**信息安全实验报告**

**Lab 8 Race Condition Vulnerability**

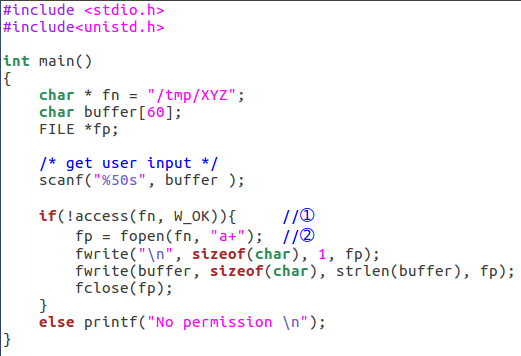
**孙铁**

**SA20225414**

实验开始之前，关闭保护机制：

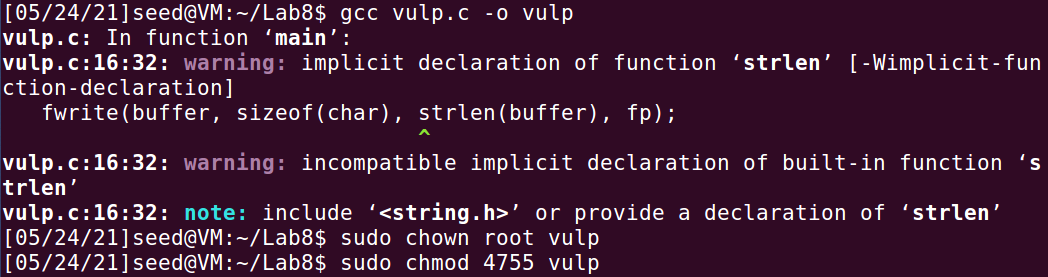


创建文件vulp.c：



代码作用是检查用户是否对文件/tmp/XYZ拥有写权限，如果有，则将其打开，并对此文件进行输入。可以看出 ➀ 与 ➁ 存在竟态条件漏洞。

编译并将其转化为root拥有的Set-UID程序：



**Task 1**

将存有密码的/etc/passwd文件作为攻击目标，在文件中添加一条记录，创建一个拥有root权限的用户test：

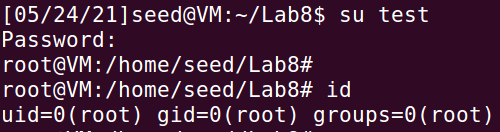
编辑/etc/passwd文件：



在/etc/passwd文件末尾加上如下字段：

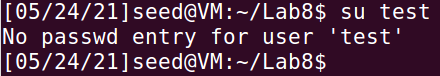


运行su test命令：



发现此时不需要输入密码即可登录到root权限用户test。

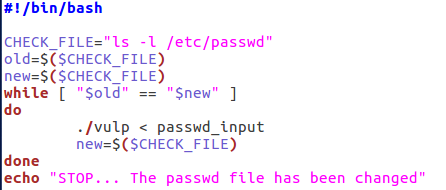
将/etc/passwd文件还原再次执行su test命令：



**Task 2**

**2.A**

创建文件target\_process.sh：

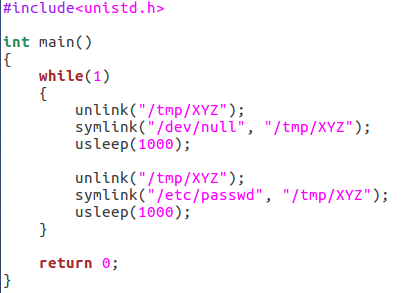


代码作用是不断运行Set-UID程序vulp，并从passwd\_input文件中读取内容作为输入。

创建文件passwd\_input：



创建文件attack\_process.c：

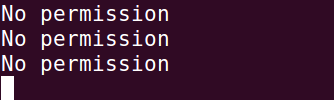


代码作用是改变/tmp/XYZ文件的指向，将/tmp/XYZ链接到任何人都可写的文件/dev/null上；然后删除链接，将/tmp/XYZ链接到目标文件/etc/passwd上。不断进行这两种链接操作来与目标进程竞争。

运行攻击程序attack\_process：

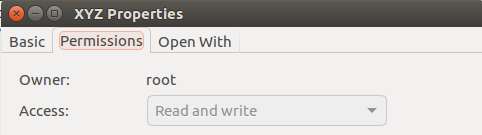


在另一个终端运行目标程序target\_process：



发现执行到一半会卡住。

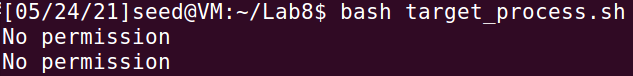
查看/tmp/XYZ属性，发现拥有者为root：



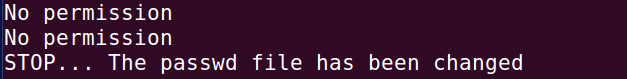
删除/tmp/XYZ：



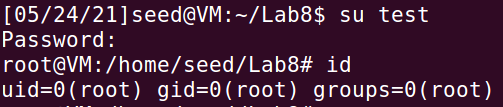
在另一个终端重新运行目标程序target\_process：



**…**



执行su test命令：

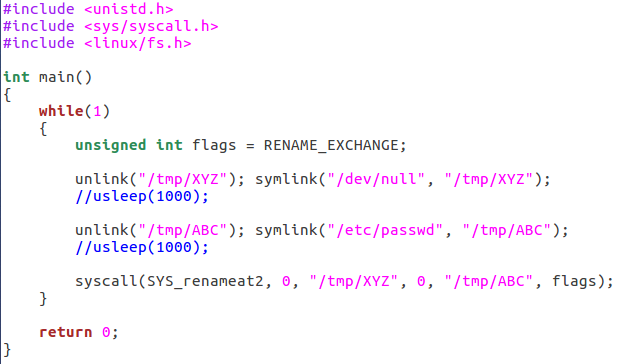


不需要输入密码即可登录到root权限用户test。

**2.B**

首先将被修改的/etc/passwd文件还原。

创建文件attack\_process2：



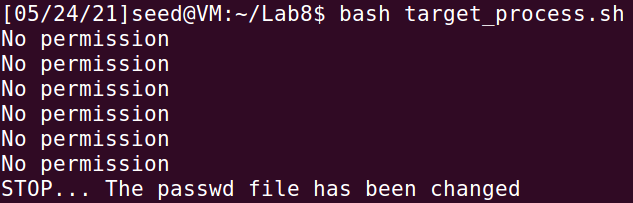
代码作用是将/tmp/XYZ链接到/dev/null，同时将/tmp/ABC链接到目标文件/etc/passwd，并交换/tmp/XYZ与/tmp/ABC的链接，不断执行上述操作来与目标进程竞争。

由于在Task 2.A中，偶尔会发生/tmp/XYZ文件所有者突然变为root的情况，这就会使得attack\_process无法改变/tmp/XYZ的链接，从而使得攻击无法成功。为了解决这种情况，我们使用能够交换两个链接的系统调用SYS\_renameat2来改变root所有的/tmp/XYZ的链接。

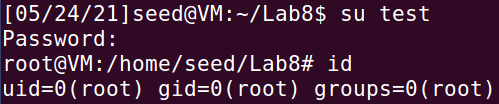
编译运行：



在另一个终端运行目标程序target\_process：



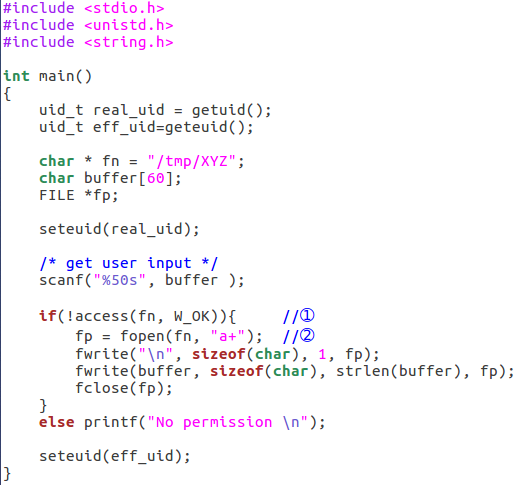
执行su test命令：



成功创建root权限用户test。

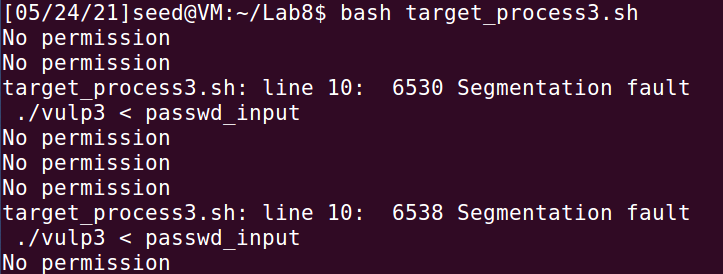
**Task 3**

创建文件vulp3.c：



在vulp.c的基础上加入了seteuid函数，让不需要root权限就可以正常运行的部分无法获得root权限。

将重新进行攻击：



攻击失败，出现段错误。

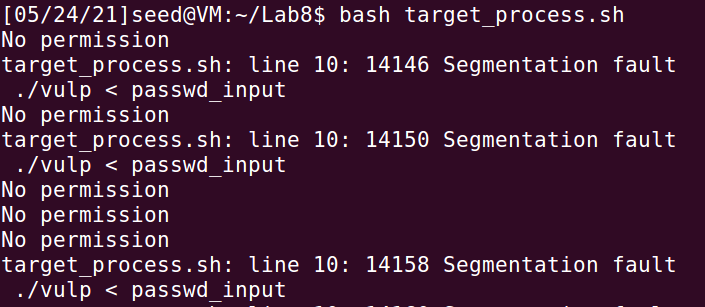
这是因为seteuid在open函数被调用前将root权限改为了普通seed用户权限，此时攻击程序即使赢得竟态条件也无法以seed用户权限调用open函数打开受保护的文件。系统发现seed用户试图访问受保护的文件，则会返回段错误信息。

**Task 4**

打开保护机制：



重新进行攻击：



攻击失败，出现段错误。

fs.protected\_symlinks用于限制普通用户建立软链接，当其为1时，只有文件的所有者、目录所有者以及root用户才能重命名或者删除这个目录中的文件。此时即时攻击者赢得竟态条件，也无法造成危害。

局限性在于，这种保护机制只适用于全局可写的粘滞目录，例如\tmp。