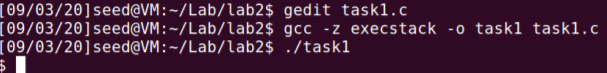
**Part1: Buffer Overflow Vulnerability Lab**

**57117223 杨克**

**Task 1: Running Shellcode**



运行结果是启动了一个新 shell

**Task 2: Exploiting the Vulnerability**

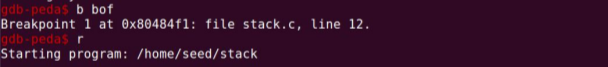
首先关闭地址随机化



使用默认 BUF SIZE，关闭 StackGuard 和不可执行的堆栈保护编译程序，并赋予其 root 用户 权限，并将其变为 SET-UID 程序



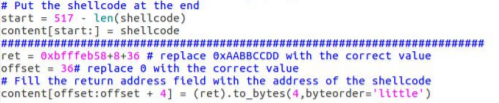
使用 gdb 工具进行调试：



得到偏移为 32+4=36：



在利用漏洞程序中设置相关参数



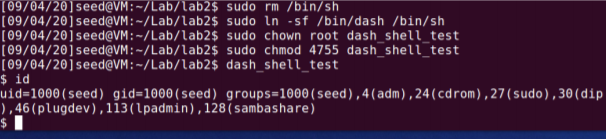
执行利用程序，最终成功获取了 root 权限：



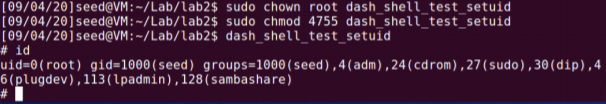
**Task 3: Defeating dash’s Countermeasure**

将所给代码保存为dash\_shell\_test.c

当取消注释所标代码时，编译执行结果如下，获取到了root权限



去掉注释：



在Task2的exploit.c中的shellcode增加以下四条指令：

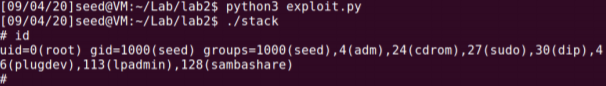
"\x31\xc0" /\* Line 1: xorl %eax,%eax \*/

"\x31\xdb" /\* Line 2: xorl %ebx,%ebx \*/

"\xb0\xd5" /\* Line 3: movb $0xd5,%al \*/

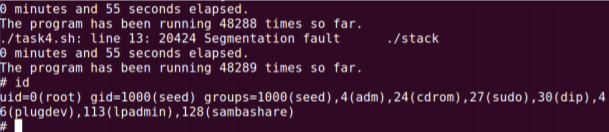
"\xcd\x80" /\* Line 4: int $0x80 \*/

重复2中攻击，利用setuid(0），成功获取root权限



**Task 4: Defeating Address Randomization**

执行结果如下

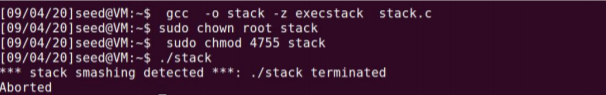


**Task 5: Turn on the StackGuard Protection**

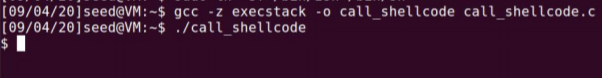
首先关闭地址随机化：



打开栈保护机制，重复 task2：

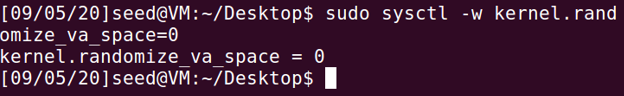


由于栈保护机制，攻击失败。 回到 task1：

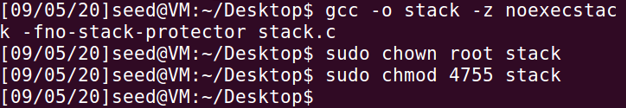


**Task 6: Turn on the Non-executable**

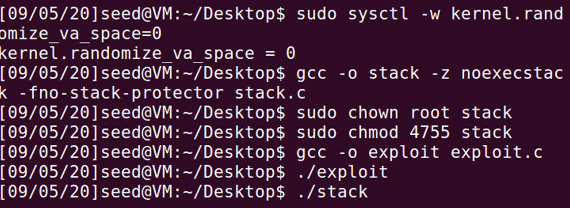
首先，关闭地址随机化，防止干扰



然后，关闭栈保护，使用栈不可执行选项重新编译易受攻击的程序stack.c



重复task2中的操作，得到结果如下

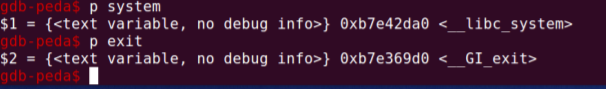


可以看出由于栈不可执行，攻击同样失败。

**Part2: Return-to-libc Attack Lab**

**Task 1: Finding out the addresses of libc functions**

查看 system 和 exit 函数的位置：

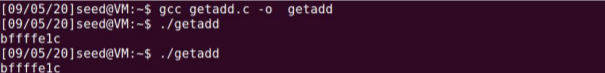


**Task 2: Putting the shell string in the memory**

定义一个新的 shell 变量：



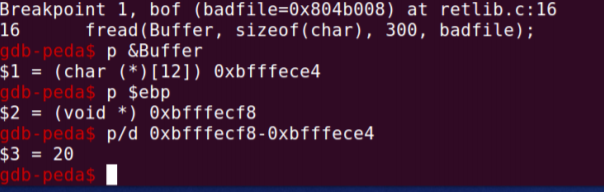
获得myshell的地址



可以看到，由于关闭了地址随机化，每次获得的地址都是相同的。

**Task 3: Exploiting the buffer-overflflow vulnerability**

首先仍然使用 gdb 调试器查看 retlib.c 中 bof 函数执行时，字符串 Buffer 头地址距离 ebp 的 长度。



可以看到距离为 20。

system()偏移为 20+4=24，即 Y=24：

计算得到 exit()偏移为 Z=20+8=28，shell 变量偏移为 X=20+12=32，并将地址填入空缺代码：

编译并运行 exploit2，执行 retlib：

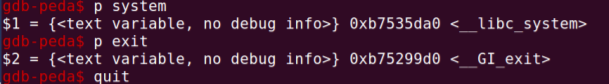


发现攻击成功，获取了 root 权限。

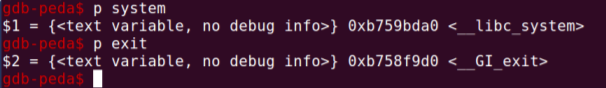
**Task 4: Turning on address randomization**

打开虚拟地址随机化，gdb 调试 retlibc 程序：

第一次：



第二次：



system 和 exit 函数的地址都发生了变化

查看环境变量：





综上，地址随机化不会改变偏移量 XYZ 的值，但变量地址、system 函数地址和 exit 函数地 址都会发生改变，因此攻击失败。

**Task 5: Defeat Shell’s countermeasure**