目录

[环境搭建 1](#_Toc151021349)

[ret2text 1](#_Toc151021350)

[ret2shellcode 3](#_Toc151021351)

[ret2syscall 4](#_Toc151021352)

[ret2libc1 5](#_Toc151021353)

[ret2libc2 6](#_Toc151021354)

[ret2libc3 7](#_Toc151021355)

[中级 ROP：2016 XDCTF pwn100 8](#_Toc151021356)

环境搭建

**1.操作系统**：Ubuntu20.04

**2.安装32位环境以及python3**：

sudo apt-get update

sudo apt-get install -y lib32z1 libc6-dev-i386 lib32readline6-dev

wget https://www.python.org/ftp/python/3.7.1/Python-3.7.1.tgz

tar xzvf Python-3.7.1.tgz

sudo apt-get install build-essential libncursesw5-dev libgdbm-dev libc6-dev zlib1g-dev libsqlite3-dev tk-dev libssl-dev openssl libffi-dev libbz2-dev -y

cd ~/ Python-3.7.1

./configure

sudo make && sudo make install

**3.修改了sh**：

sudo apt install zsh

sudo su

cd /bin

rm sh

ln -s zsh sh

exit

**4.安装了反汇编工具IDA**：

用ubutun中的Firefox浏览器找到：https://www.hex-rays.com/products/ida/support/download\_freeware/

下载安装包，安装。

需要qt的运行环境：

sudo apt install libxcb-xinerama0

**5.安装pwntools工具**：

sudo apt-get install python3-pip

sudo apt-get install python3-dev libffi-dev build-essential（安装依赖项）

pip3 install pwntools

python3 -c "import pwn; print(pwn)"（验证）

**6.安装gef调试工具**：

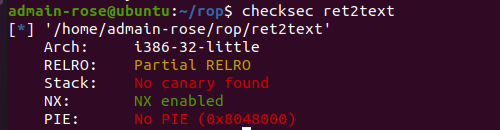
https://github.com/hugsy/gef

$ wget -O ~/.gdbinit-gef.py -q https://gef.blah.cat/py

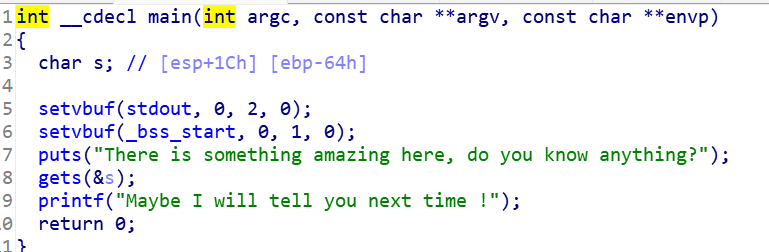
$ echo source ~/.gdbinit-gef.py >> ~/.gdbinit

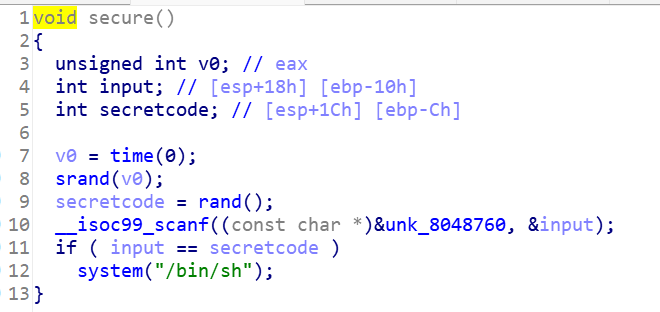
ret2text

首先检查程序的保护机制：

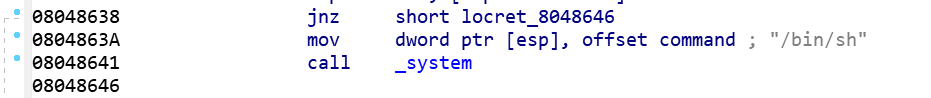


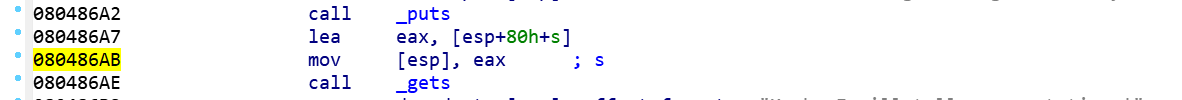
可得出是32位程序，开启了NX，然后查看源代码：

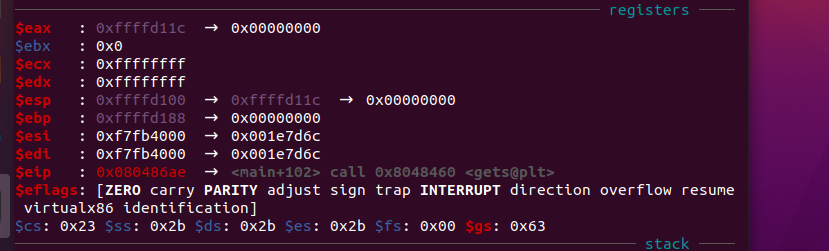




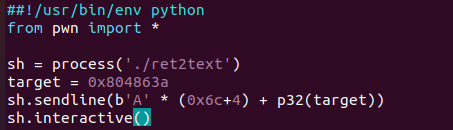
主函数中有gets函数可以利用，secure函数中有system（“/bin/sh”）可以利用。只要覆盖主函数返回地址，让其指向system（“/bin/sh”）就可以攻击成功。



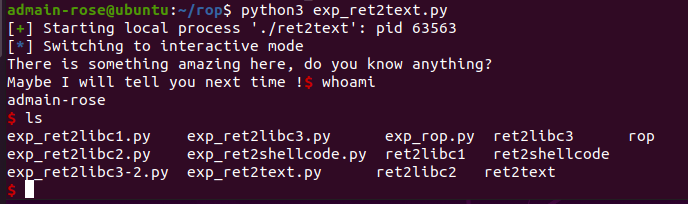




通过查看IDA，gef设置断点，查看寄存器内容。计算从get到main函数的返回地址的字节数。最后构造payload如下：

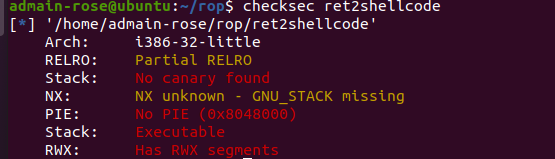


攻击效果如下：

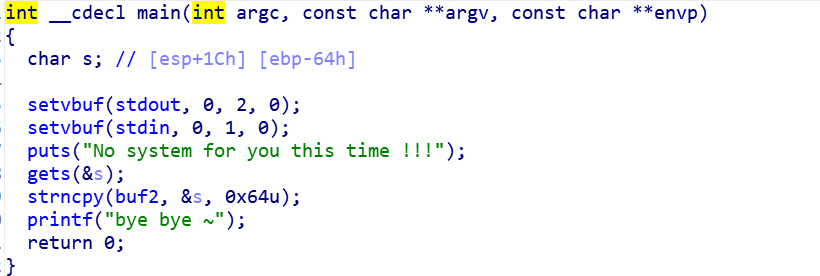


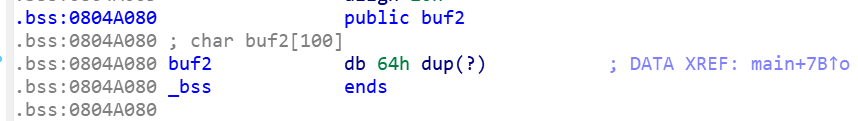
ret2shellcode

首先检测程序开启的保护:



可看到，程序有可读可写可执行段。然后用IDA查看程序：

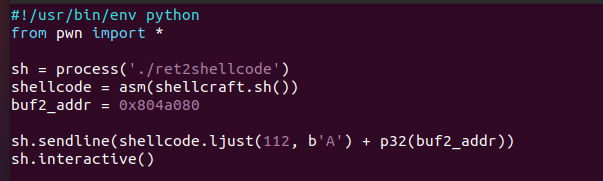




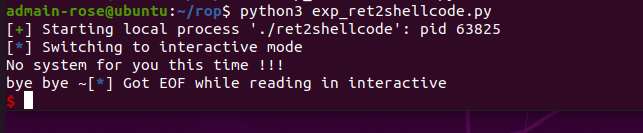


有get函数可以利用，其次，buf2被存到了.bss区域，该区域有读写执行的权限，只要把shellcode写入这个区域，并控制main返回地址指向该位置即可攻击成功。

构造的payload如下：

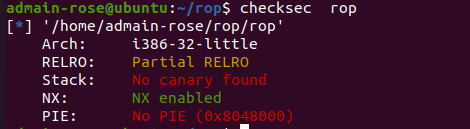


攻击效果如下：

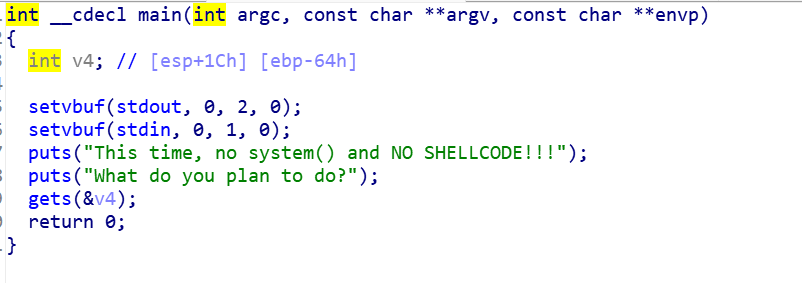


ret2syscall

首先检测程序开启的保护：



可以看到程序开启了NX，然后看源代码：



可以看到有get函数，可以利用来覆盖main返回地址。除此之外没有可以直接利用的代码。但是可以把获取shell的系统调用参数放到对应的寄存器中，再执行int 0x80就可以执行对应的系统调用。

比如execve("/bin/sh",NULL,NULL)对应为：

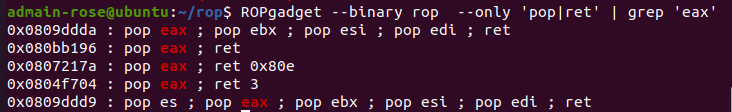
系统调用号，即 eax 应该为 0xb

第一个参数，即 ebx 应该指向 /bin/sh 的地址，其实执行 sh 的地址也可以。

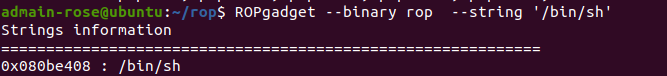
第二个参数，即 ecx 应该为 0

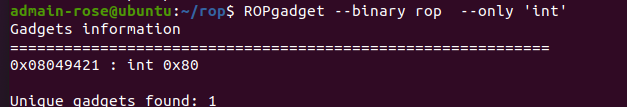
第三个参数，即 edx 应该为 0

然后寻找控制eax的gadgets：

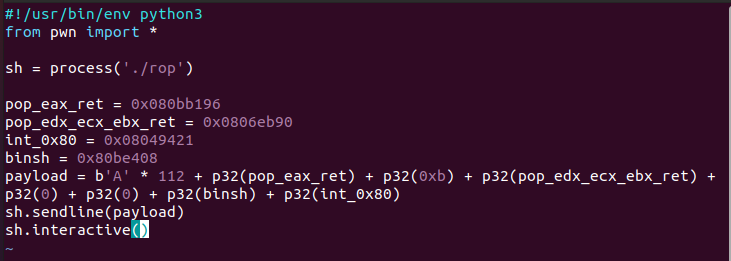


如上图，使用ROPgadget --binary rop --only 'pop|ret' | grep 'eax'命令，修改eax字段来寻找剩余的寄存器的gadets。然后还需要找到/bin/sh和int 0x80对应的地址：

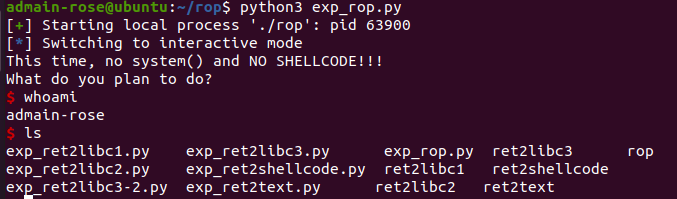




最后构造的payload如下：

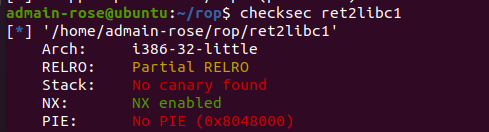


攻击效果如下：

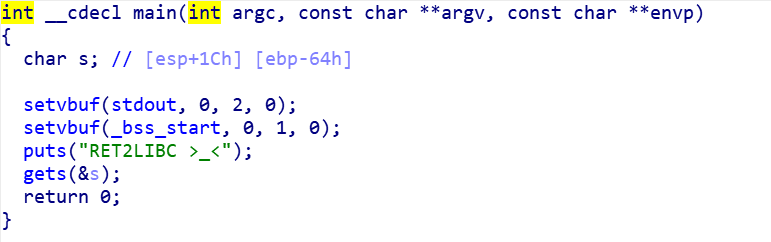


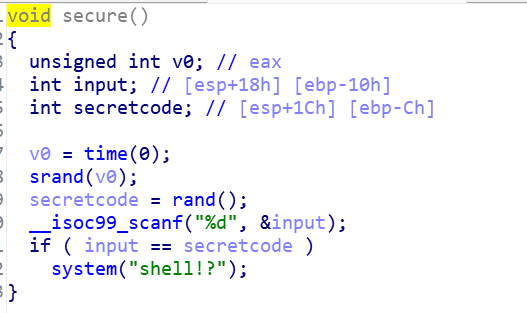
ret2libc1

首先检测一下程序的安全保护：

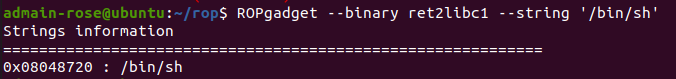


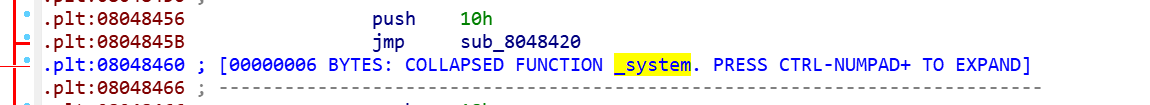
然后IDA查看源程序：



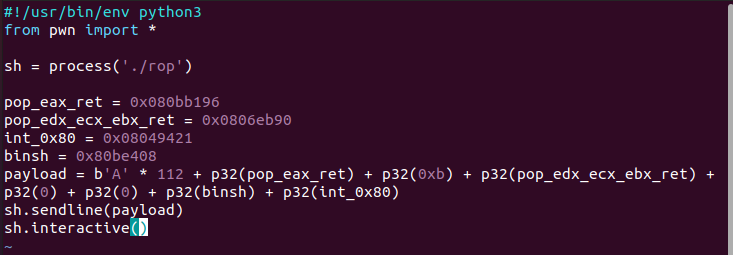


可见，main函数中有get可以利用，secure里有system可以利用，还缺一个/bin/sh：

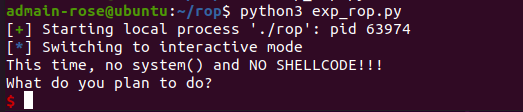




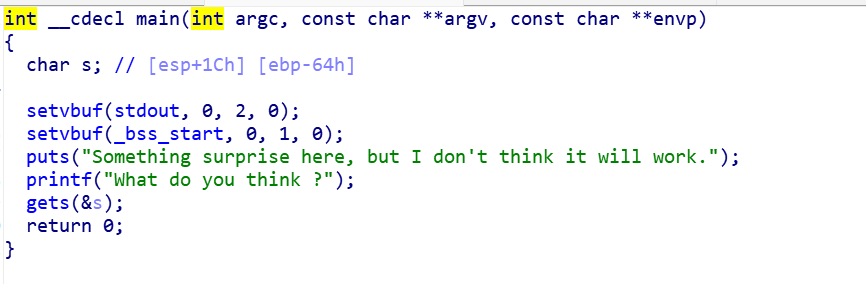
在IDA中，system函数确实存在，那么就可以利用get，system，/bin/sh来发起攻击。Payload如下：

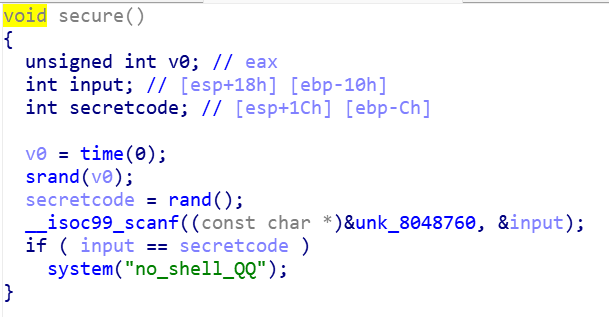


攻击效果如下：



ret2libc2

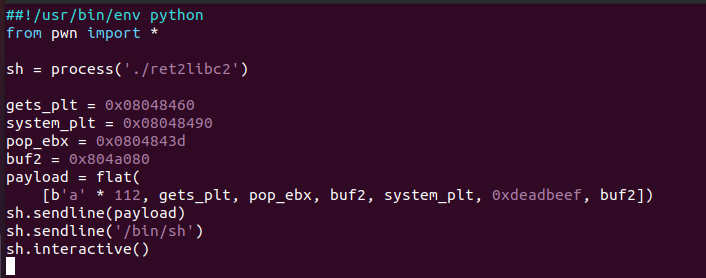




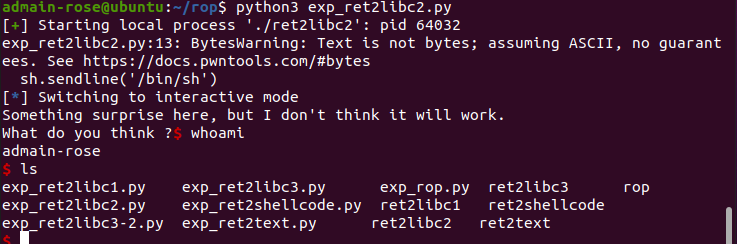


可以看到ret2libc2有get，system函数，但是没有/bin/sh。

这次，向程序中 bss 段的 buf2 处写入 /bin/sh 字符串，并将其地址作为 system 的参数传入。这样以便于可以获得 shell。Payload如下：

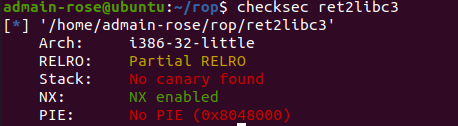


攻击效果如下：

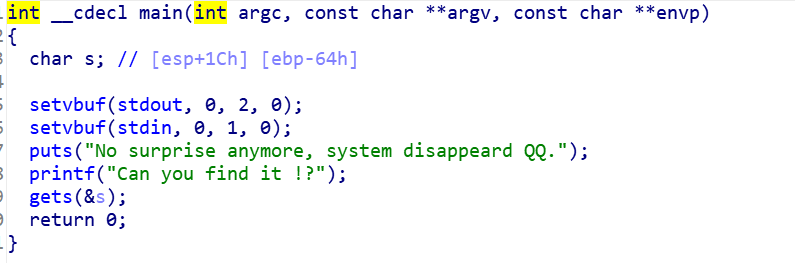


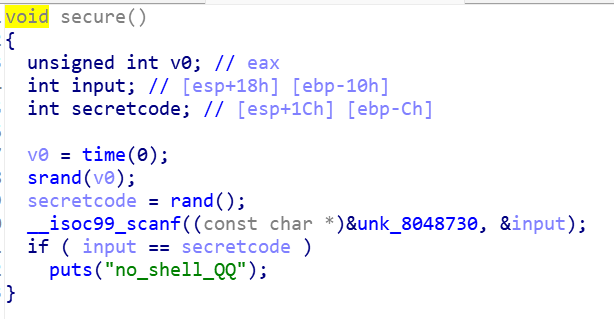
ret2libc3

首先检测程序开启的保护：



然后查看源代码：







可以看出，有get函数，但是没有system函数了，也没有/bin/sh了。

这次尝试利用外部libc库中的代码。如果我们知道 libc 中某个函数的地址，那么我们就可以确定该程序利用的 libc。进而我们就可以知道 system 函数的地址。一般常用的方法是采用 got 表泄露，即输出某个函数对应的 got 表项的内容。基本思路是：

1.泄露 \_\_libc\_start\_main 地址

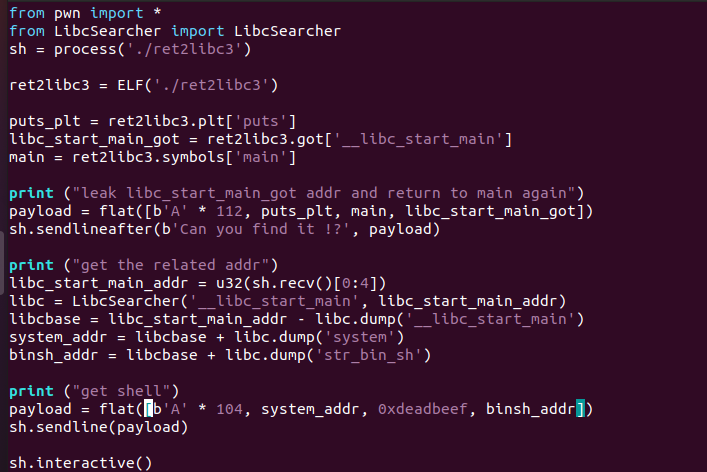
2.获取 libc 版本

3.获取 system 地址与 /bin/sh 的地址

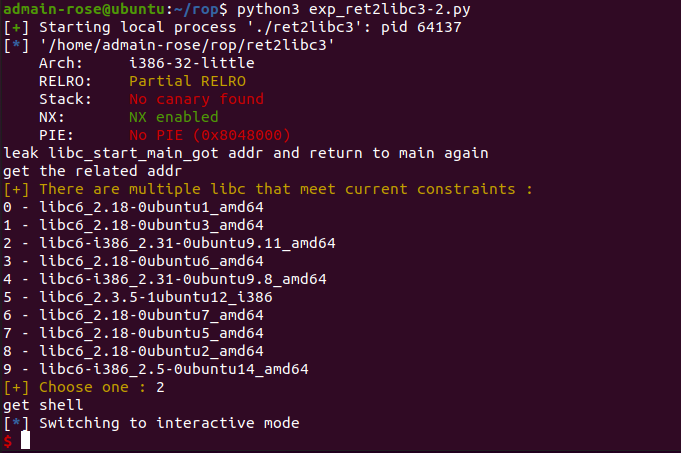
4.再次执行源程序

5.触发栈溢出执行 system(‘/bin/sh’)

Payload如下：

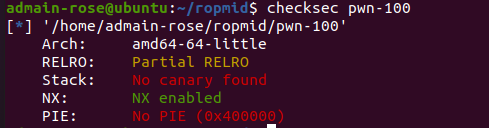


攻击效果如下：

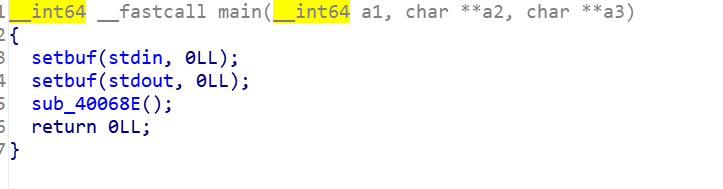


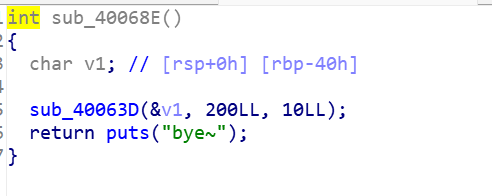
中级 ROP：2016 XDCTF pwn100

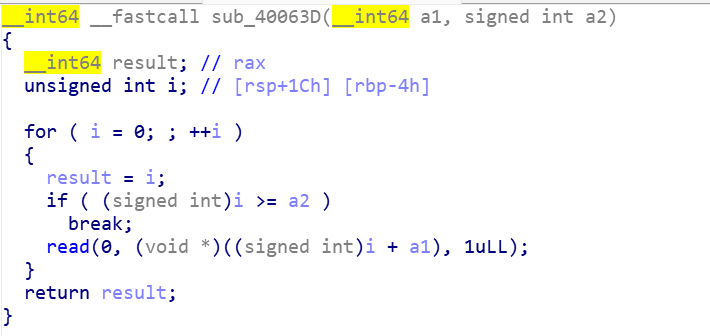
首先检测一下程序：

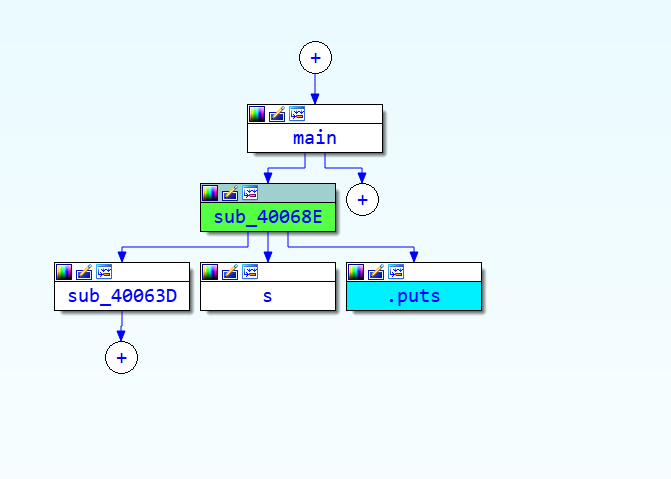


可以看到仅开启了NX，然后用IDA看一下源码：









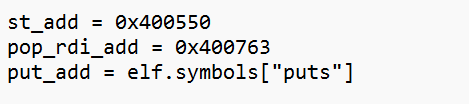
可以看到，程序的大致逻辑就是：main中调用sub\_40068E函数，在sub\_40068E中调用了sub\_40063D这个函数，并且传入了v1的地址和整型200。然后在sub\_40063D函数中循环了200次，这两百次每次都都一个字符到i+a1的内存中。也就是v1中读入了200个字节，但是v1的容量只有64个字节，这会造成缓冲区溢出。

在查看IDA后，没有system函数，没有/bin/sh字符串，还有NX保护，那么就使用ret2csu 与泄漏libcbase的方法。

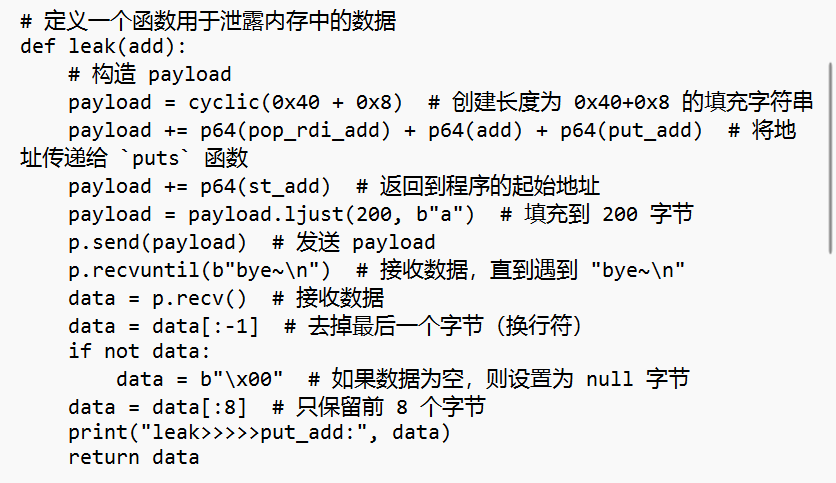
首先把用ROPgadget把基地址找出，并使用dynelf函数以及puts的plt地址，返回函数开头的地址。



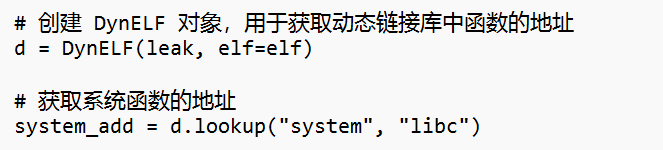
相应代码如下：



然后构造一个泄露具体某函数的真实地址的函数，代码如下：

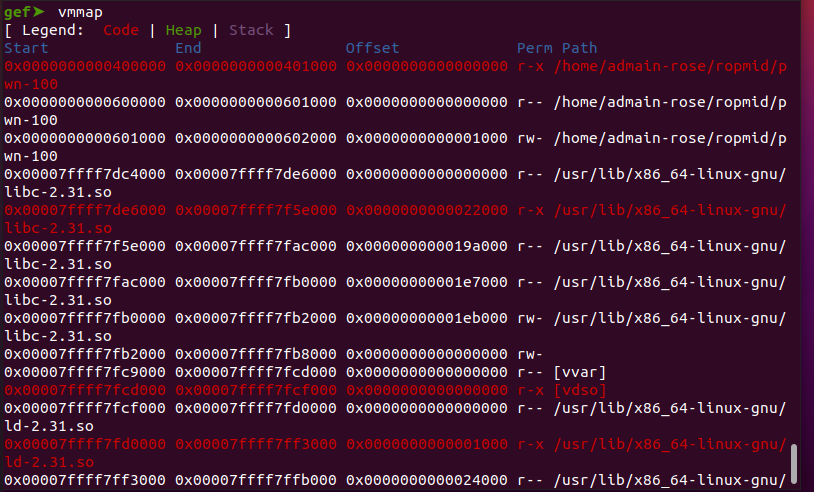


然后实例化,并且拿到system函数的地址：

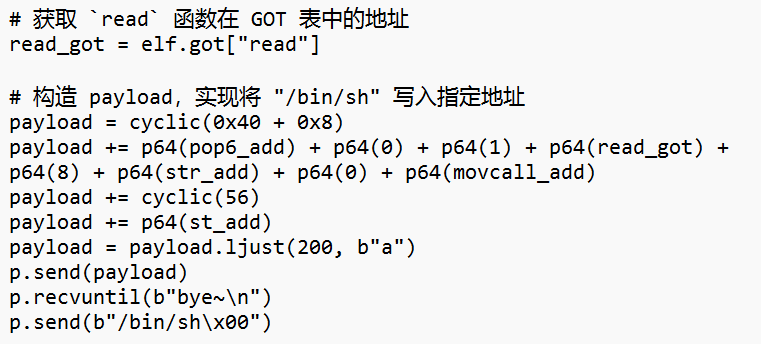


由于不存在/bin/sh，需要我们手动写入一个可以写入的地址：

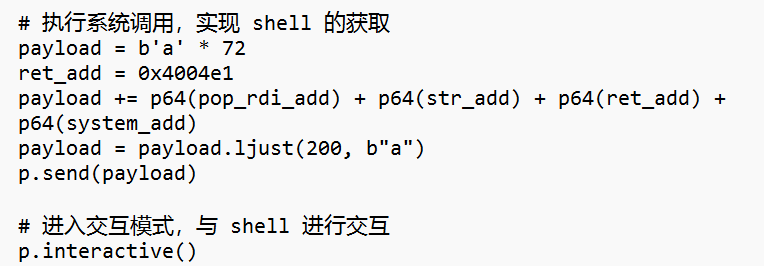




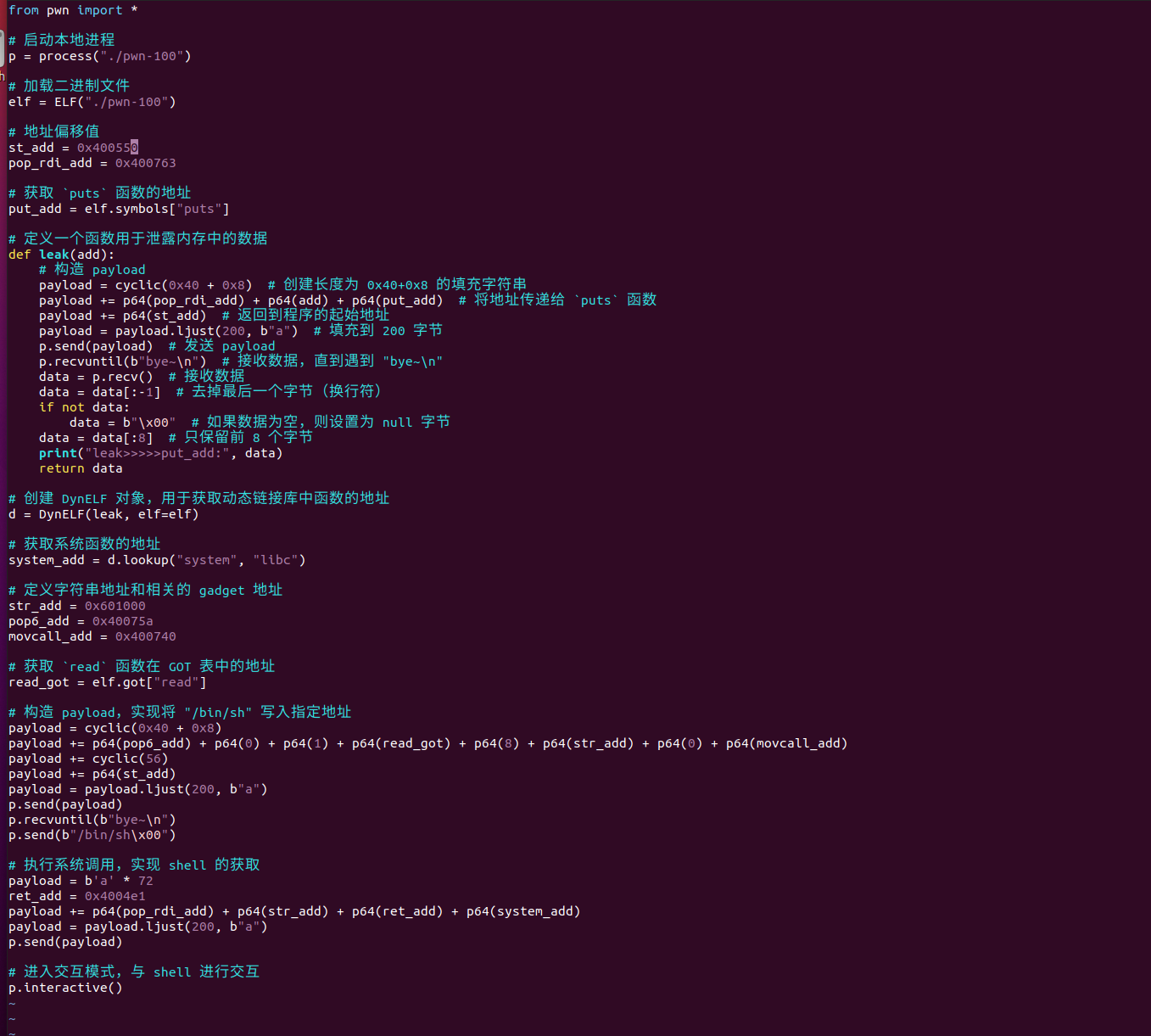
这里我们选取的是0x401000这块内存地址，开始往0x401000这块内存写入/bin/sh



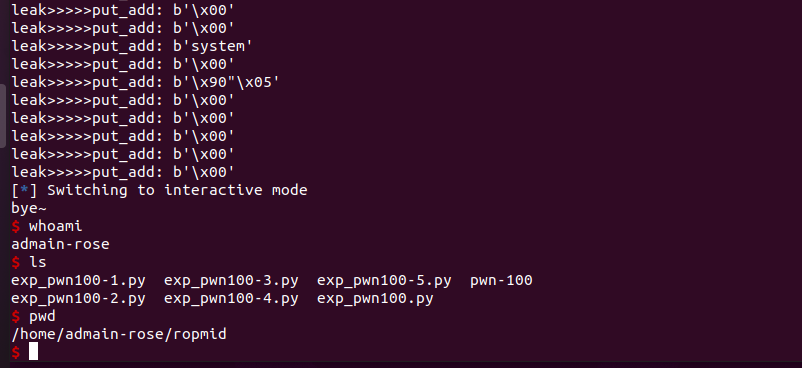
内存写入后，就可以调用system("/bin/sh")拿到shell：



最终完整的payload为：



攻击效果如下：



所有代码和文档均发布在GitHub网站：https://github.com/fakerjac/ROP.git