这题是非常经典的一道题，可以用两种方法来攻击

1. exit\_hook

之前听过malloc\_hook，free\_hook还有realloc\_hook就是没听过exit\_hook，本来就准备学FSOP的结果还学到了exit\_hook感觉赚到了

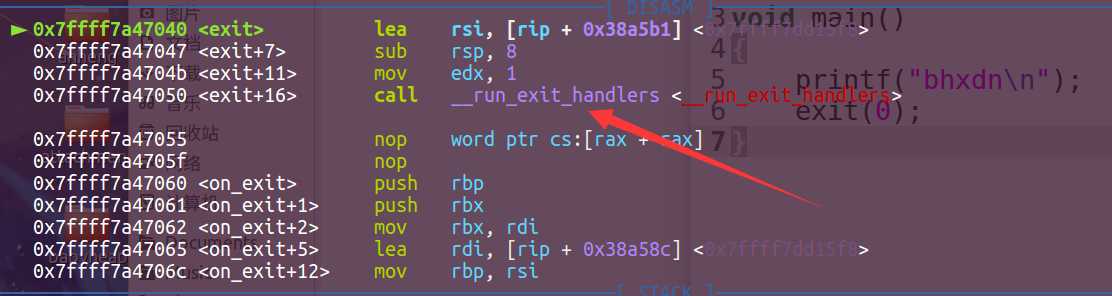
<https://www.cnblogs.com/bhxdn/p/14222558.html>

[不会修电脑](https://www.cnblogs.com/bhxdn/p/14222558.html" \t "https://www.cnblogs.com/bhxdn/p/_blank)师傅的博客讲解得非常清楚，主要是有找偏移的方法，记录下以后可以试试2.31的libc

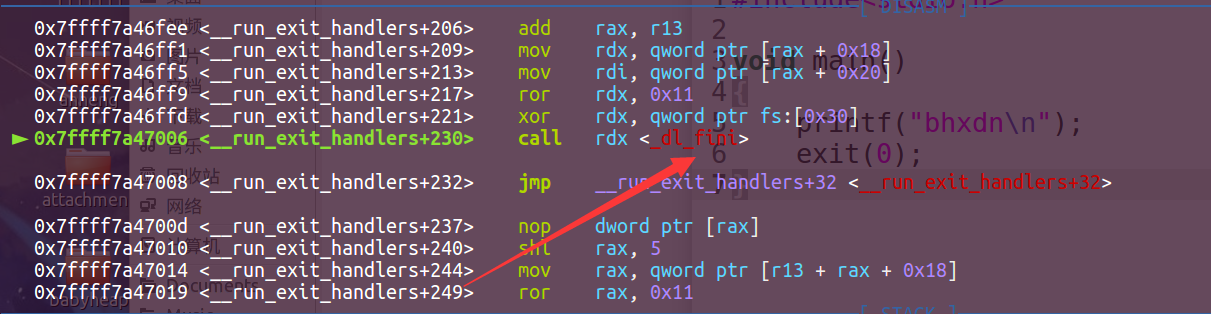
　首先我们自己写一个程序，调试一下exit是怎么执行的。

1 #include<stdio.h>2 3 void main()4 {5 printf("bhxdn\n");6 exit(0);7 }

　　在第六行下断点看一下。

[](https://img2020.cnblogs.com/blog/1941744/202101/1941744-20210102113406921-1225940756.png)

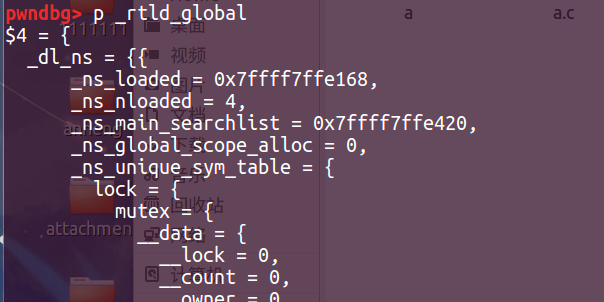
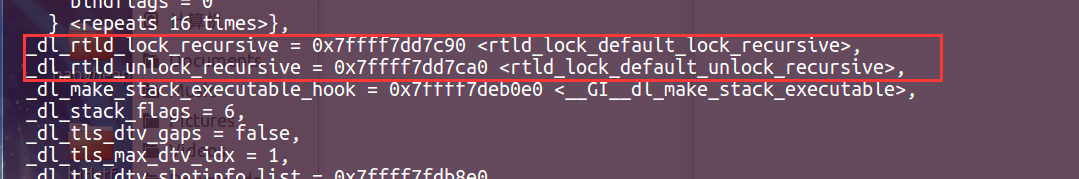
　　这里可以看到是执行了\_\_run\_exit\_handlers。进入这个函数，看看它调用了哪些函数。

[](https://img2020.cnblogs.com/blog/1941744/202101/1941744-20210102113714040-179657517.png)

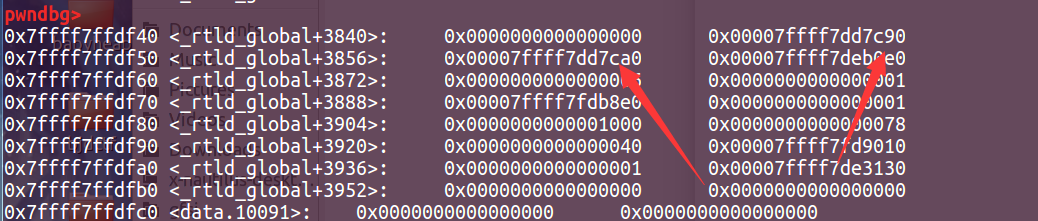
　　这里显示其中调用的一个函数，是\_dl\_fini。这里为了方便看，我们直接看\_dl\_fini的关键源码。

　　看8行和18行，发现是调用了 \_\_rtld\_lock\_lock\_recursive 和 \_\_rtld\_lock\_unlock\_recursive 。

　　这里我们看一下这两个函数在哪。

[](https://img2020.cnblogs.com/blog/1941744/202101/1941744-20210102115256979-1536370792.png)[](https://img2020.cnblogs.com/blog/1941744/202101/1941744-20210102115331046-615394032.png)

　　这两个函数在\_rtld\_global结构体里面。只要我们将其中一个指向one\_gadgets，在CTF中，就可以拿到shell了。

[](https://img2020.cnblogs.com/blog/1941744/202101/1941744-20210102115841026-204097627.png)　　接下来就是计算偏移了，发现在64位libc-2.23中，这两个指针在结构体中的偏移分别是3848和3856。为了方便，我们可以直接记住这两个指针和libc的基地址之间的距离。

**在libc-2.23中**  
**exit\_hook = libc\_base+0x5f0040+3848**

**exit\_hook = libc\_base+0x5f0040+3856**

**在libc-2.27中**

**exit\_hook = libc\_base+0x619060+3840**

**exit\_hook = libc\_base+0x619060+3848**

这样一来，只要知道libc版本和任意地址的写，我们可以直接写这个指针，执行exit后就可以拿到shell了。（其实不用非要执行exit，就程序正常返回也可以执行到这里）

1. FILE结构——FSOP

在exp里面记录了需要覆盖的关键位置的偏移

FSOP 的核心思想就是劫持 \_IO\_list\_all 的值来伪造链表和其中的 \_IO\_FILE 项，但是单纯的伪造只是构造了数据还需要某种方法进行触发。FSOP 选择的触发方法是调用 \_IO\_flush\_all\_lockp，这个函数会刷新 \_IO\_list\_all 链表中所有项的文件流，相当于对每个 FILE 调用 fflush，也对应着会调用 \_IO\_FILE\_plus.vtable 中的 \_IO\_overflow。

FSOP 利用的条件：泄露 libc.so 基址，因为 \_IO\_list\_all 是作为全局变量储存在 libc.so 中的；用任意地址写把 \_IO\_list\_all 改为指向可控内存的地址；伪造 \_IO\_FILE\_plus 结构体。伪造结构体需要 bypass 这些 check ：

也就是

* fp->\_mode <= 0
* fp->\_IO\_write\_ptr > fp->\_IO\_write\_base

而 \_IO\_flush\_all\_lockp 不需要攻击者手动调用，在一些情况下这个函数会被系统调用：

1. 当 libc 执行 abort 流程时
2. 当执行 exit 函数时
3. 当执行流从 main 函数返回时

这个题目的这个利用方法是劫持vtable，伪造vtable，改IO\_overflow为system，在io\_file的第一个位置上面写上bin/sh。当调用虚表函数时，他是第一个参数。

最后从平凡之路（博客）上截图三张更容易理解exp如何构建



