第二次实验报告

**Buffer Overflow Vulnerability Lab**

**Return-to-libc Attack Lab**

实验人: 57117104 孙舒雯

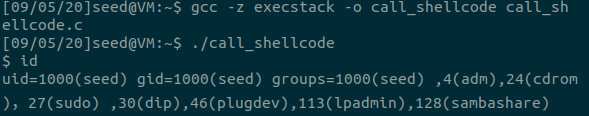
实验时间：9.03-9.06

**Buffer Overflow Vulnerability Lab**

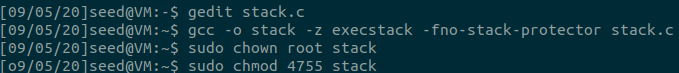
1. Task 1:Running Shellcode

实验过程：

运行一段 shellcode 以及具有缓冲区溢出漏洞的程序实验结果如图:



执行所给 call\_shellcode 代码，成功进入一个 shell 。

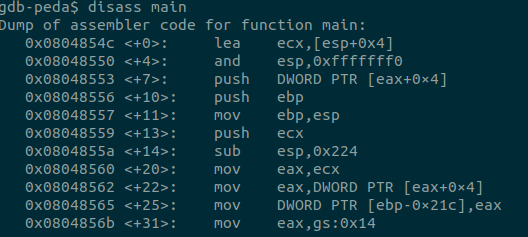


实验给出了一个有缓冲区溢出漏洞的程序 stack.c，利用这个漏洞获得 root权限，生成 stack 文件。

1. Task 2: Exploiting the Vulnerability

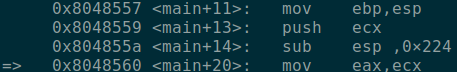
实验过程：获得 str 在内存中的地址，在 exploit.c 中，shellcode 保存在 exploit.c 的buffer 的某处；在 stack.c 中，将 exploit.c 中的 buffer 完整写入 stack.c 的 str 中。





在str 起始地址对应处设置断点

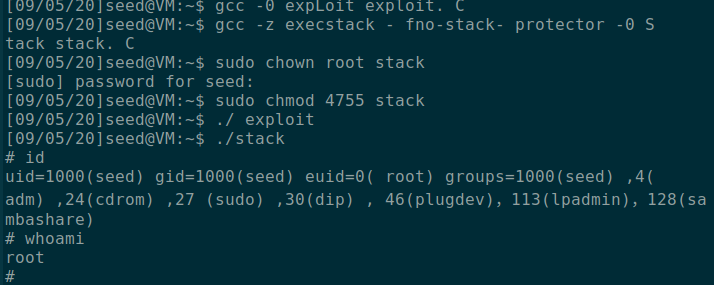




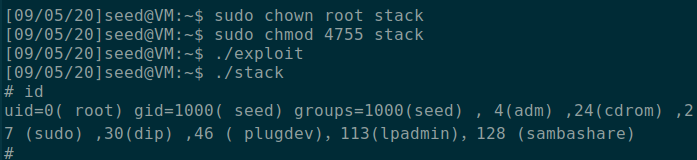


漏洞程序读取 badfile 文件到缓冲区 str。

参考手册构造 exploit.c，将 shellcode 的地址写到 stack.c 中的 bof 的返回地址处。

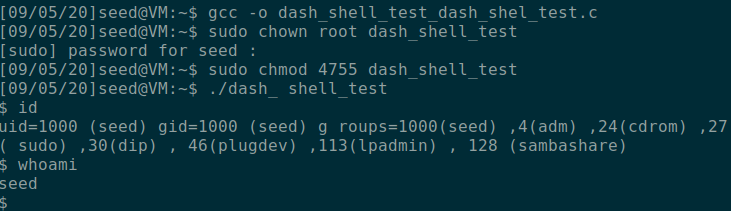


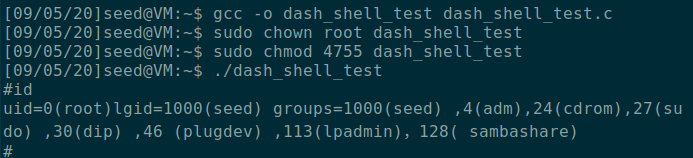
测试成功，然后修改上面的 uid 为 0



1. Task 3:Defeating dash’s Countermeasure

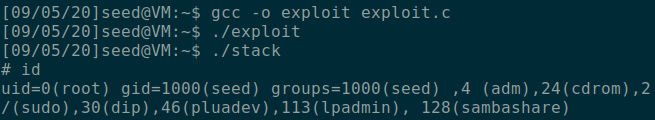
实验过程：





不设置 UID ，获取 shell 后不能获取 root 权限，设置 UID 后 Shellcode 可直接获取 root 权限。

更换后的 shellcode 在调用/bin/sh 之前就调用了 setuid(0)，获得了 root 权限，所以可以调用/bin/sh，绕过了dash shell 的防御机制。

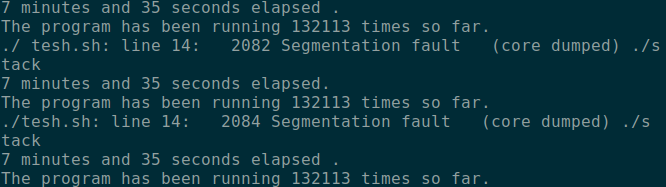


1. Task 4:Defeating Address Randomization

实验过程：

地址随机化：$ sudo /sbin/sysctl -w kernel.randomize\_va\_space=2

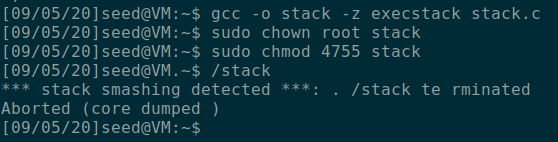
执行 task2，然后循环执行 shell 脚本，在执行 7 多分钟后，没有获得正确的地址，攻击不成功，说明有效地阻止了攻击者暴力穷举进行攻击。



1. Task 5: Turn on the Stack Guard Protection

实验过程：

关闭地址随机选项$ sudo sysctl -w kernel.randomize\_va\_space=0，在无-fno-stack-protector 选项的情况下编译程序，重复执行stack.c。



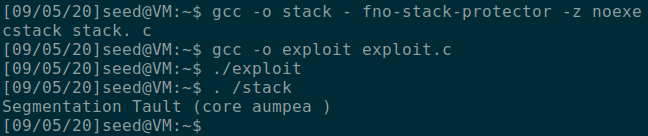
攻击不成功，程序崩溃，说明 linux 有栈保护机制。

1. Task 6: Turn on the Non-executable Stack Protection

实验过程：关闭地址随机选项。打开栈不可执行保护机制

$ gcc -o stack -fno-stack-protector -z noexecstack stack.c

重复 task2。不能成功，因为开启了栈不可执行机制，使得对栈的使用有边界限制。



**Return-to-libc Attack Lab**

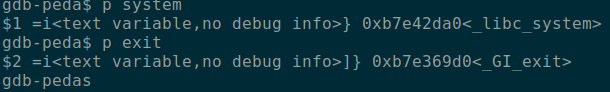
一、Task 1: Finding out the addresses of libc function

实验过程：通过gdb找到libc 库中system和exit函数地址，

seed@VM:~/lab/lab2 retlib$ gdb -q retlib

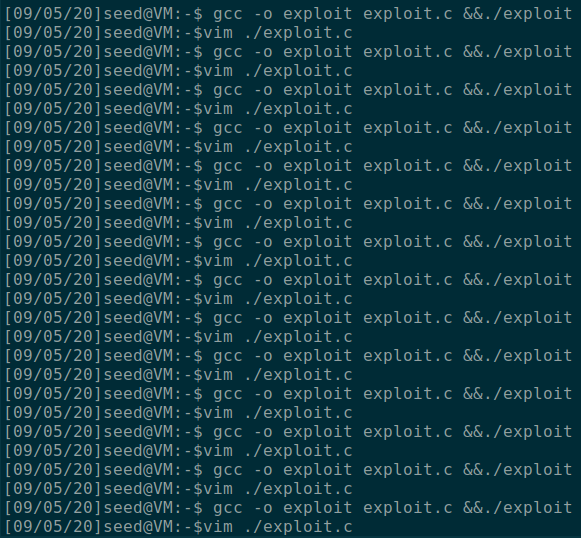
Reading symbols from retlib...(no debugging symbols found)...done.

gdb-pedas runSystem 函数的地址位于 0xb7e42da0,exit 函数的地址位于0xb7e369d0



二、Task 2: Putting the shell string in th

实验过程：将字符串/bin/sh 放入内存，创建环境变量 MYSHELL，取值为/bin/sh，反复穷举逼近目标地址。



三、Task 3: Exploiting the buffer-over flow vulnerability

实验过程：基于Task 1和Task 2，进行缓冲区溢出攻击

找到buffer位置，返回地址，exit地址，环境变量地址。

若改变exit地址，能提权但是会发生段错误，若改变system地址，不能正常提权，会发生参数错误，然后引发段错误。

四、Task4: Turning on address randomization

实验过程：开启地址随机化

$ sudo /sbin/sysctl -w kernel.randomize\_va\_space=2，

再次运行 retlib



发生了段错误。地址随机化导致原先寻找到的system地址无效化了。

1. Task5：Defeat Shell’s countermeasure

实验过程：找到setuid在内存中地址，改写exploit.c中buf，



Uid=0，shell取得root权限。成功绕过 dash shell 的保护机制。

1. Task6：Defeat Shell's countermeasure without putting zeros in input

实验过程：在buf中不填充0实现task5。