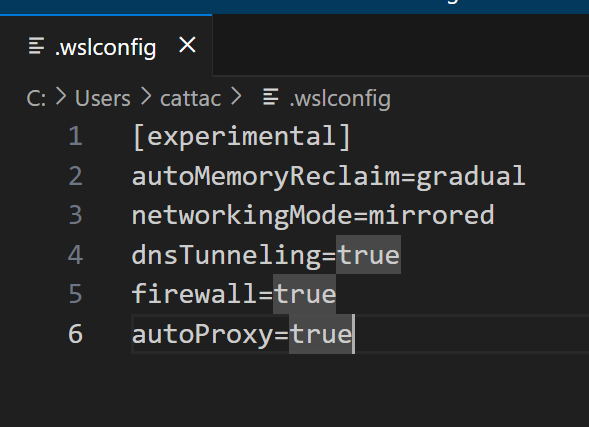
**黄子越 202428013229125**

**GitHub项目地址：**

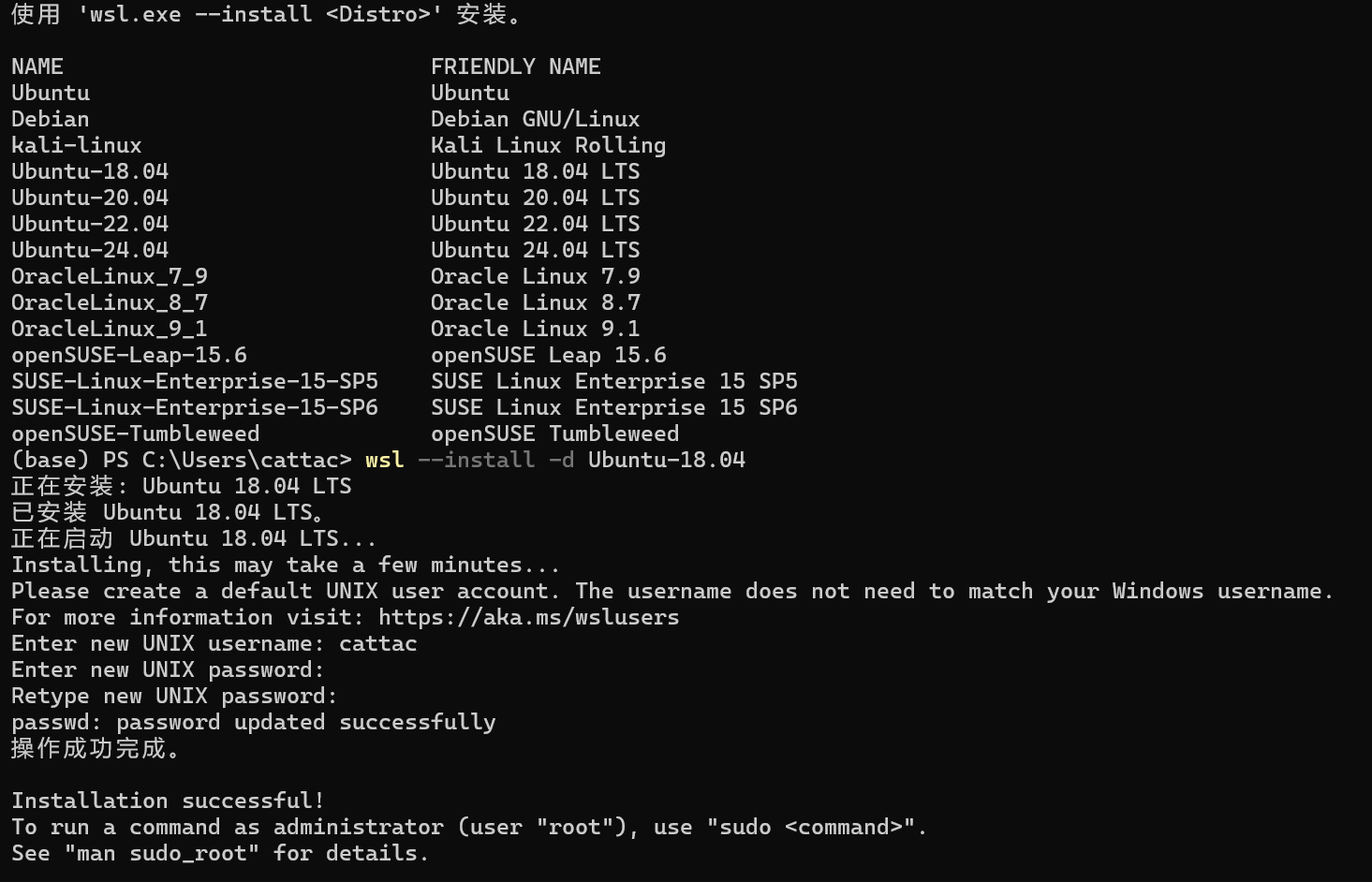
[**https://github.com/Hzyuer/Return-Oriented-Programming**](https://github.com/Hzyuer/Return-Oriented-Programming)

**git 链接：https://github.com/Hzyuer/Return-Oriented-Programming.git**

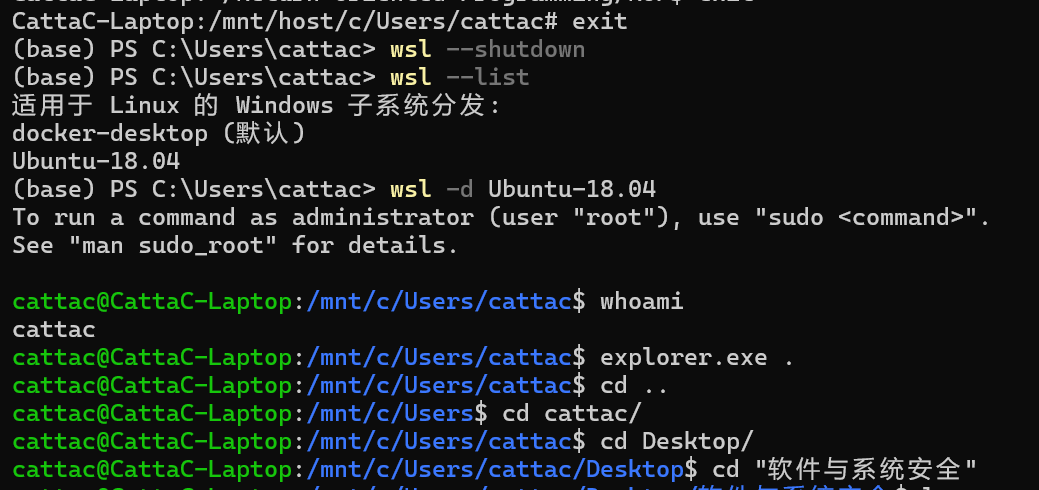
**配置WSL2的NAT代理**



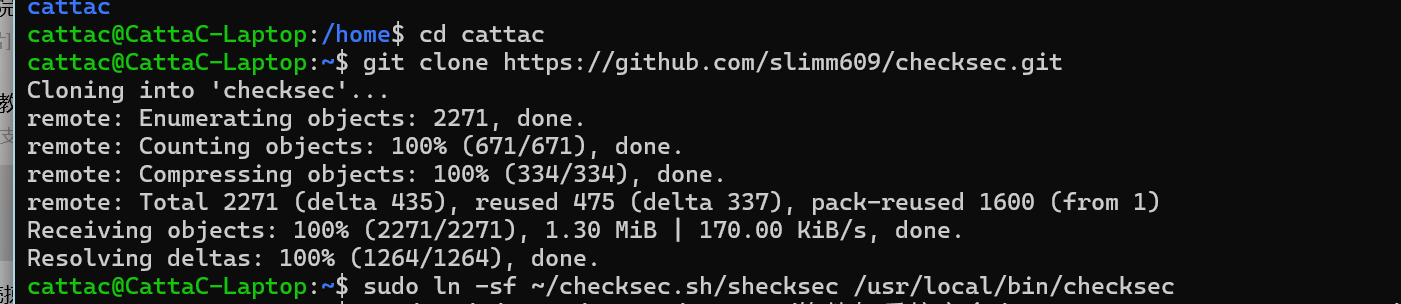
**安装Ubuntu18.04**



**启动正确的子系统**

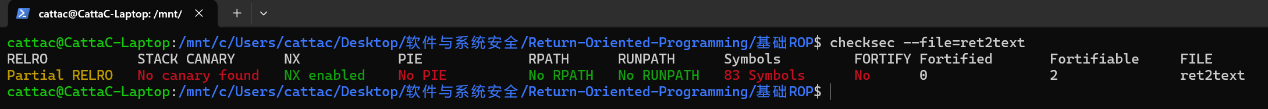
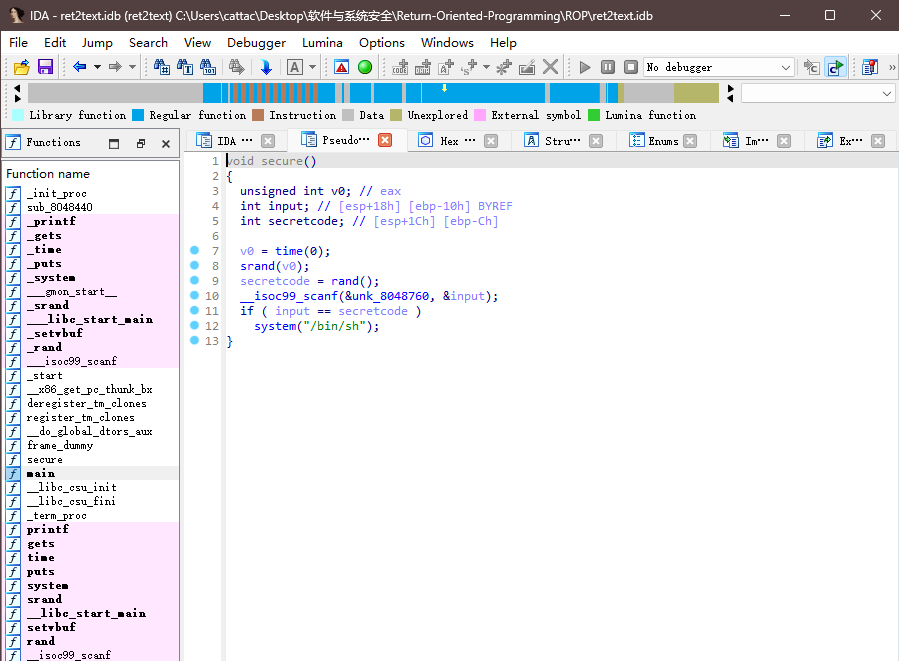


**下载并链接checksec**

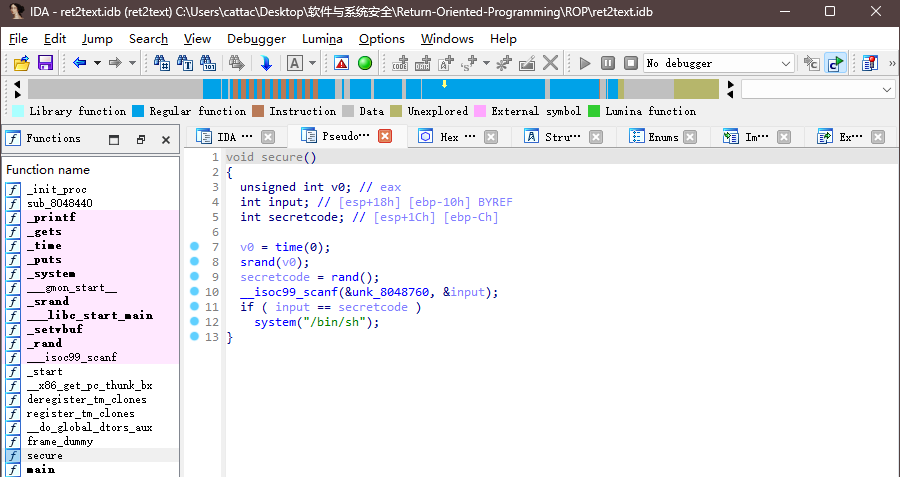


**Ret2text:**

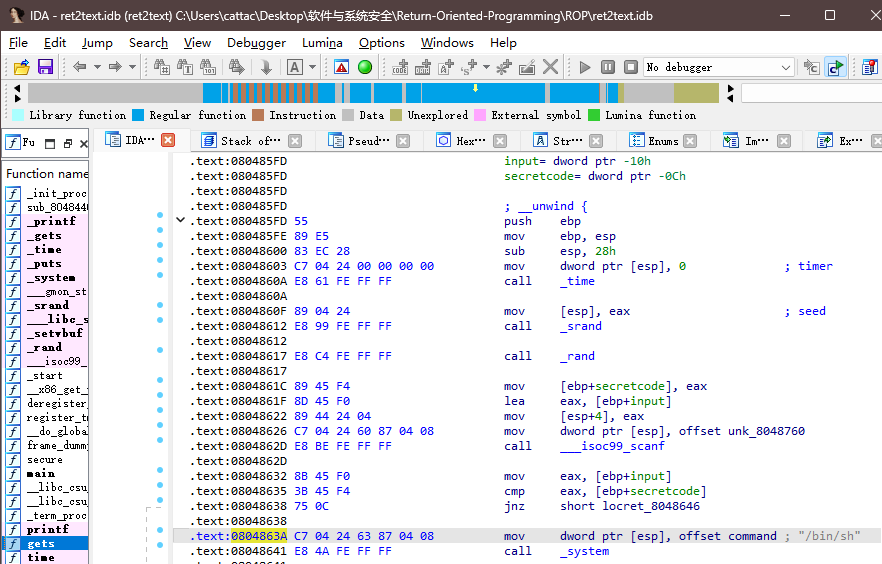
**运行checksec**

**使用IDA** **Pro分析代码**

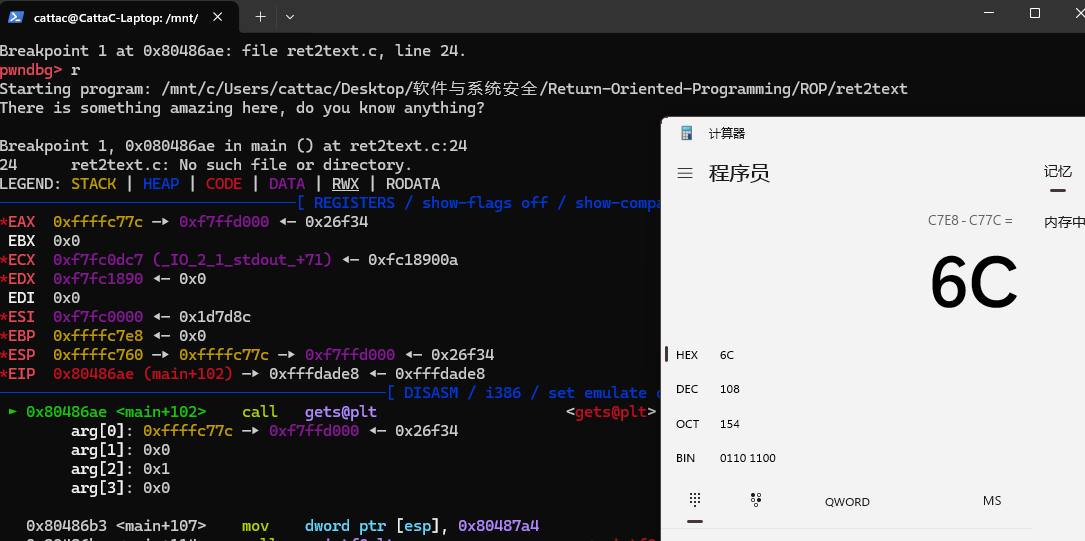
**发现调用shell的方式：使input和生成的随机数相同即可**



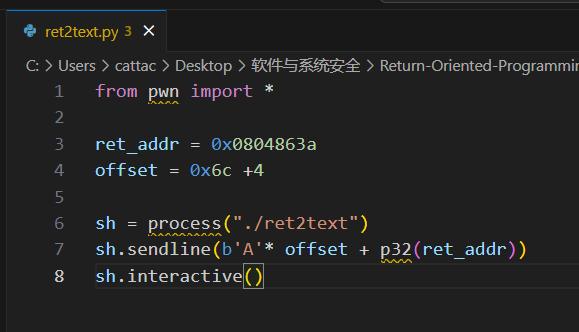
**查看汇编的地址内容，计算出input和secretcode的相对距离**



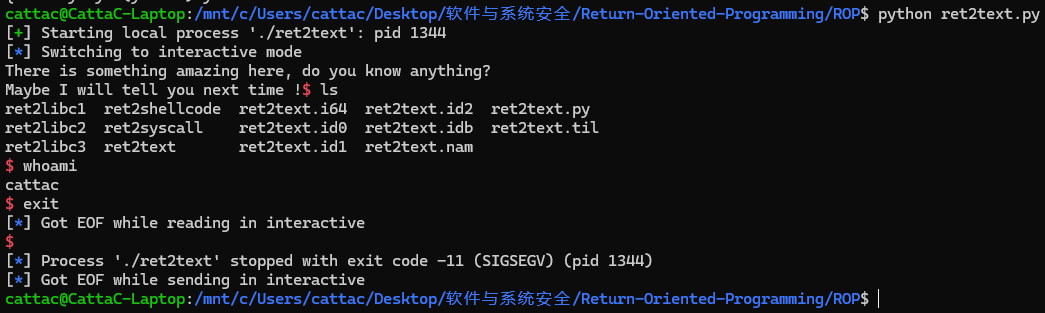
**查看栈长如图：**



**编写得到交互代码如图**

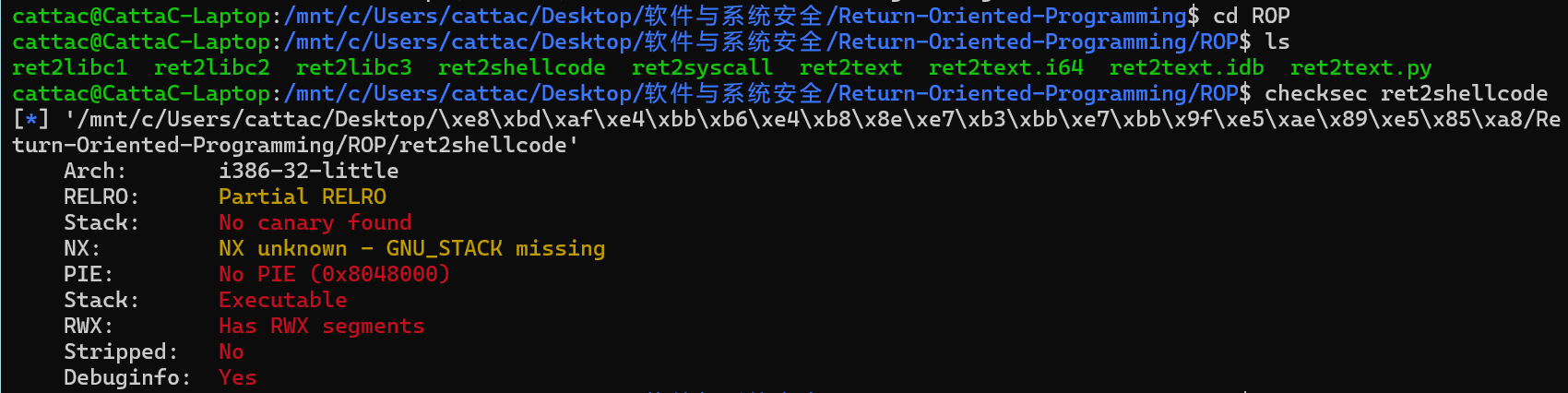


**运行结果如下：**

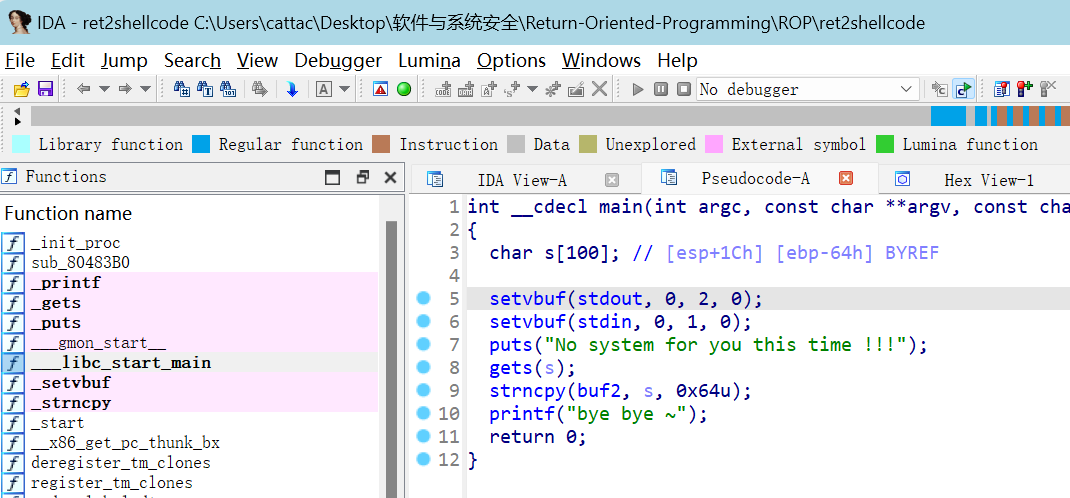


**Ret2shellcode**

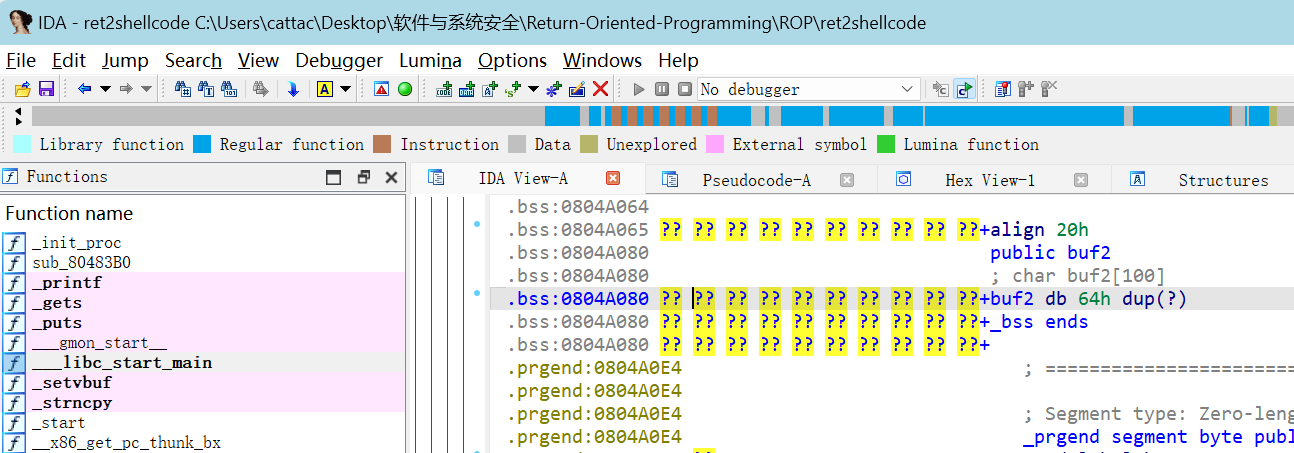
**使用checksec分析**



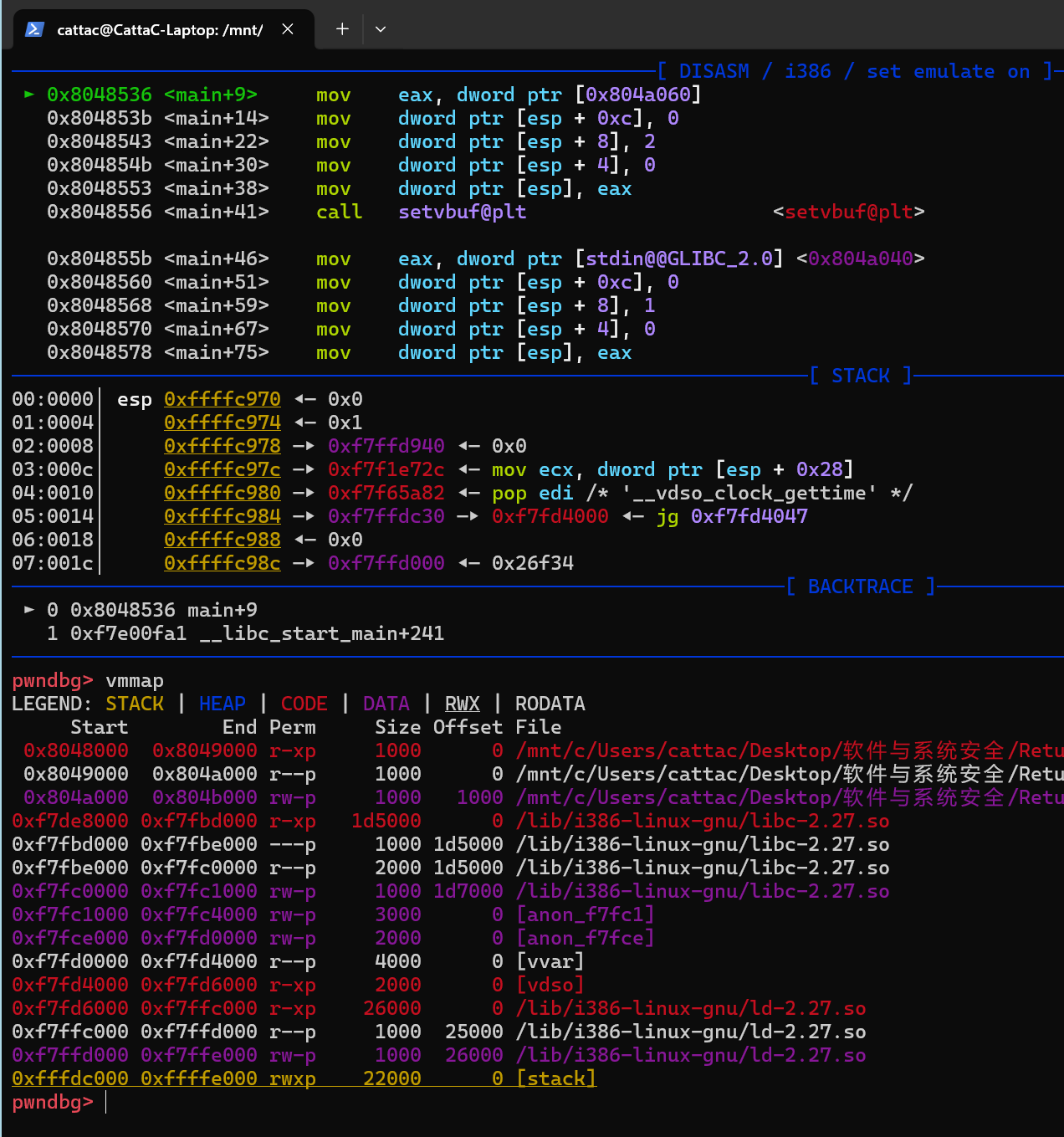
**查看反编译代码**



**可见buf2的位置**

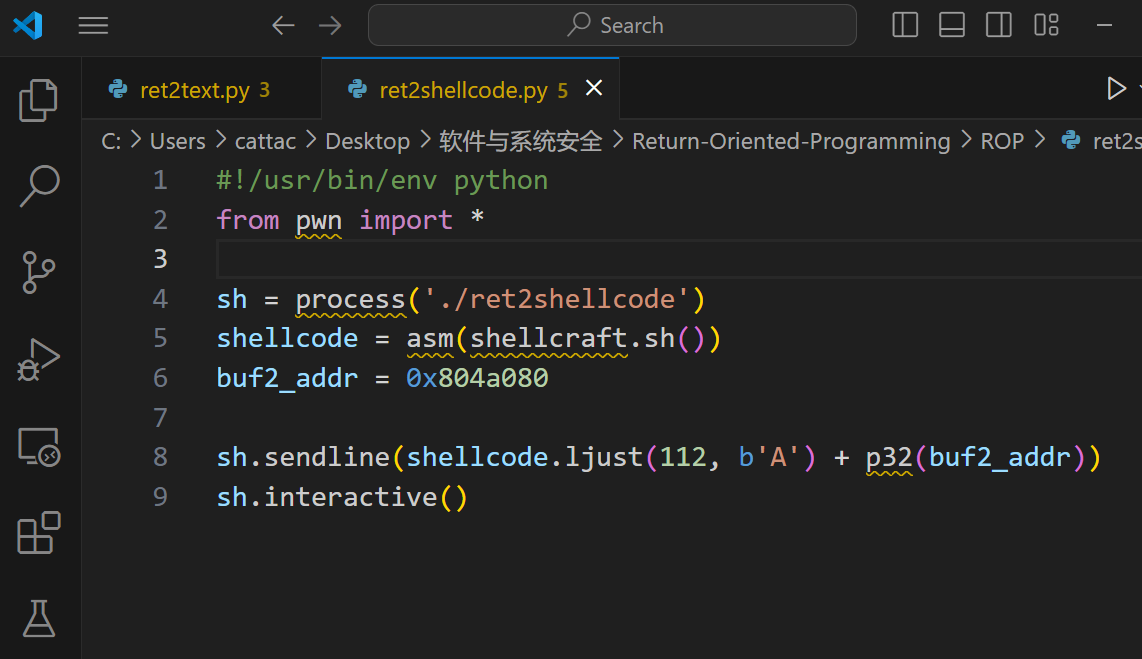


**使用pwndbg后，可见buf2所在位置没有shell的执行权限**

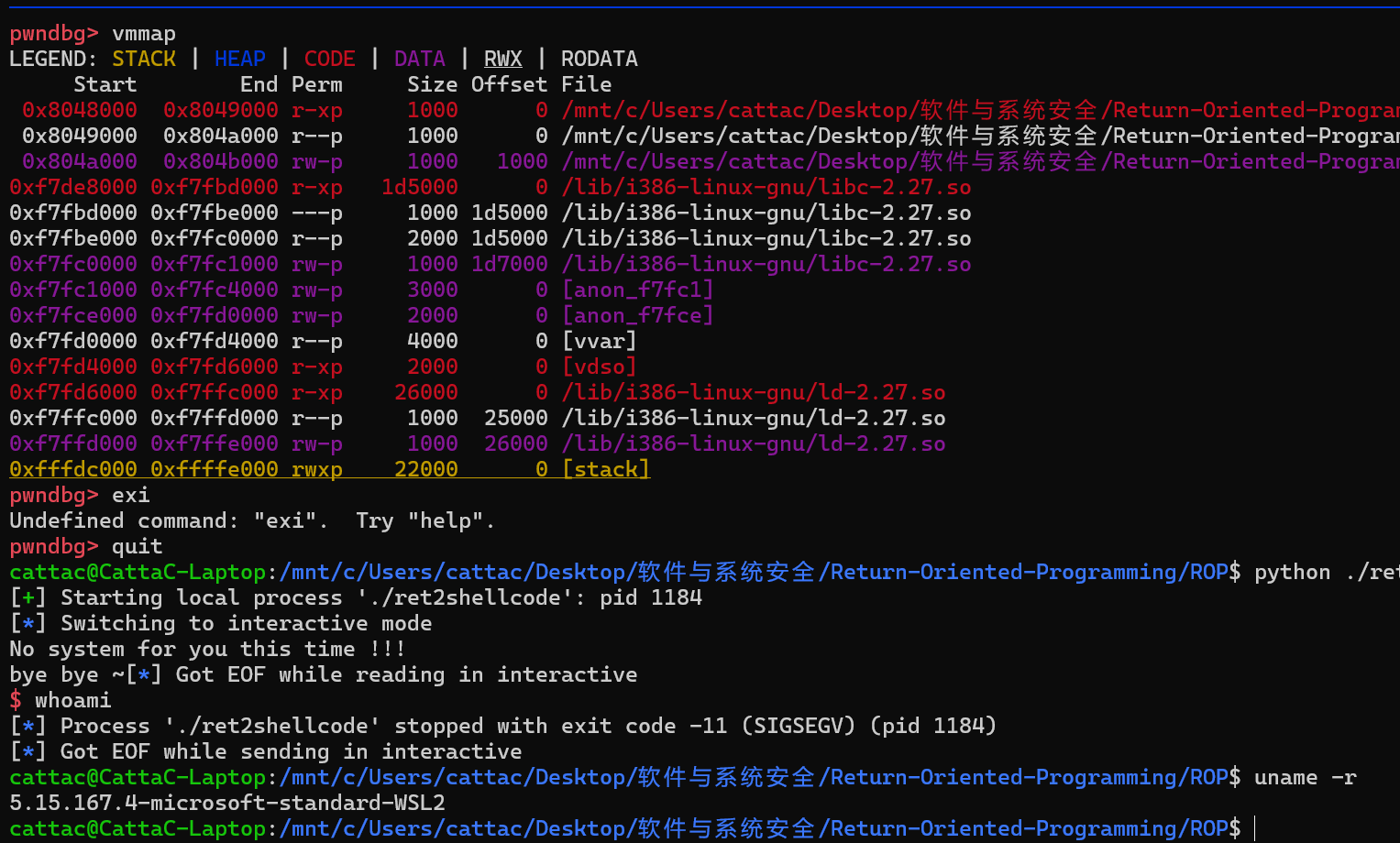


**（这里就没办法直接从.bass段注入了）**

**构建注入代码：**



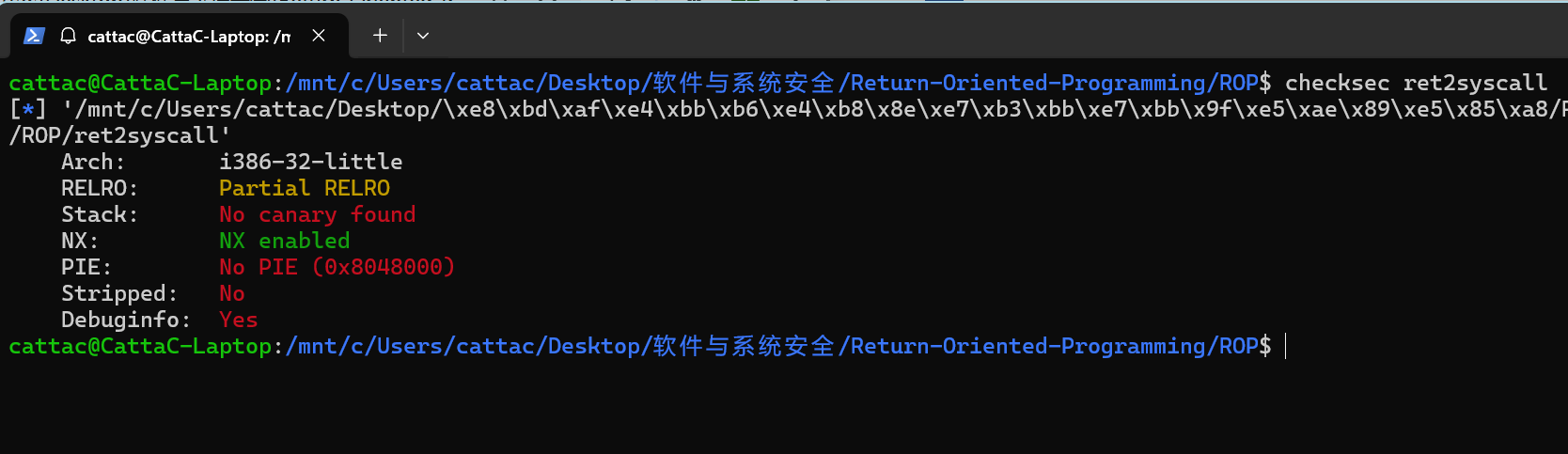
**仍然尝试使用.bass注入的方式，结果如下**



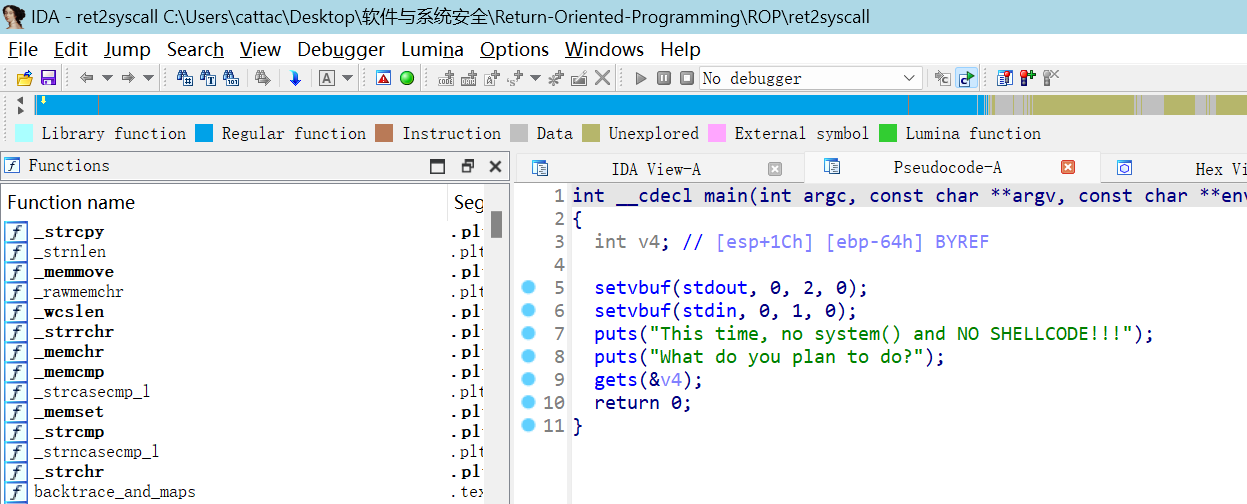
**可见失败的主要原因就是使用了WSL2 的系统内核，自动将.bss段保护了，因此攻击失败**

**Ret2syscall**

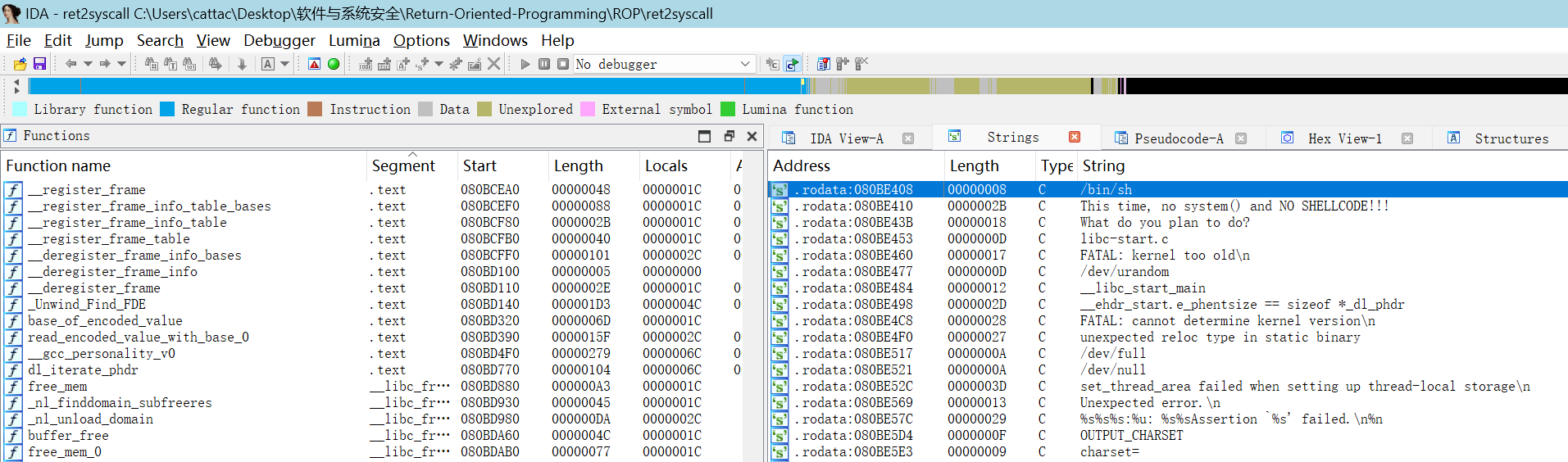
**首先进行checksec分析，可见NX保护开启**



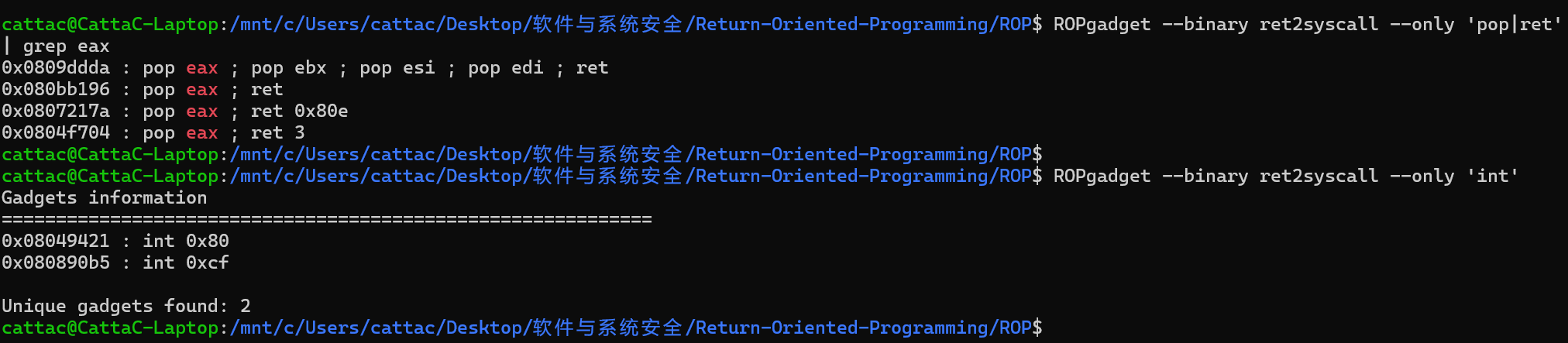
**反编译可执行文件，可见gets漏洞可利用：**

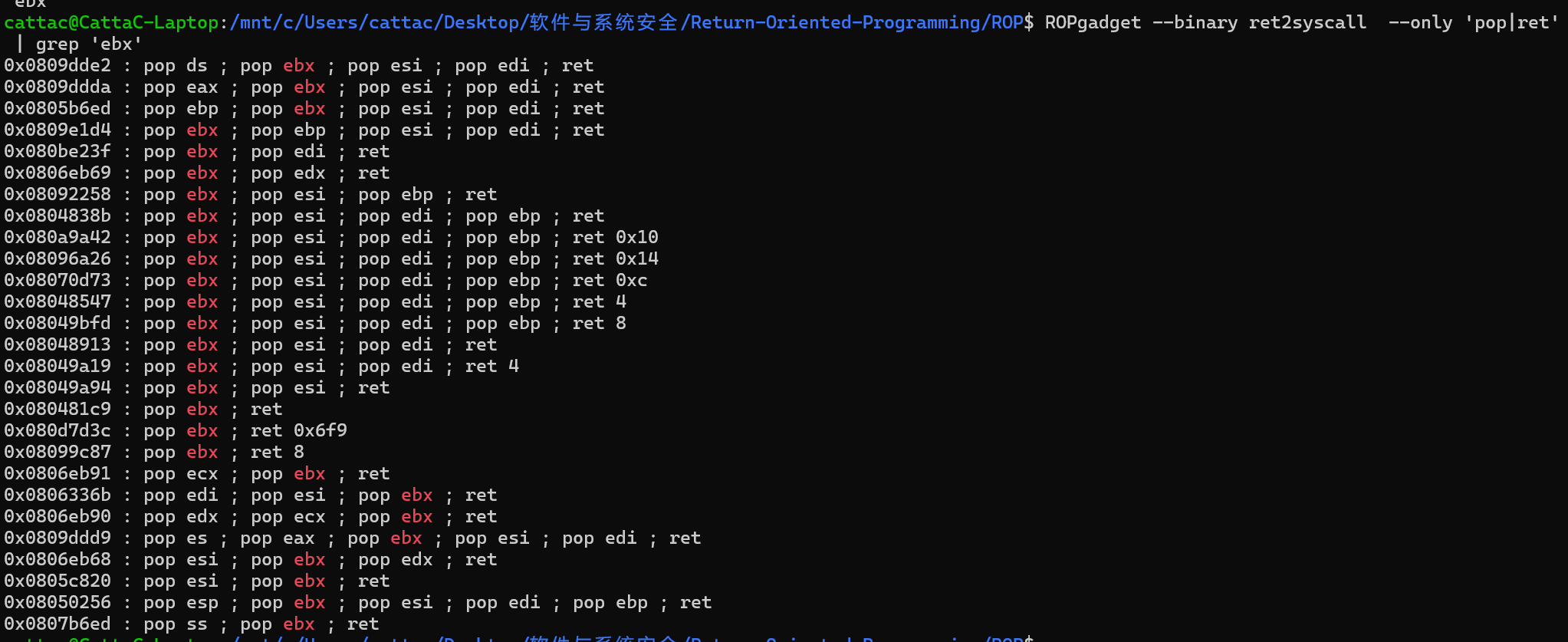


**在string的subview中可见/bin/sh：**



**使用ROPgadget寻找可用代码片段**





**可以获得如下寄存器和data的地址：**

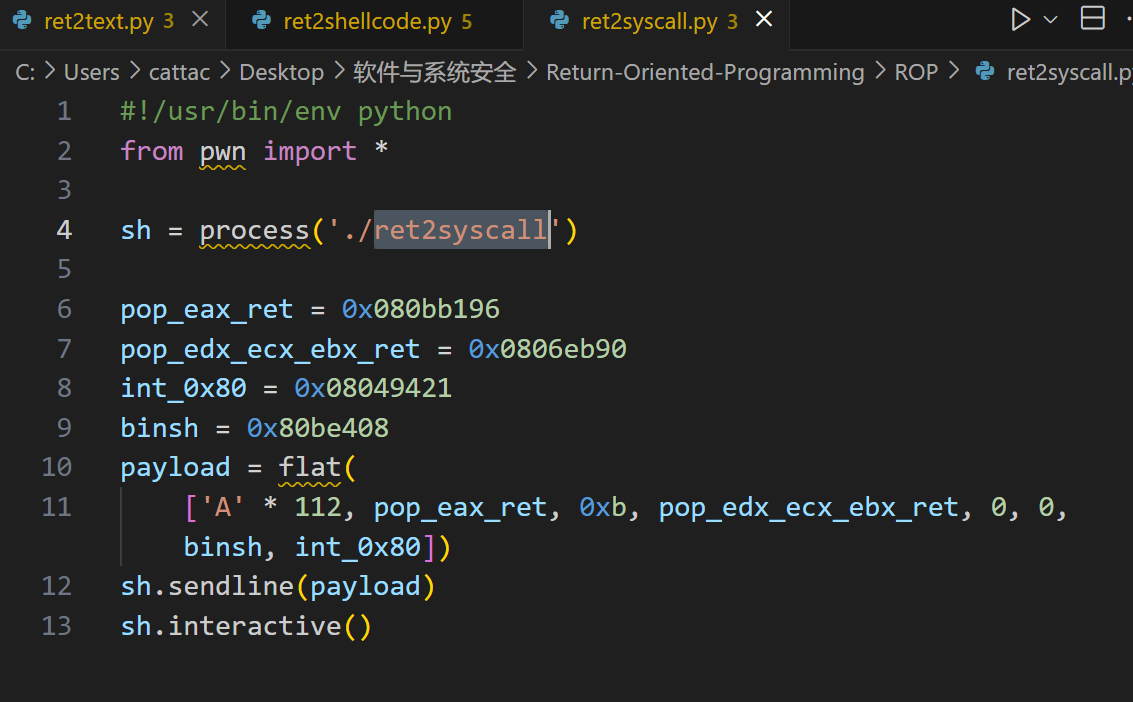
**0x080bb196 : pop eax ; ret**

**0x0806eb90 : pop edx ; pop ecx ; pop ebx ; ret**

**0x080be408 : /bin/sh**

**0x08049421 : int 0x80**

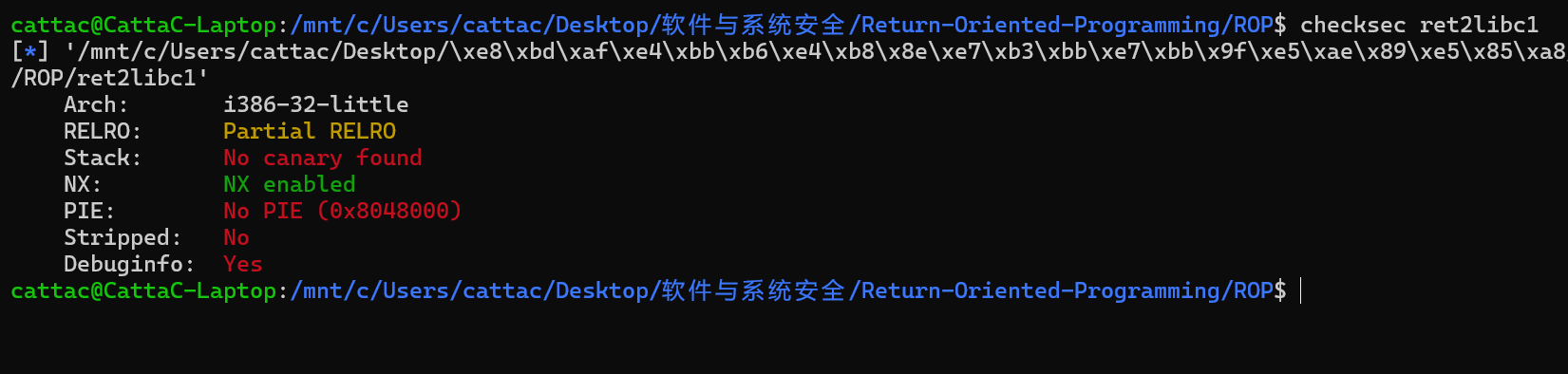
**编写代码**



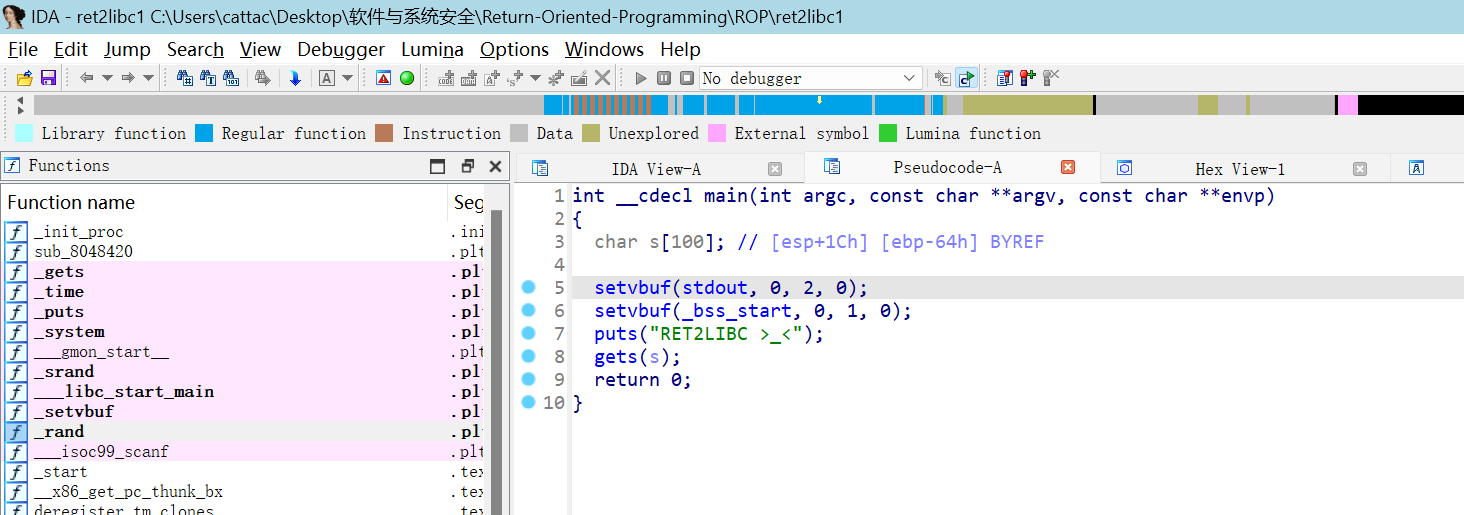
**运行结果**



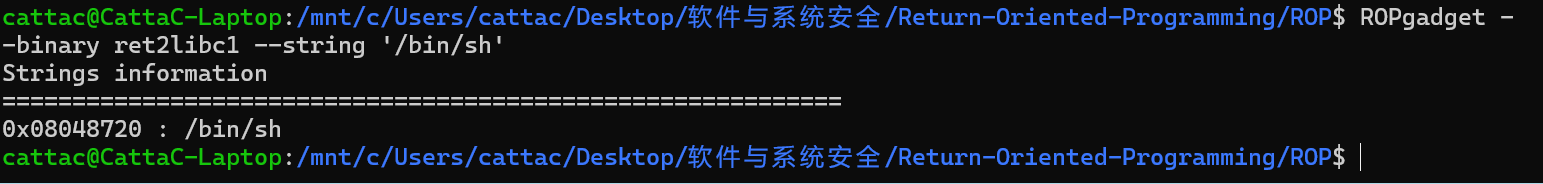
**ret2libc1同样是首先使用checksec进行保护模式的检测**



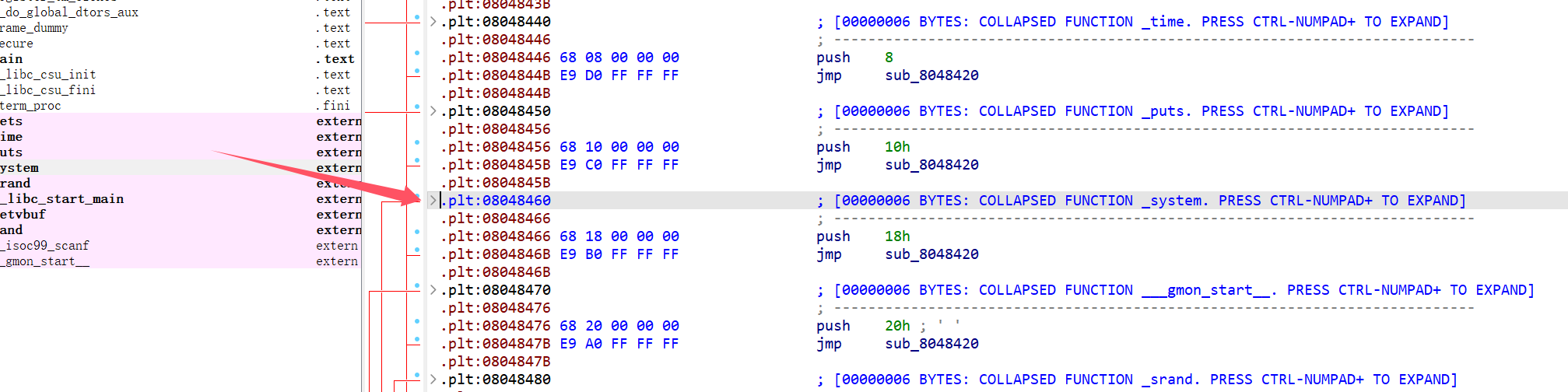
**接着使用IDA Pro进行反编译**



**这次实验首先使用ROPgadget来寻找’/bin/sh’**



**此外，本次实验比较幸运的地方在于可以直接找到调用system函数的位置：**

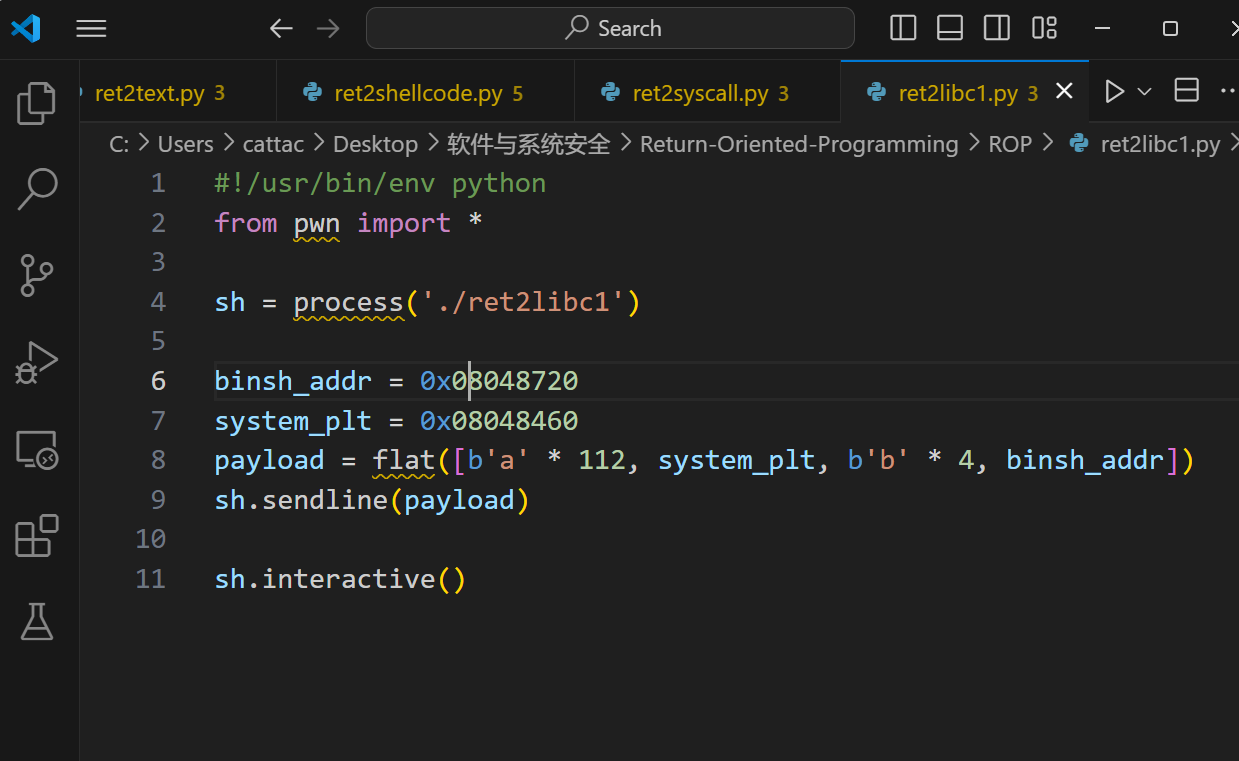


**可见，所需内容的位置如下：**

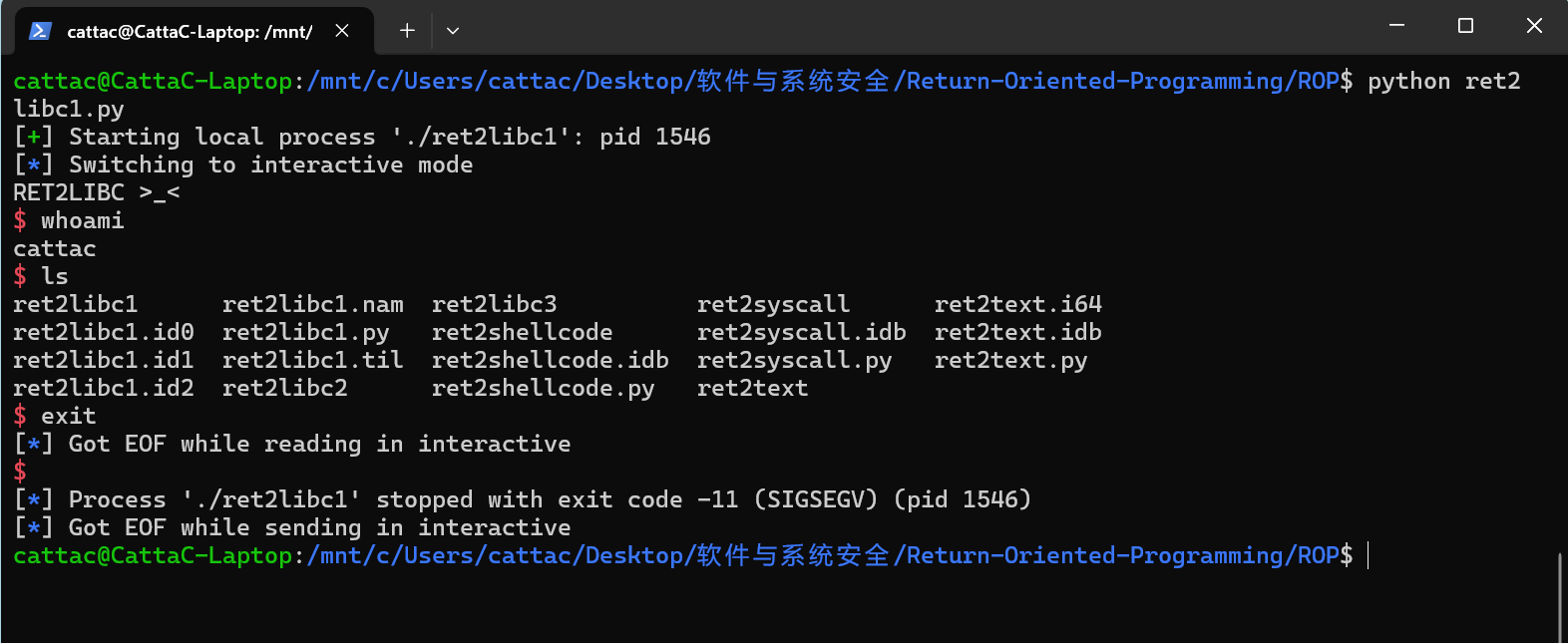
**0x08048720 : ‘/bin/sh’**

**0x08048460：system**

**编写代码如下：**

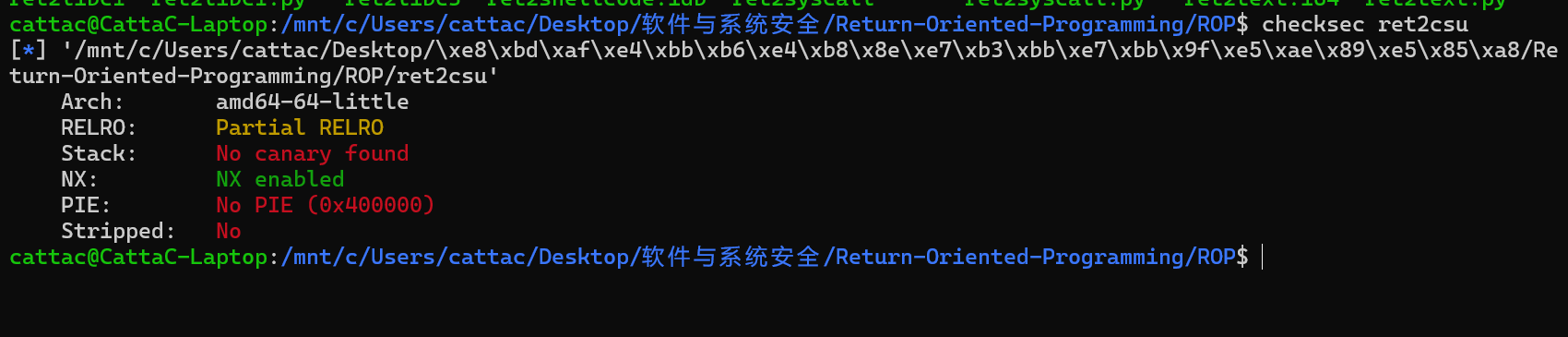


**运行结果如图：**

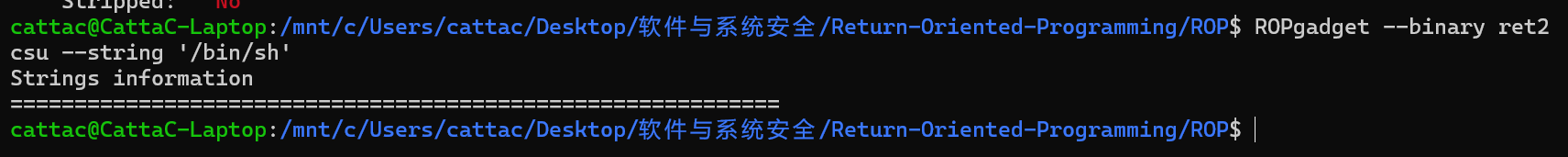


**Ret2csu**

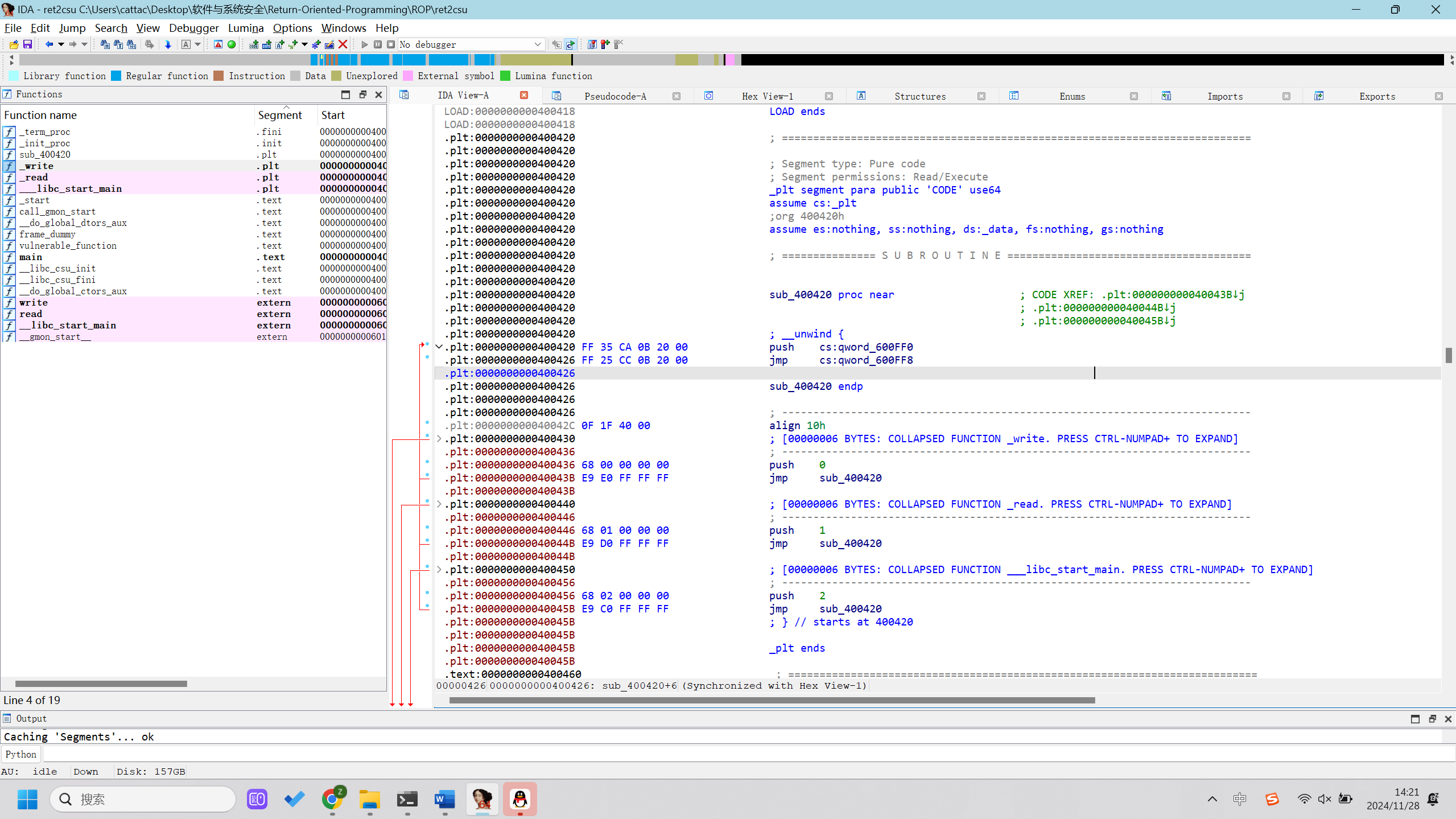
**首先查看程序信息：**



**没有发现’/bin/sh’:**



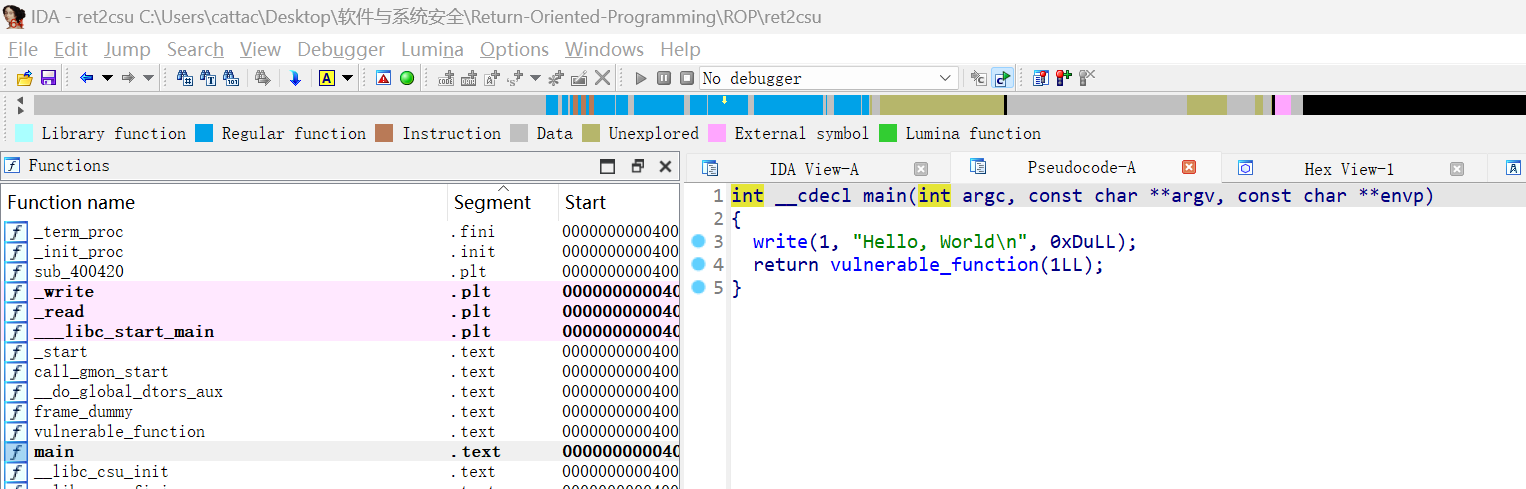
**同时，在.plt段中也没有发现system函数的调用情况**



因此，考虑到攻击所需的常用手段，我们需要在程序代码之外的合适地方寻找到system和‘/bin/sh’这两个关键的gadgets，也即寻找通用gadget。

1. 程序分析

首先，我们还是使用IDA Pro来对ret2csu的可执行文件进行分析。通过checksec的分析，我们已知该文件是64位的可执行文件，因此需要使用IDA Pro x64版本。

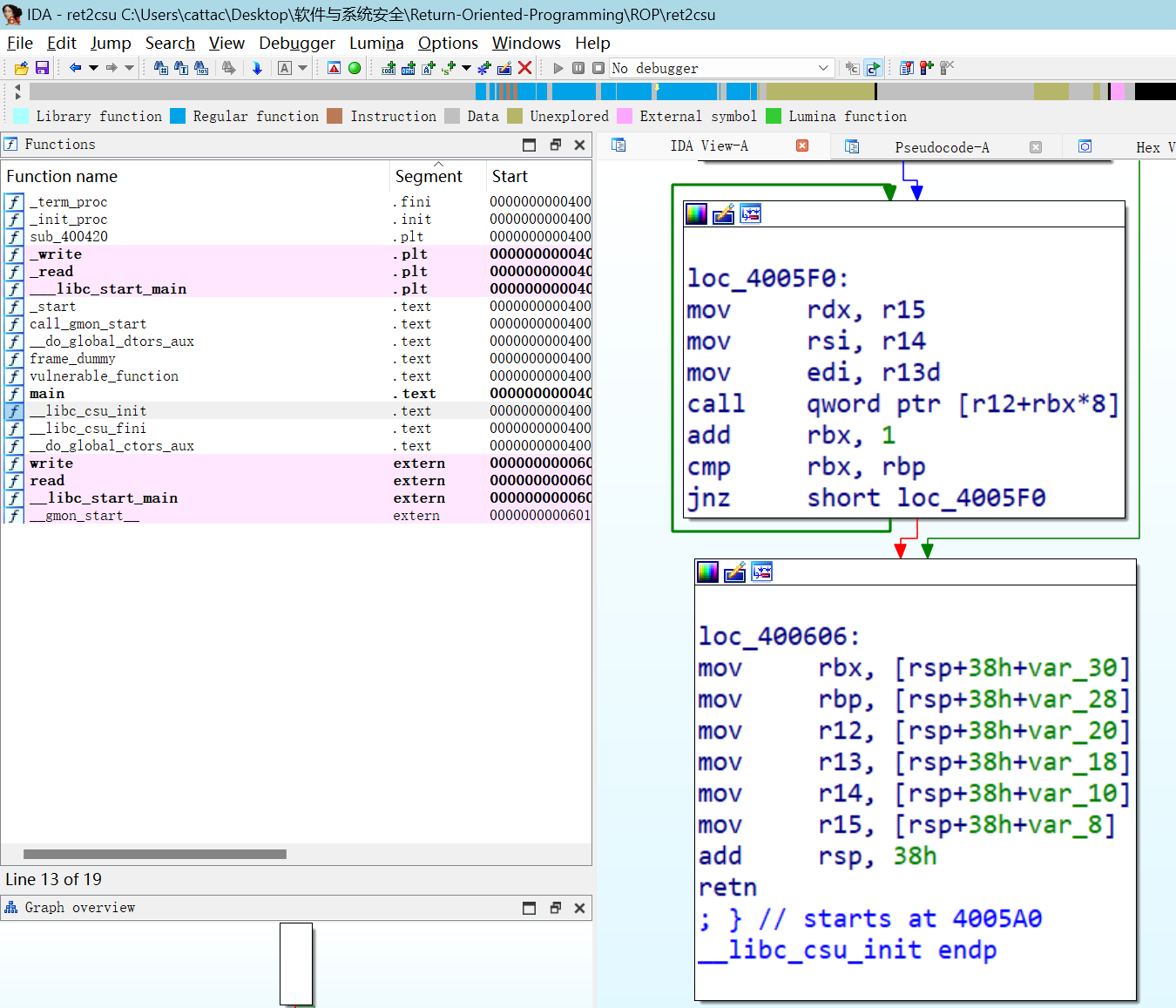


主函数可见，在进行简单的消息输出之后调用了vulnerable\_function这一函数，因此我们只能到该函数中序寻找可能的漏洞并发起攻击。因为程序在编译过程中会加入一些通用函数用来进行初始化操作（比如加载libc.so的初始化函数），所以虽然很多程序的源码不同，但是初始化的过程是相同的，因此针对这些初始化函数，我们可以提取一些通用的gadgets加以使用，从而达到我们想要达到的效果。

在通过查询网络资料加以学习后，了解了利用x64编译下的\_\_libc\_csu\_init方式，简称为CSU。

1. 漏洞原理

csu实际上是基于libc的一种泄露函数基地址的做法，其原理还是基于rop链的构造。大多时候我们难以找到每个寄存器对应的gadgets，但是当我们遇到wirte函数泄露的时候却必须要控制三个寄存器（64位首先存入数据的rdi，rsi，edi），这时候，我们就可以利用 x64 下的 \_\_libc\_csu\_init 中的 gadgets。



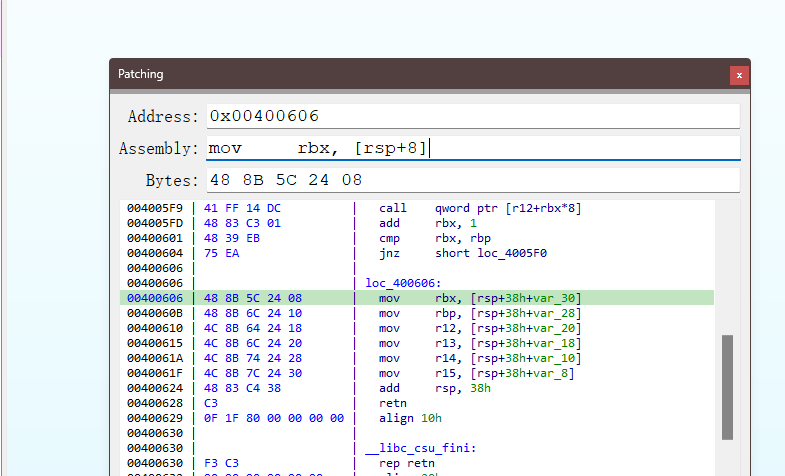
在IDA Pro中可见，csu中有如下两块代码值得我们注意和利用——前者记gadget1，后者记为gadget2。可见我们可以利用gadget1对rdx，rsi和edi进项赋值，而它们的数值来源，即r14，r13和r12是可以通过gadge2进行栈的溢出来注入数值的。

在完成以上三个数值的修改后，我们将会调用**call qword ptr [r12+rbx\*8]**。这时候我们只要再将rbx的值赋值为0，再通过构造栈上的数据，我们就可以控制pc指针去调用我们想要调用的函数了。执行完**call qword ptr [r12+rbx\*8]**之后，程序会对rbx+=1，然后对比rbp和rbx的值，如果相等就会继续向下执行并ret到我们想要继续执行的地址。】

三、攻击方式

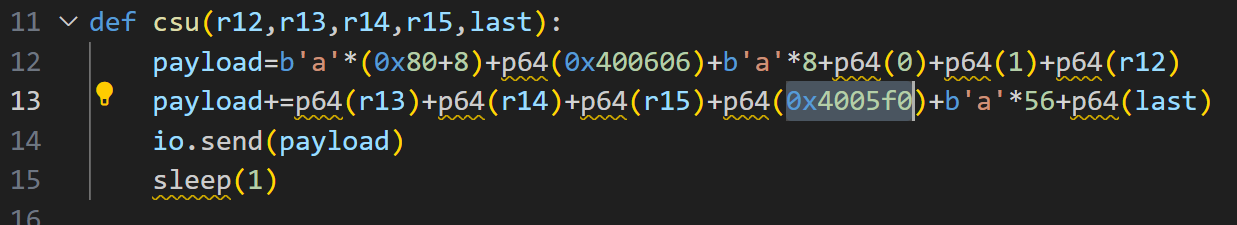
首先，我们利用栈溢出的方式，让write()函数输出其本身在内存中的地址，并返回到main函数重新执行，这里我们需要构建payload1来完成该任务，即使用gadget2

通过IDA Pro，我们可以看到csu对应的内存地址：

©

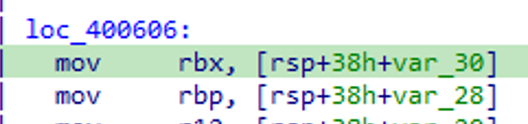
可见其gadget2地址为0x400606，gadget1的地址为0x4005f0。

因此在跳过多余的pop和rbx的8位长度指令之后，我们得到了通过csu获取地址的方式：

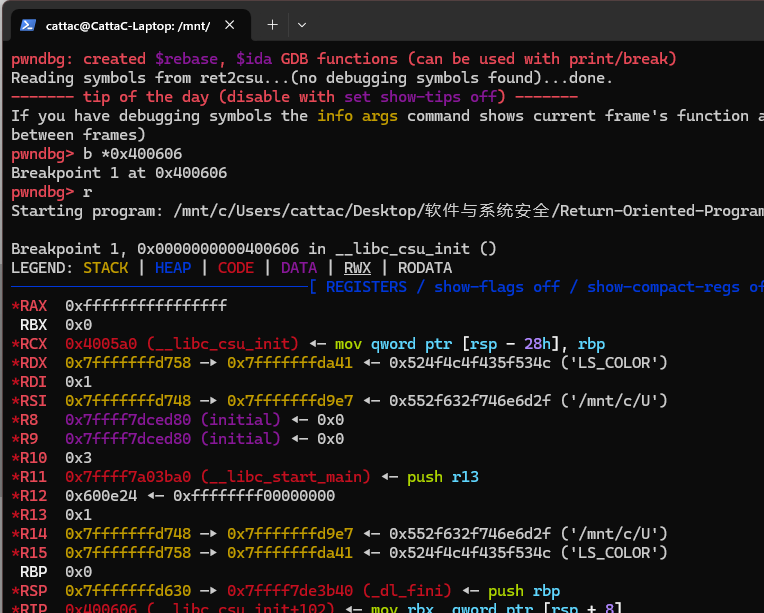


注意这里因为调用gadget2后使用sp指针传递数据，因此需要位置+’a’\*8来调整数据。

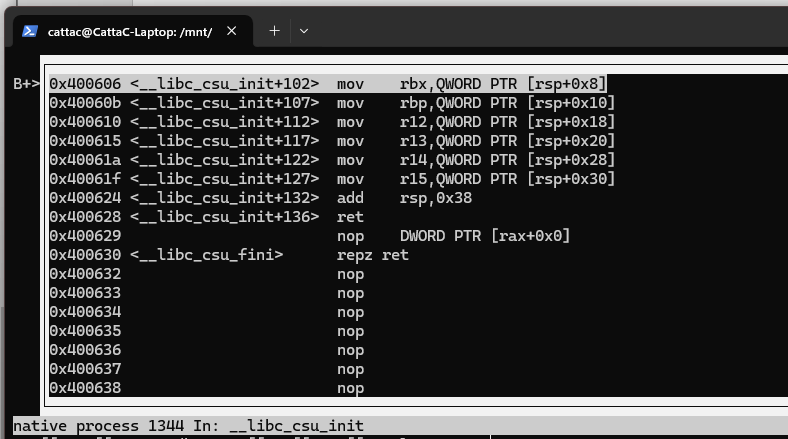
我们看到此处的偏移量为var\_30，于是使用gdb查看其大小：



在gadget2处断点运行：

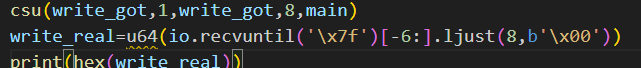


使用layout asm查看其汇编代码，过程如图:

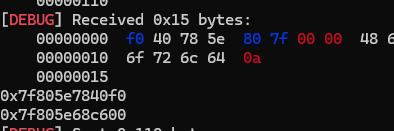


可见0x8大小的偏移。

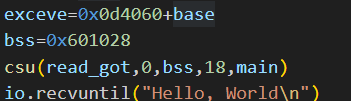
在执行完成csu的获取之后，我们发现栈顶指针rsp距离retn有0x8\*7个偏移(相当于传参6个参数和前面提到的8个垃圾数据，我们只有填充完才能覆盖retn返回main,来进行接下来的操作。



可见泄漏的system函数地址为：

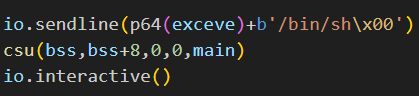


算出了基地址之后，我们就可以利用read（）将system()的地址以及“/bin/sh”读入到.bss段内存中。



.bss的位置也是由IDA Pro得出，此处不再赘述。

完成代码注入之后，我们还需要最后的一个执行步骤——此处system的地址被保存到了.bss+8的位置上——重复调用csu函数带来的相同偏移。



通过以上操作，我们最终得到了ret2csu攻击代码，运行结果如下：



至此，本次攻击圆满结束。

三，难点与总结

本次所进行的csu攻击，其主旨在于如何通过目标代码之外的内容获得gadget并展开攻击。首先通过栈溢出的方式获取程序的内部结构，绕过了NX保护和没有system与’/bin/sh’的难点，通过编译时基本常见的\_\_libc\_csu\_init（）来完成攻击——这就是本实验的最大亮点：除去代码主体函数外，gcc默认还会编译进去一些其它的函数，我们也可以尝试利用其中的一些代码来进行执行。因此，只要我们合理的控制了PC指针，就可以将源程序中一些地址进行偏移从而来获取我们所想要的指令。

就难点而言，首先第一步是确定原有代码是否开启了现代操作系统引入的 ASLR 和堆栈保护（Canaries）机制，有这些机制的话这中简单的攻击就难以奏效。其次是在编写payload2的时候，一定要搞清楚不同语句对应的偏移长度，否则将对试图对没有权限的位置进行RWX操作，引发程序崩溃，导致攻击失败。

1. 攻击的防范

通过开启RELORO，NX防护，栈保护，堆执行保护等方式，可以有效的阻止攻击者运行注入的代码；也可以通过ASLR地址随机化的方式来防止攻击者获取注入共计的地址。