信息安全导论第三次实验报告

- 焦培淇　PB17151767

实验目的

- 将课堂上所讲的缓冲区溢出攻击知识融入实践。
- 了解现代操作系统应对缓冲区溢出攻击的措施。

实验内容

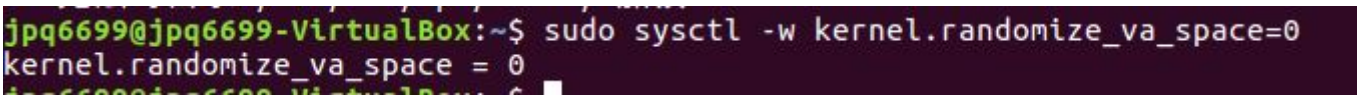
1. 初始化步骤：了解如何关闭操作系统地址虚拟化，gcc编译器的堆栈保护措施以及生成带有可执行栈的可执行文件。
2. 编译运行shellcode，确保shellcode运行正常。
3. 利用root用户编译stack.c文件，并修改文件权限。
4. 在关闭地址虚拟化的情况下，利用exploit.c文件生成badfile并通过stack可执行文件获取具有root权限的shell。
5. 打开地址虚拟化，再次尝试攻击，观察结果。
6. 开启gcc栈保护措施，再次尝试攻击。
7. 编译生成不带可执行栈的文件，再次尝试攻击。

实验步骤

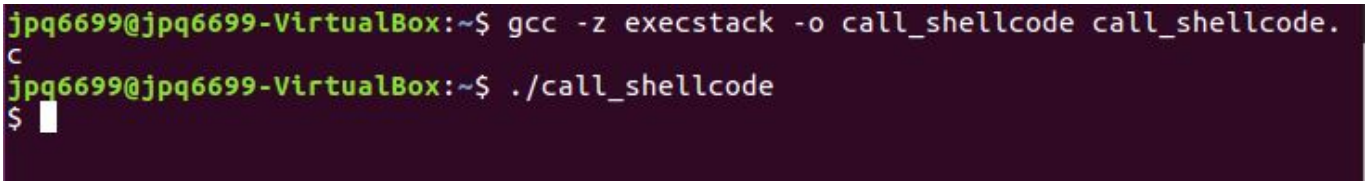
初始化

切换到root用户，运行以下命令，关闭系统地址随机化。

```
sysctl -w kernel.randomize_va_space=0
```



编译运行call_shellcode.c



可见成功获取shell

利用root用户编译stack.c文件并修改权限

```
jpq6699@jpq6699-VirtualBox:~$ su root
密码:
root@jpq6699-VirtualBox:/home/jpq6699# gcc -o stack -z execstack -fno-stack-prot
ector stack.c
root@jpq6699-VirtualBox:/home/jpq6699# chmod 4755 stack
root@jpq6699-VirtualBox:/home/jpq6699# exit
exit
jpq6699@jpq6699-VirtualBox:~$
```

利用exploit.c文件生成badfile，并运行stack以获取具有root权限的shell

```
jpq6699@jpq6699-VirtualBox:~$ gcc -o exploit exploit.c
jpq6699@jpq6699-VirtualBox:~$ ./exploit
492
jpq6699@jpq6699-VirtualBox:~$ ./stack
# u
# id
uid=1000(jpq6699) gid=1000(jpq6699) euid=0(root) groups=1000(jpq6699),4(adm),24(
cdrom),27(sudo),30(dip),46(plugdev),113(lpadmin),128(sambashare)
#
```

通过查看id可知，已经获取具有root权限的shell

开启地址虚拟化，再次尝试攻击

```
jpq6699@jpq6699-VirtualBox:~$ su root
密码:
root@jpq6699-VirtualBox:/home/jpq6699# sysctl -w kernel.randomize_va_space=2
kernel.randomize_va_space = 2
root@jpq6699-VirtualBox:/home/jpq6699# exit
exit
jpq6699@jpq6699-VirtualBox:~$ ./stack
段错误（核心已转储）
jpq6699@jpq6699-VirtualBox:~$ sh -c "while [ 1 ]; do ./stack; done;"
```

可见在开启地址虚拟化的情况下，攻击失败。

采用循环不断运行该程序，结果如下

```
jpq6699@jpq6699-VirtualBox:~$ sh -c "while [ 1 ]; do ./stack; done;"
Segmentation fault (core dumped)
Segmentation fault (core dumped)
Segmentation fault (core dumped)
Segmentation fault (core dumped)
Segmentation fault (core dumped)
Segmentation fault (core dumped)
Segmentation fault (core dumped)
Segmentation fault (core dumped)
Segmentation fault (core dumped)
Segmentation fault (core dumped)
Segmentation fault (core dumped)
Segmentation fault (core dumped)
Segmentation fault (core dumped)
Segmentation fault (core dumped)
```

可见在地址虚拟化的保护下，暴力的攻击取得成功的概率也比较小

开启gcc栈保护措施，再次尝试攻击

首先关闭地址随机化，并开启保护措施重新编译stack.c

```
jpq6699@jpq6699-VirtualBox:~$ su root
密码:
root@jpq6699-VirtualBox:/home/jpq6699# sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
root@jpq6699-VirtualBox:/home/jpq6699# gcc -o stack -z execstack  stack.c
root@jpq6699-VirtualBox:/home/jpq6699# chmod 4755 stack
root@jpq6699-VirtualBox:/home/jpq6699# exit
exit
```

再次运行./stack，结果如下：

```
jpq6699@jpq6699-VirtualBox:~$ ./stack
*** stack smashing detected ***: ./stack terminated
已放弃 (核心已转储)
```

可见报两个错误，第一个是因分配空间不足引起的"stack smashing detected"，第二个是段错误。

编译生成不带可执行栈的程序，再次尝试攻击

首先重新编译程序如下：

```
jpq6699@jpq6699-VirtualBox:~$ su root
密码:
root@jpq6699-VirtualBox:/home/jpq6699# gcc -o stack -z noexecstack -fno-stack-pr
otector stack.c
root@jpq6699-VirtualBox:/home/jpq6699# chmod 4755 stack
root@jpq6699-VirtualBox:/home/jpq6699# exit
exit
```

运行stack结果如下：

```
jpq6699@jpq6699-VirtualBox:~$ ./stack
段错误 (核心已转储)
```

可见报段错误信息。此处因为带有不可执行栈，因此shellcode部分并不在栈中，因此修改返回地址后并不能跳入shellcode执行，因此出现段错误。