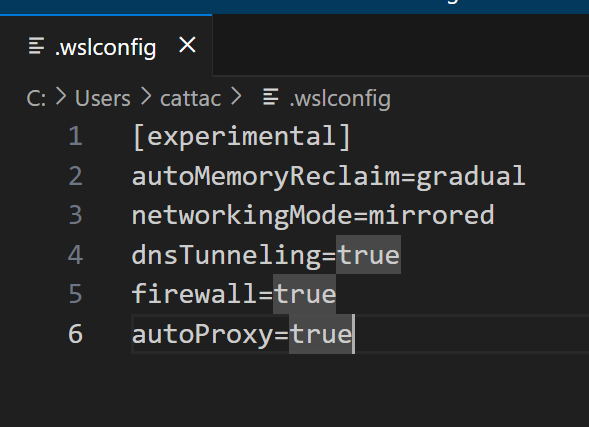
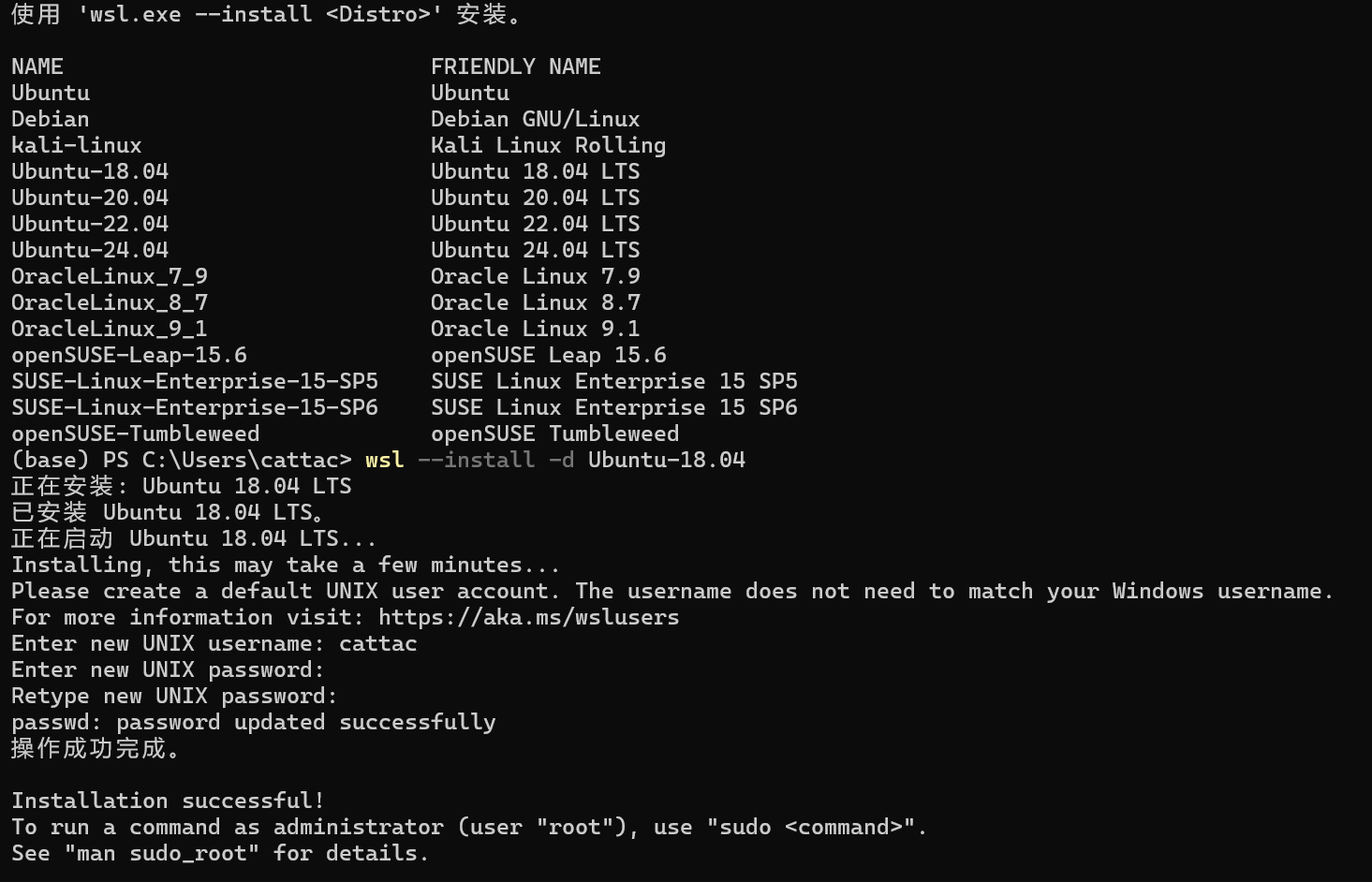
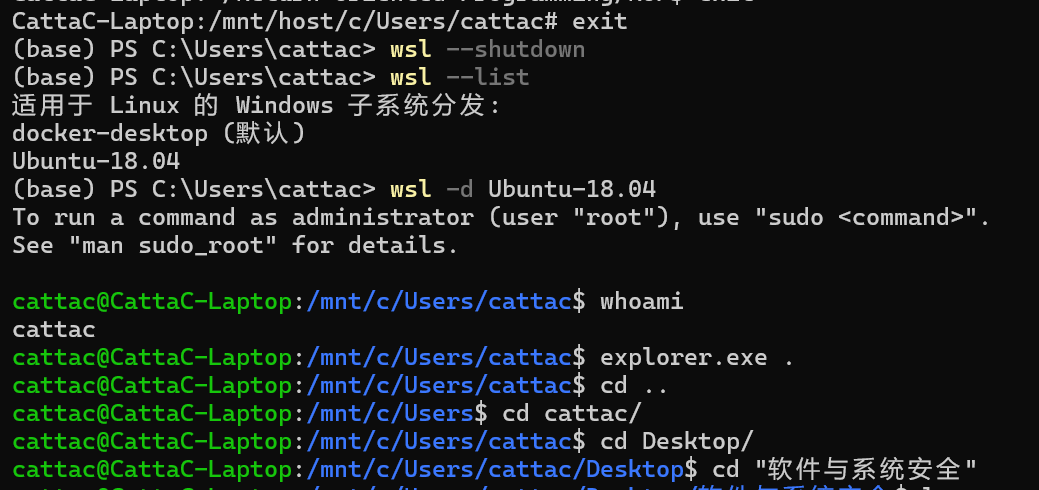
**配置WSL2的NAT代理**



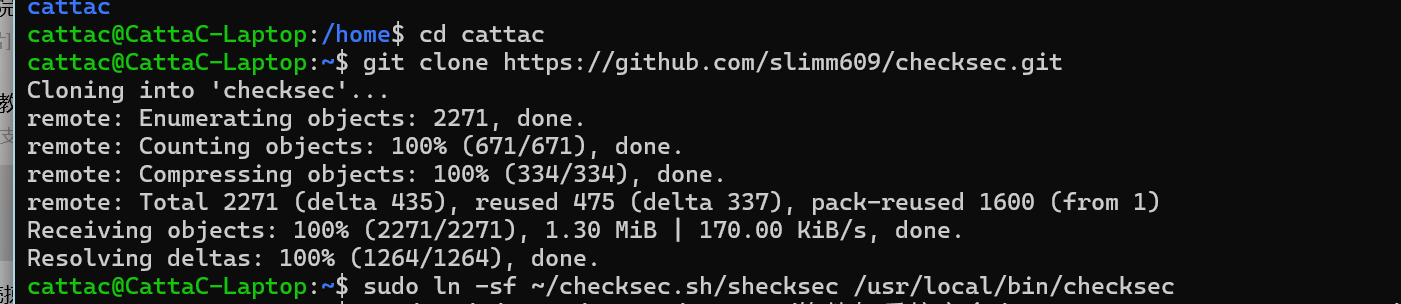
**安装Ubuntu18.04**



**启动正确的子系统**

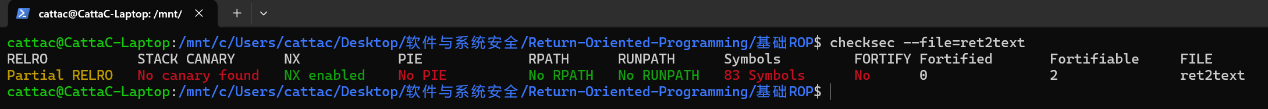
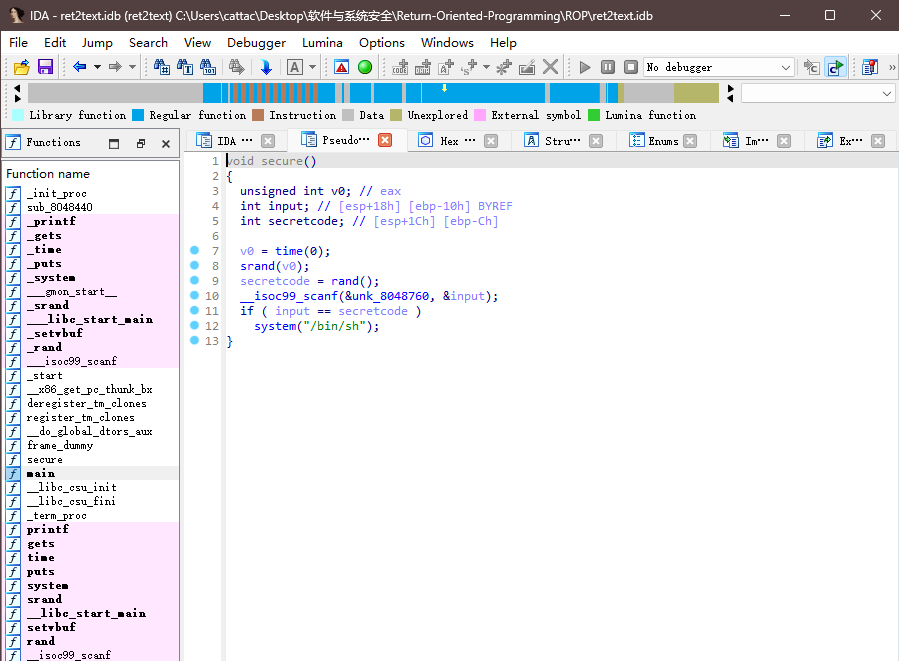


**下载并链接checksec**

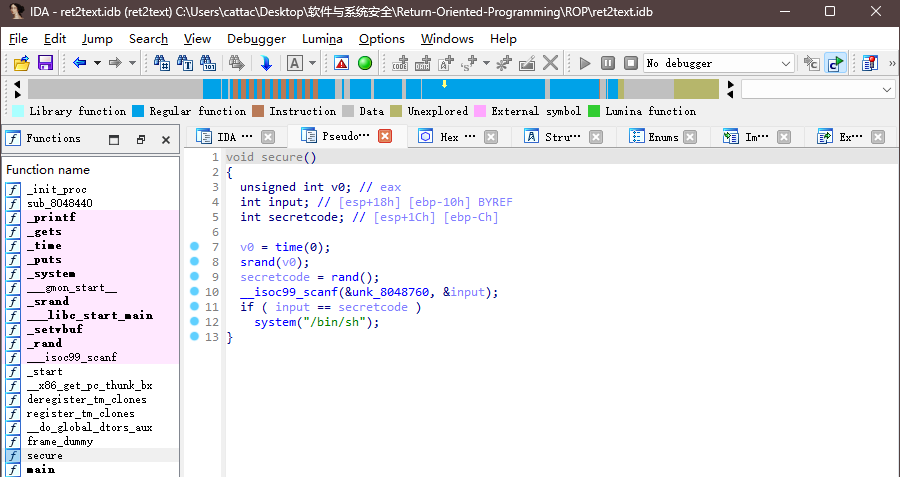


**Ret2text:**

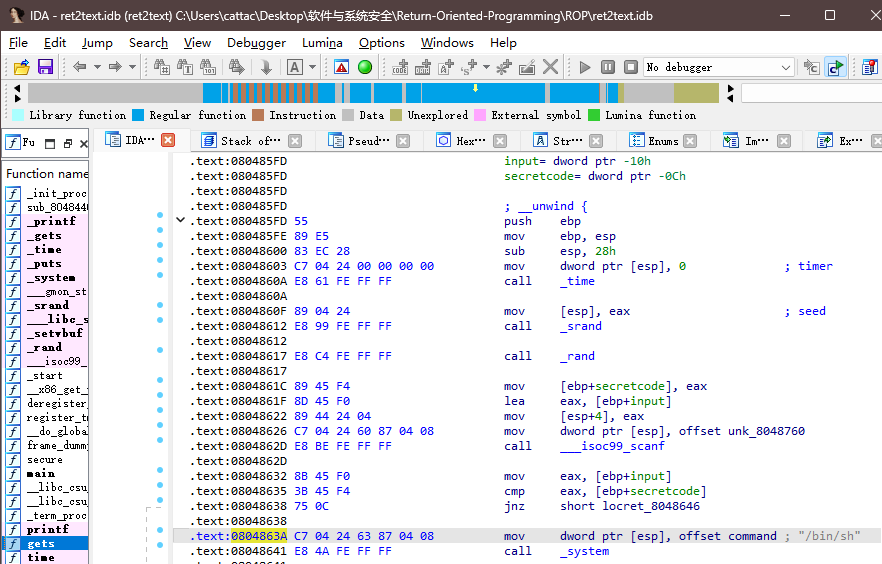
**运行checksec**

**使用IDA** **Pro分析代码**

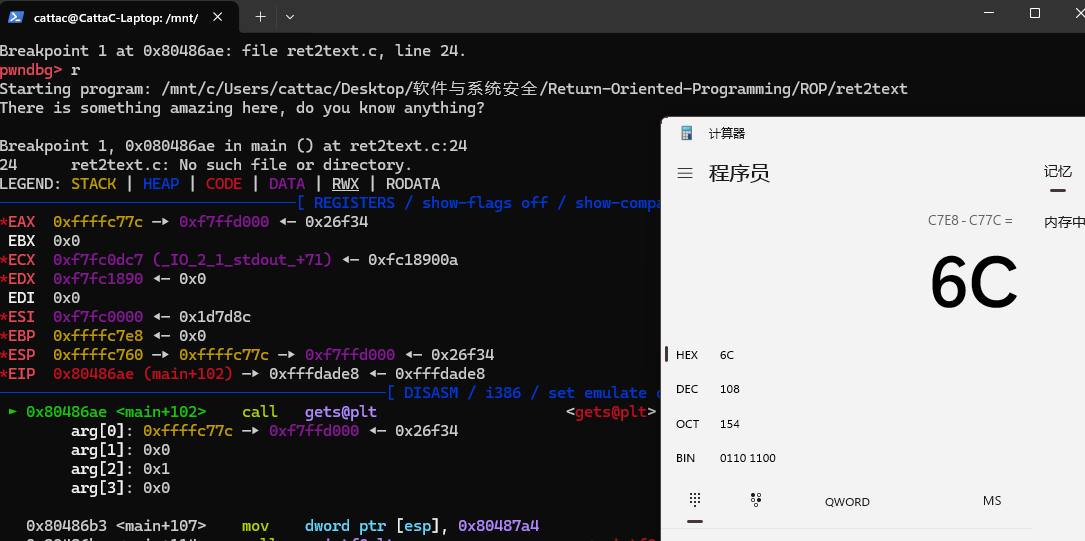
**发现调用shell的方式：使input和生成的随机数相同即可**



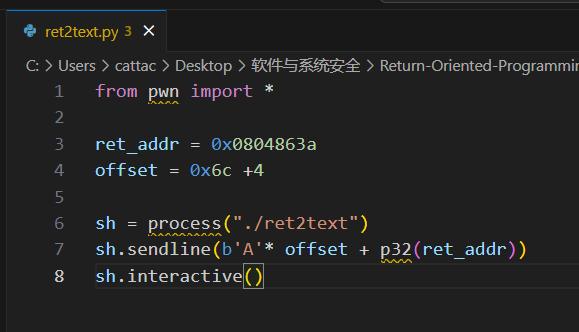
**查看汇编的地址内容，计算出input和secretcode的相对距离**



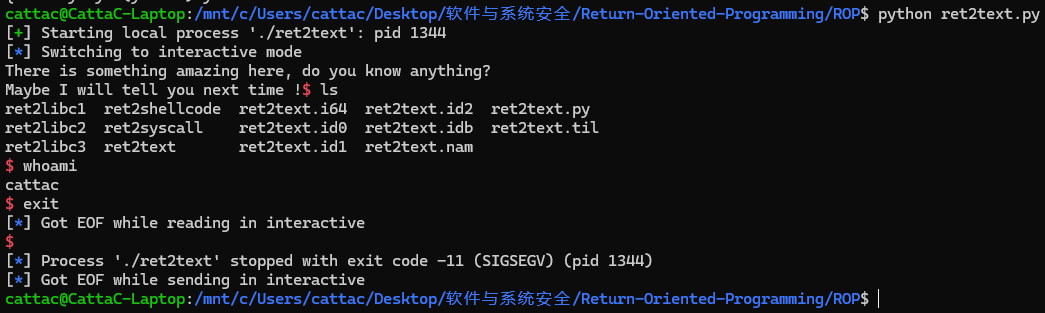
**查看栈长如图：**



**编写得到交互代码如图**

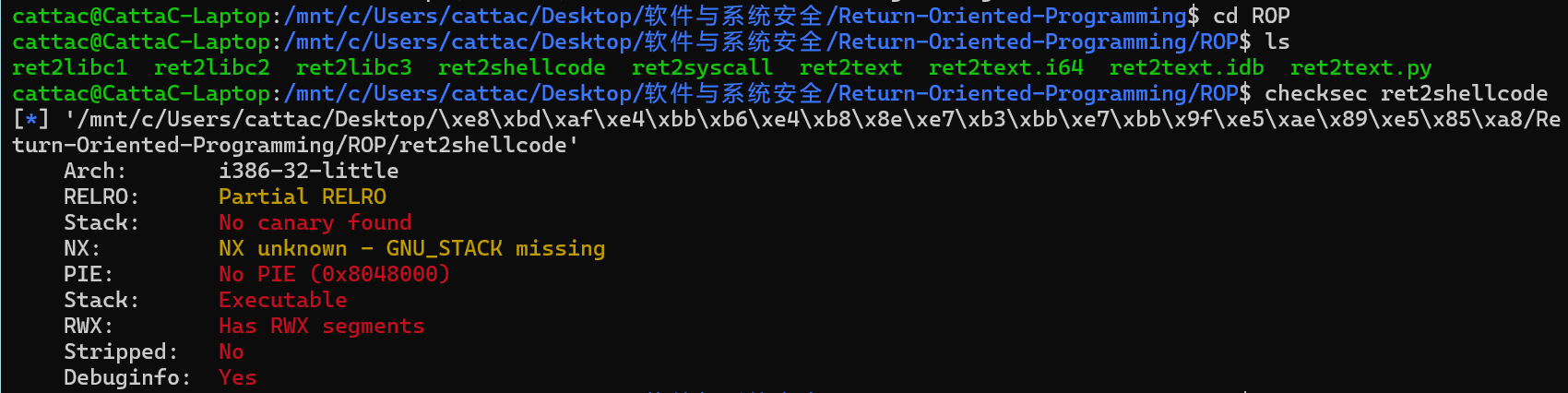


**运行结果如下：**

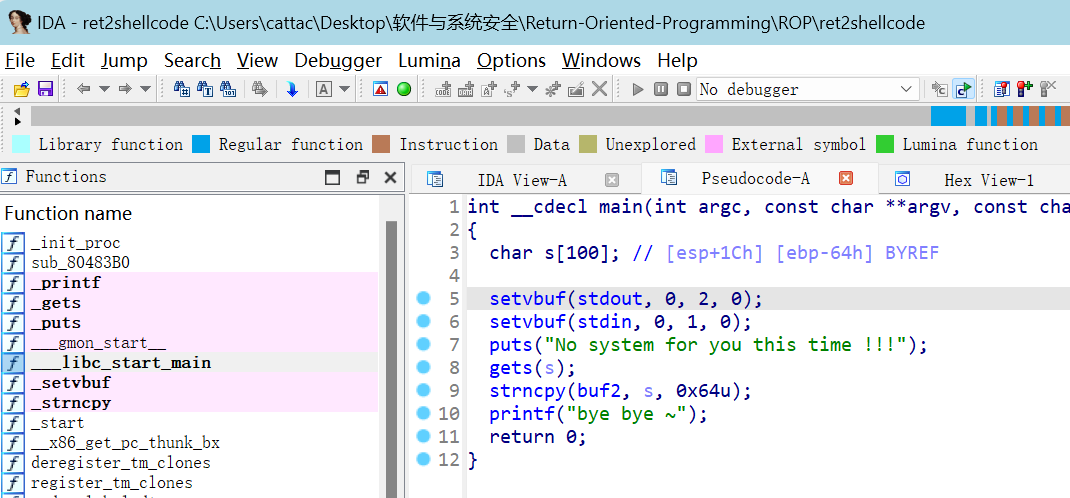


**Ret2shellcode**

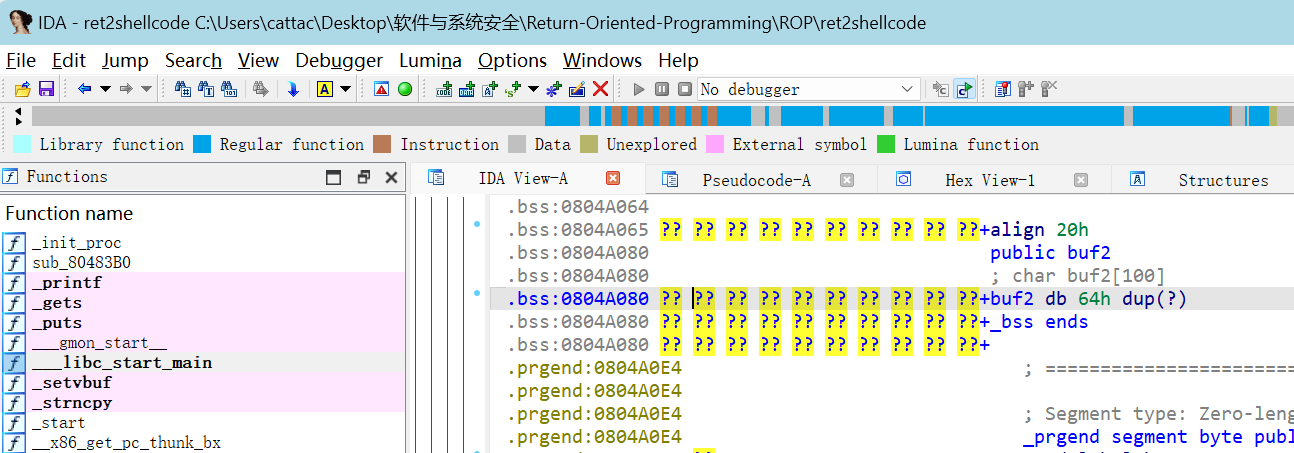
**使用checksec分析**



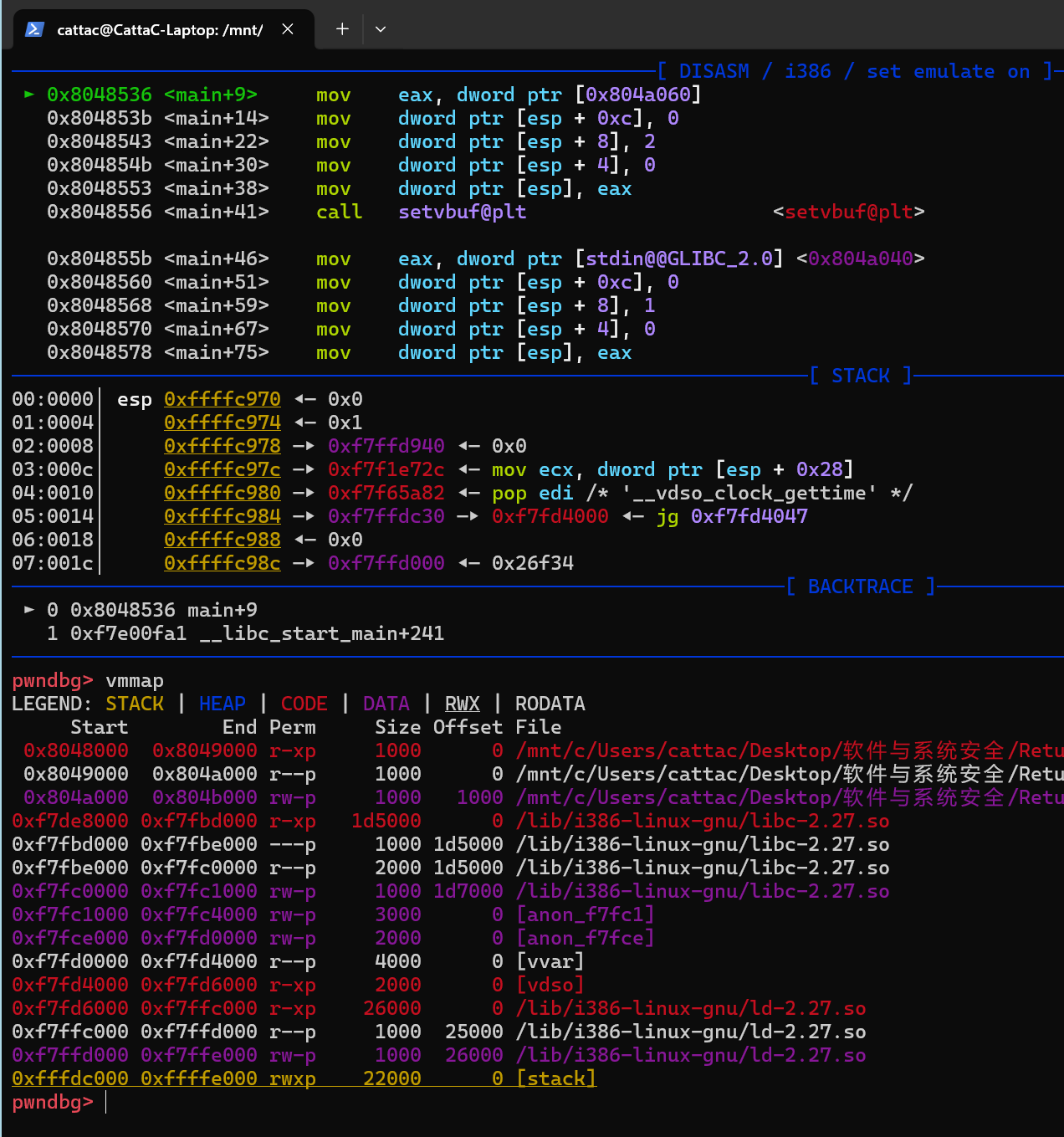
**查看反编译代码**



**可见buf2的位置**

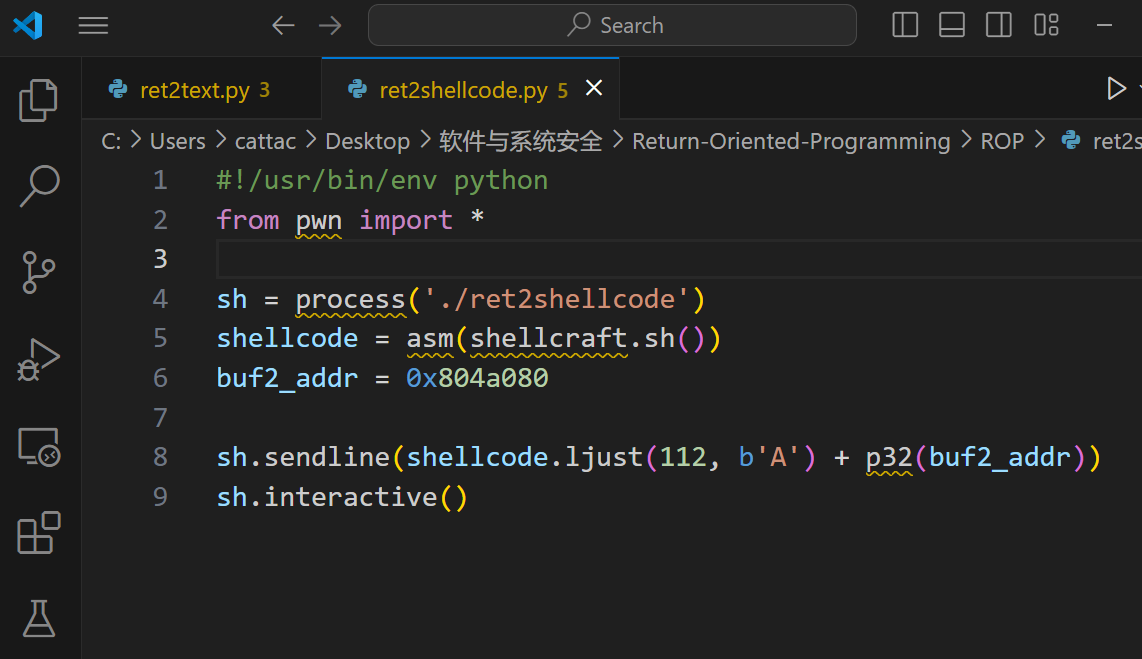


**使用pwndbg后，可见buf2所在位置没有shell的执行权限**

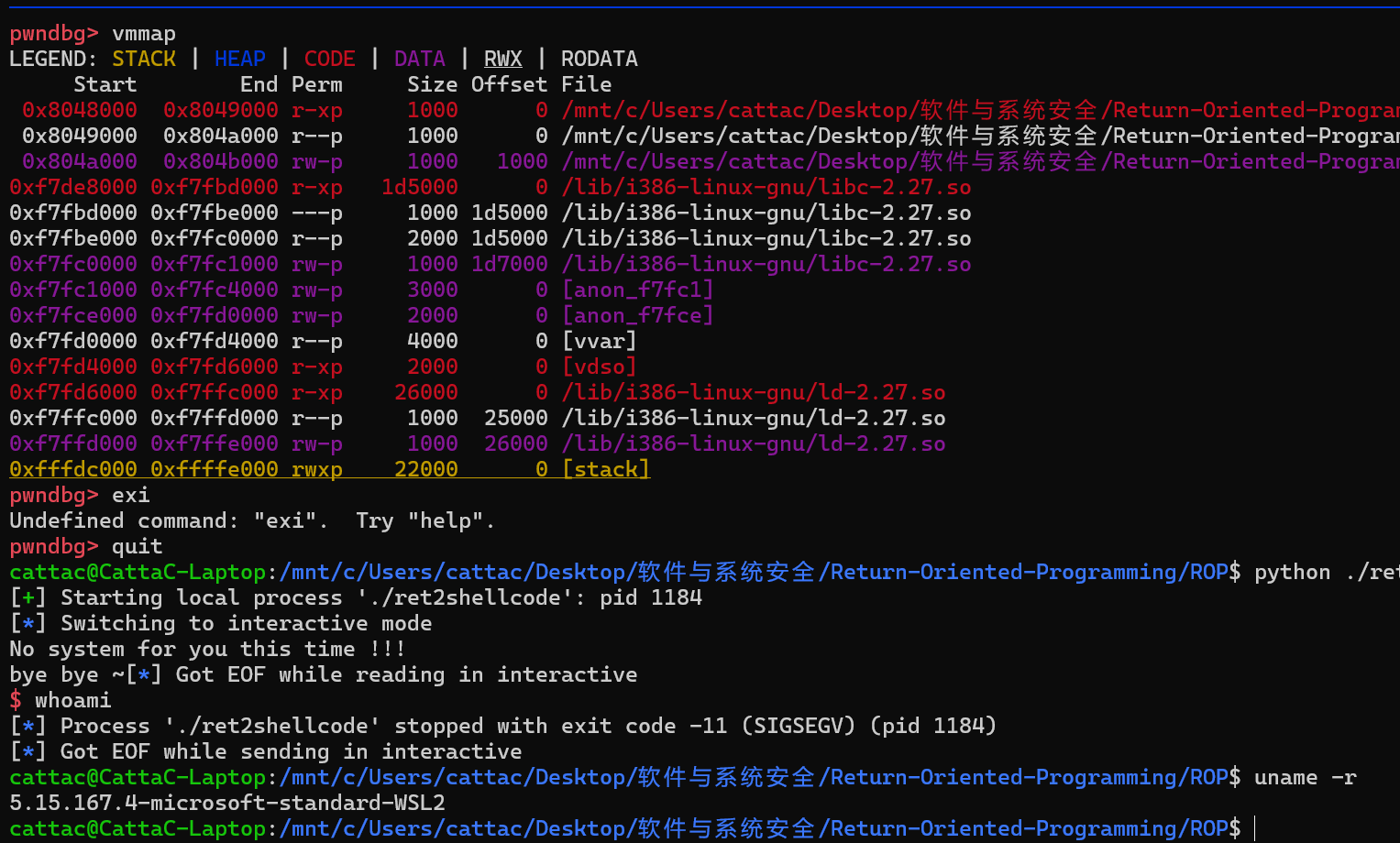


**（这里就没办法直接从.bass段注入了）**

**构建注入代码：**



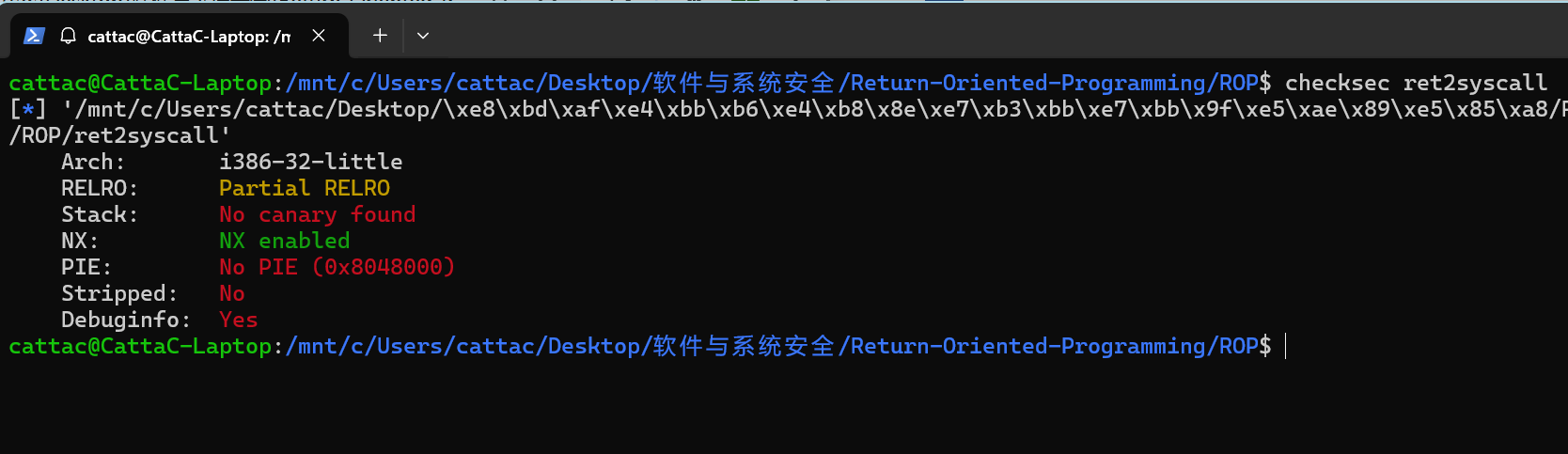
**仍然尝试使用.bass注入的方式，结果如下**



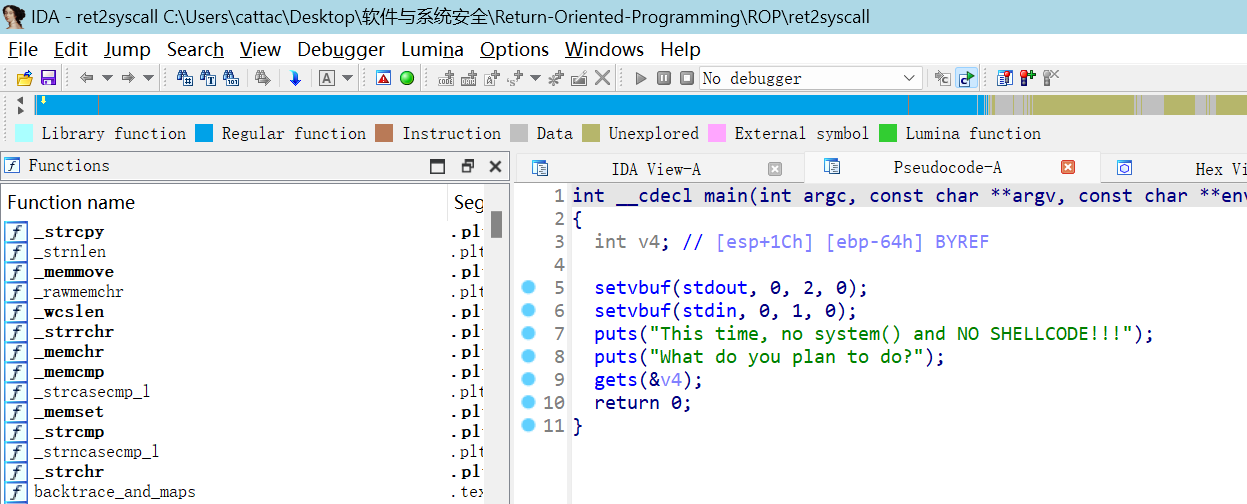
**可见失败的主要原因就是使用了WSL2 的系统内核，自动将.bss段保护了，因此攻击失败**

**Ret2syscall**

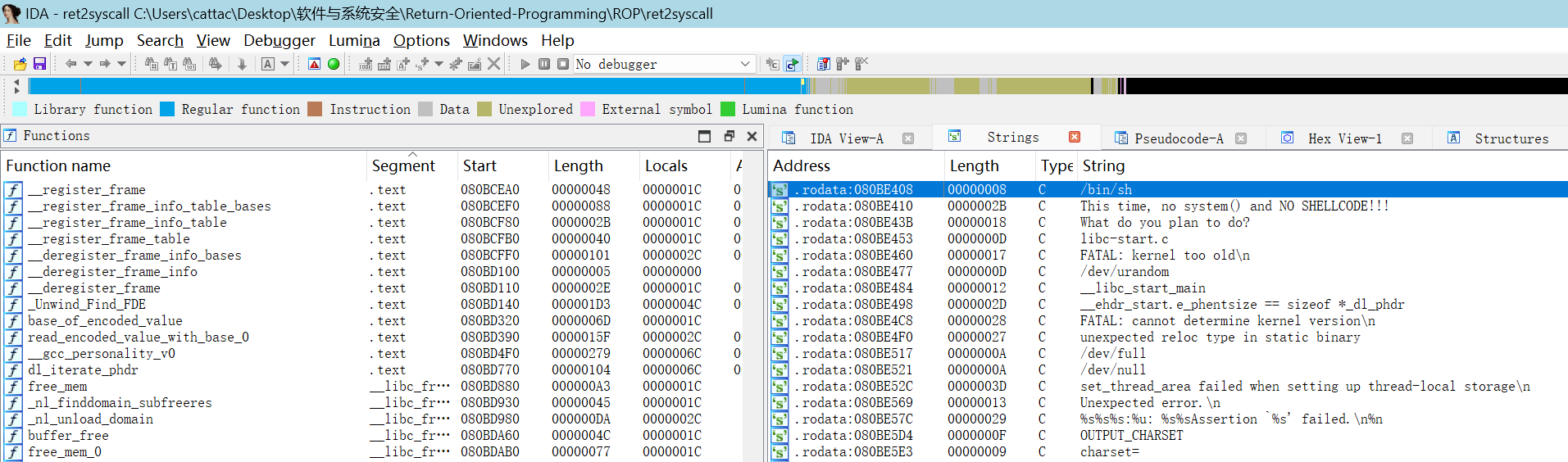
**首先进行checksec分析，可见NX保护开启**



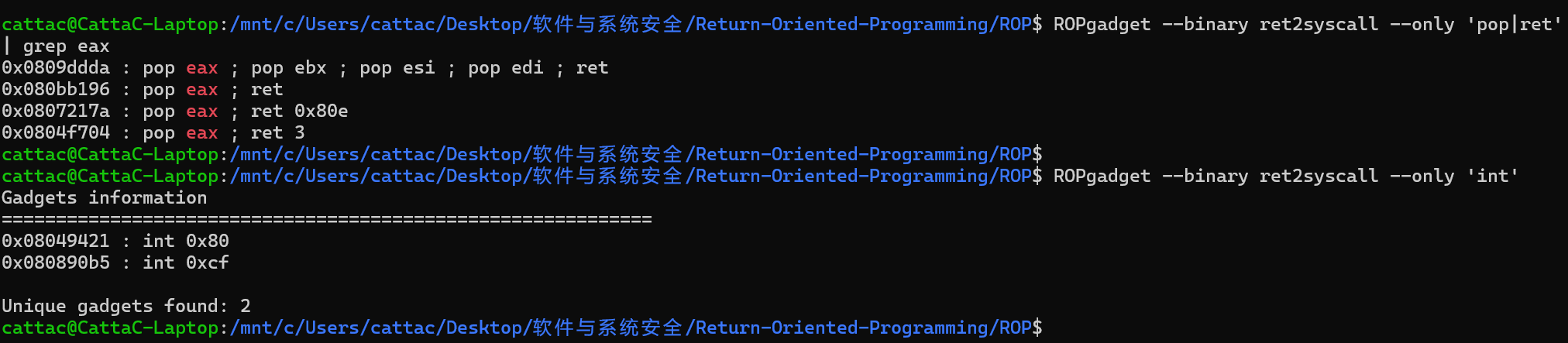
**反编译可执行文件，可见gets漏洞可利用：**

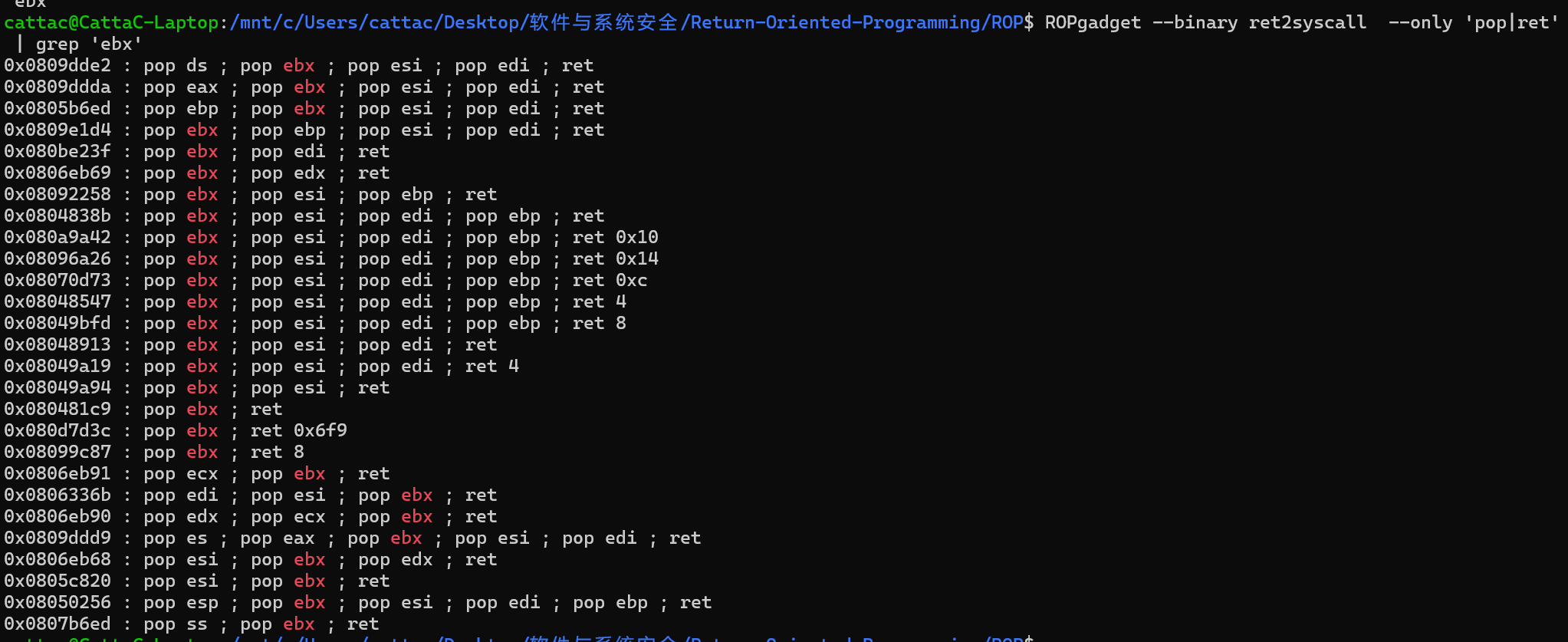


**在string的subview中可见/bin/sh：**



**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：**

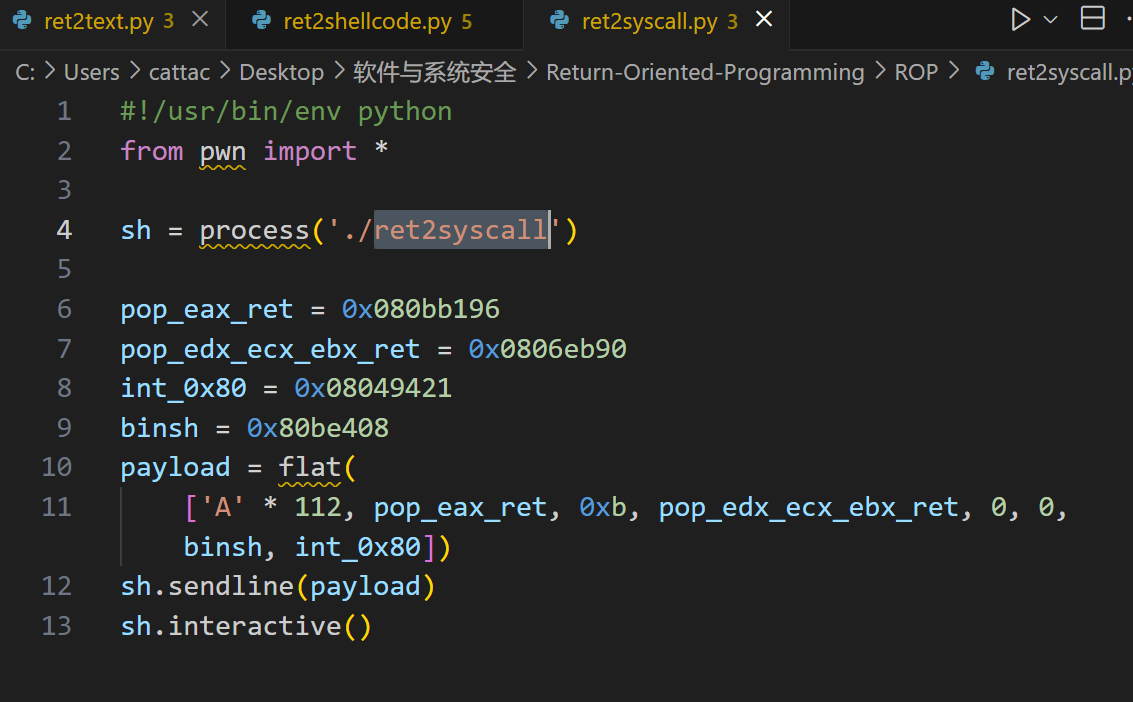
**0x080bb196 : pop eax ; ret**

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

**0x080be408 : /bin/sh**

**0x08049421 : int 0x80**

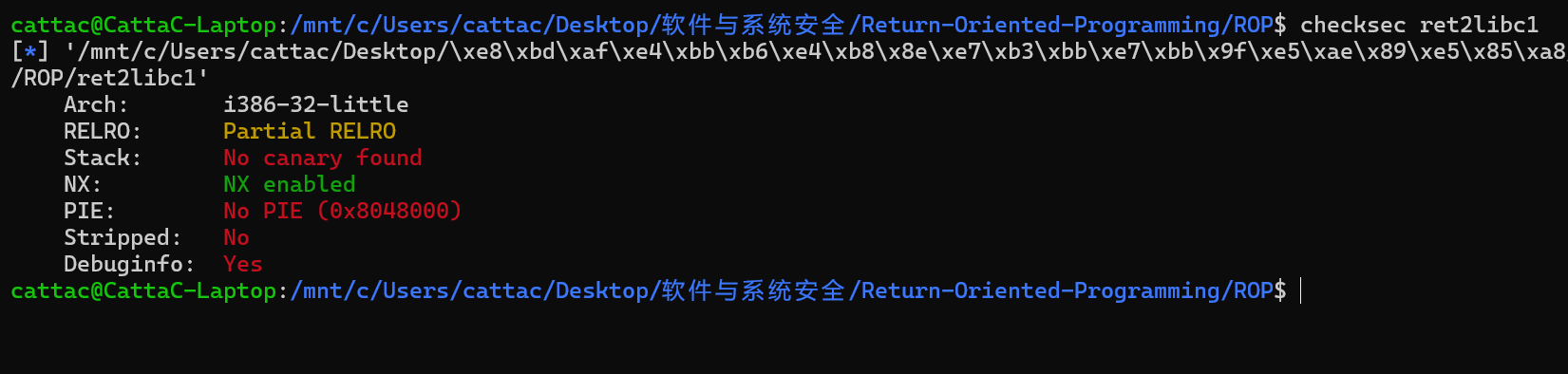
**编写代码**



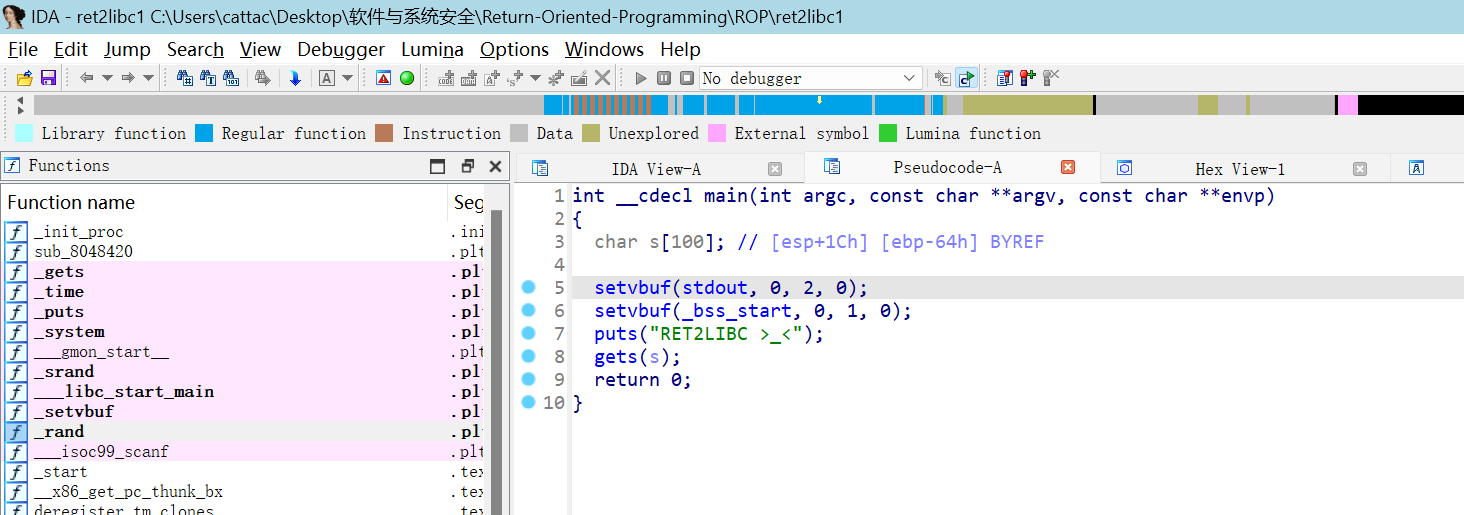
**运行结果**



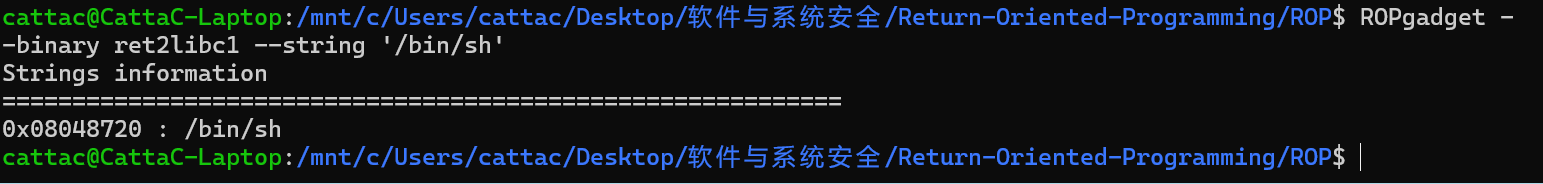
**ret2libc1同样是首先使用checksec进行保护模式的检测**



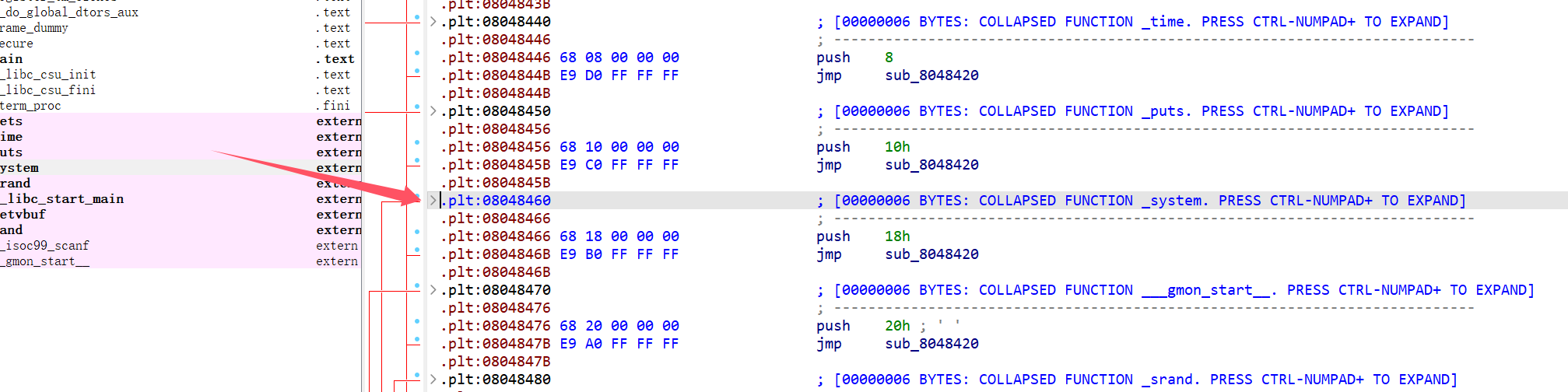
**接着使用IDA Pro进行反编译**



**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：**

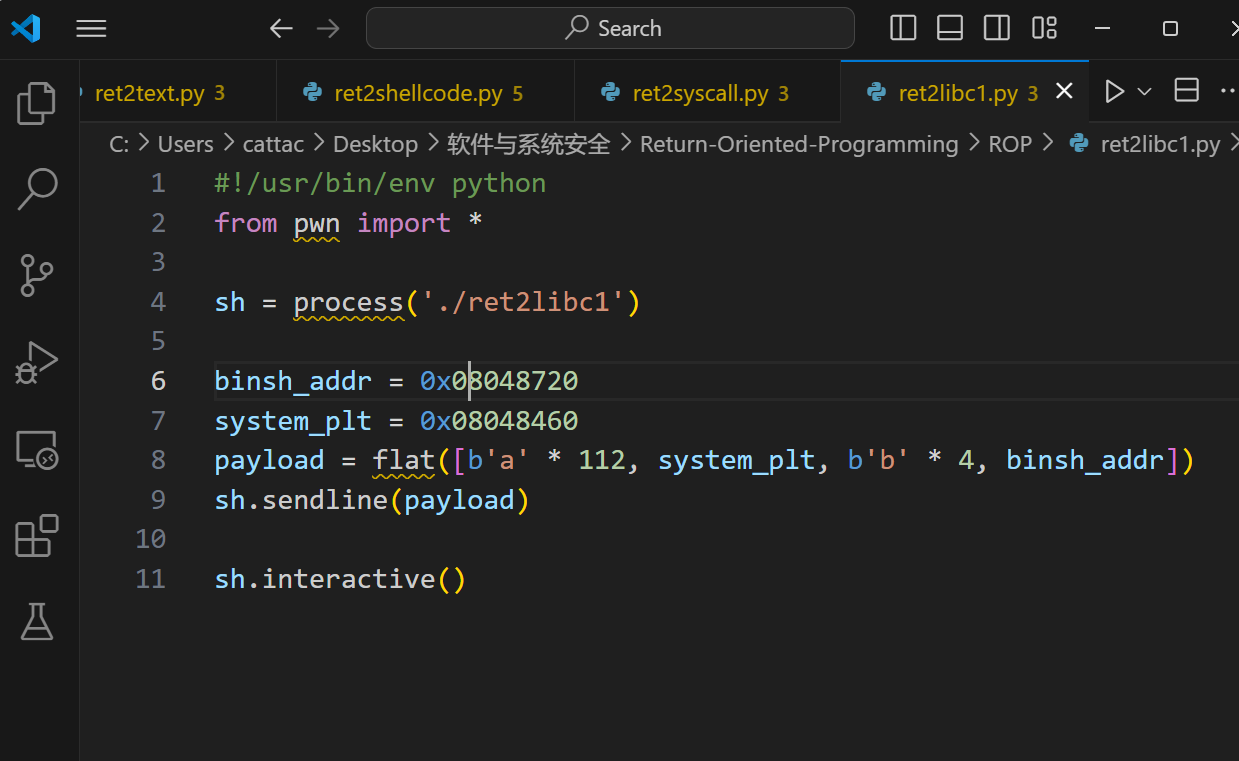


**可见，所需内容的位置如下：**

**0x08048720 : ‘/bin/sh’**

**0x08048460：system**

**编写代码如下：**



**运行结果如图：**

