Principle of Least Privilege**:

   As a countermeasure, we restrict the privileges of the user using `seteuid` syscall. We `seteuid(getuid())` to set the effective uid as that of users. This drops the extra privileges of the user. So, when the program tries to open the file `/tmp/XYZ`, which is in turn a link to `/etc/passwd`, the open fails because the effective uid is user, which should not have access to `/etc/passwd`. This corresponds to the line printed `Open failed: Permission denied` in the below screenshots. Later, we restore the privileges to root using `seteuid(geteuid())`.

   I try both normal attack and improved attack against the new program. For normal attack, see files: `5.1_vulp.c, 5.1_attack_process.sh, 5.1_target_process.sh`, and the screenshot below. (all scripts are the same as before, only `vulp.c` has been changed for this task).

```
access granted
Open failed: Permission denied
No permission
access granted
No permission
access granted
access granted
No permission
No permission
No permission
No permission
access granted
No permission
No permission
No permission
No permission
No permission
^C
[03/03/23]seed@VM:~/.../Labsetup$
```

   For improved attack, see files: `5.1_vulp.c, 5.1_attack_process.c, 5.1_target_process.sh`, and the screenshot below. (scripts are same as before)

```
access granted
Open failed: Permission denied
No permission
No permission
No permission
access granted
No permission
access granted
Open failed: Permission denied
^C
[03/03/23]seed@VM:~/.../Labsetup$
```

(b) **Using Ubuntu's Built-in Scheme**:

We turn on the Ubuntu's built-in protection. We try the experiment again with the normal attack and improved attack. The attack fails with `Open failed: Permission denied`.

The reason this time is that the simlink protection does not allow to follow a symbolic link if the owner of the symbolic link does not match the follower of the link (which is root, as the effective uid is root that has permission to write to `/etc/passwd`), and the director `/tmp` owner (which is again root). Hence, when the system tries to open the file (`/etc/passwd`), the permission is denied.

For normal attack, see files: `5.2_vulp.c, 5.2_attack_process.sh, 5.2_target_process.sh`, and the screenshot below. (scripts are the same as before).

```
[03/03/23]seed@VM:~/.../Labsetup$ ./target_process.sh > /dev/null
Open failed: Permission denied
O Screenshot led: Permission denied
Open failed: Permission denied
Open failed: Permission denied
Open failed: Permission denied
Open failed: Permission denied
Open failed: Permission denied
^C
[03/03/23]seed@VM:~/.../Labsetup$
```

```
[03/03/23]seed@VM:~/.../Labsetup$ ./attack_process.sh
```

For improved attack, see files: `5.2_vulp.c, 5.2_attack_process.c, 5.2_target_process.sh`, and the screenshot below. (scripts are same as before)

```
[03/03/23]seed@VM:~/.../Labsetup$ ./target_process.sh > /dev/null
Open failed: Permission denied
Open failed: Permission denied
Open failed: Permission denied
Open failed: Permission denied
Open failed: Permission denied
Open failed: Permission denied
Open failed: Permission denied
Open failed: Permission denied
^C
[03/03/23]seed@VM:~/.../Labsetup$
```

```
attempt 53904
attempt 53905
attempt 53906
attempt 53907
attempt 53908
attempt 53909
attempt 53910
attempt 53911
attempt 53912
attempt 53913
attempt 53914
attempt 53915
attempt 53916
attempt 53917
```

The simlink protection is not the ultimate solution. This protection only works for the world-writable sticky directories (like `/tmp`). This does not work for other directories and so the vulnerability can still be exploited in those directories.