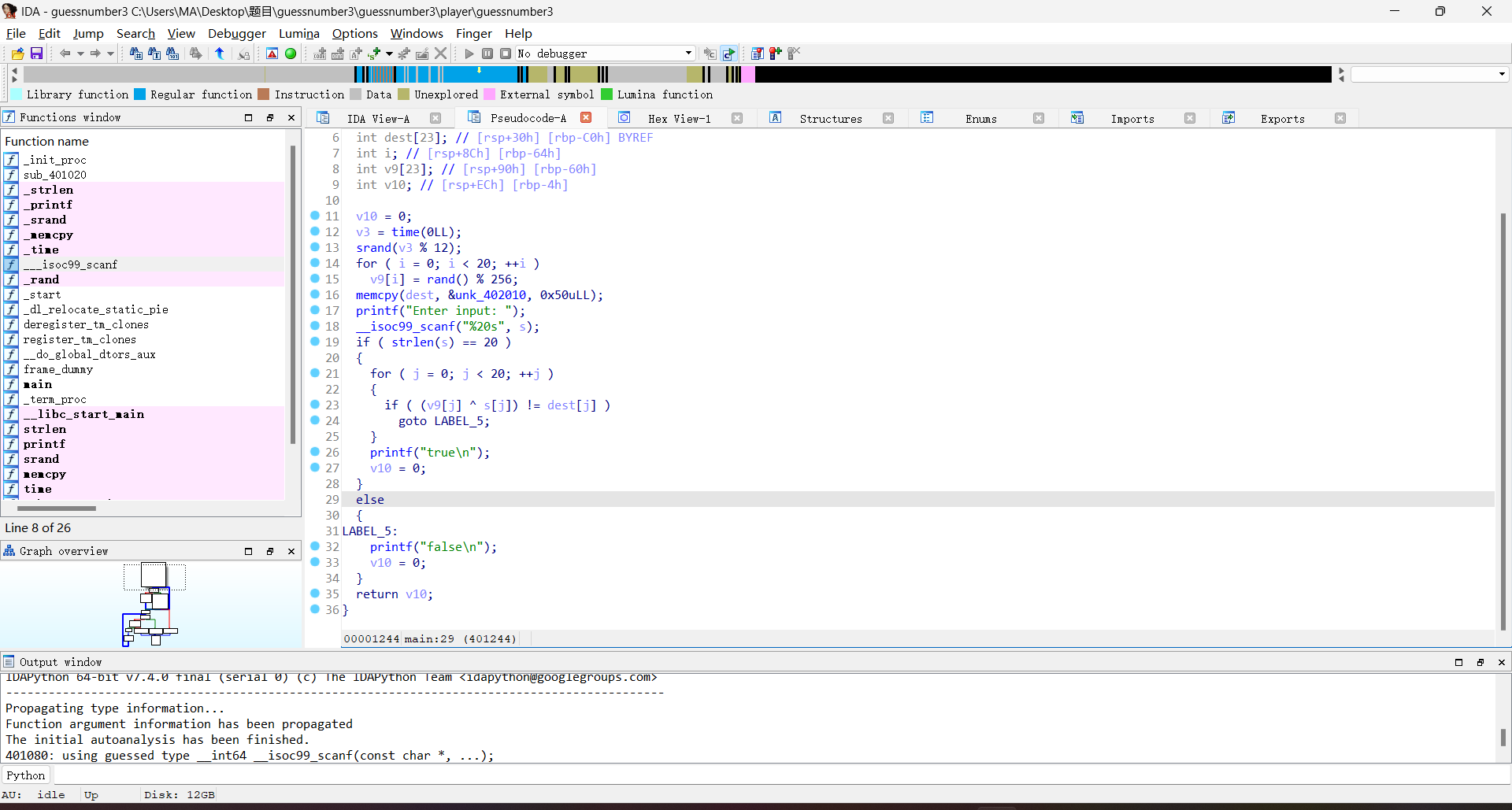
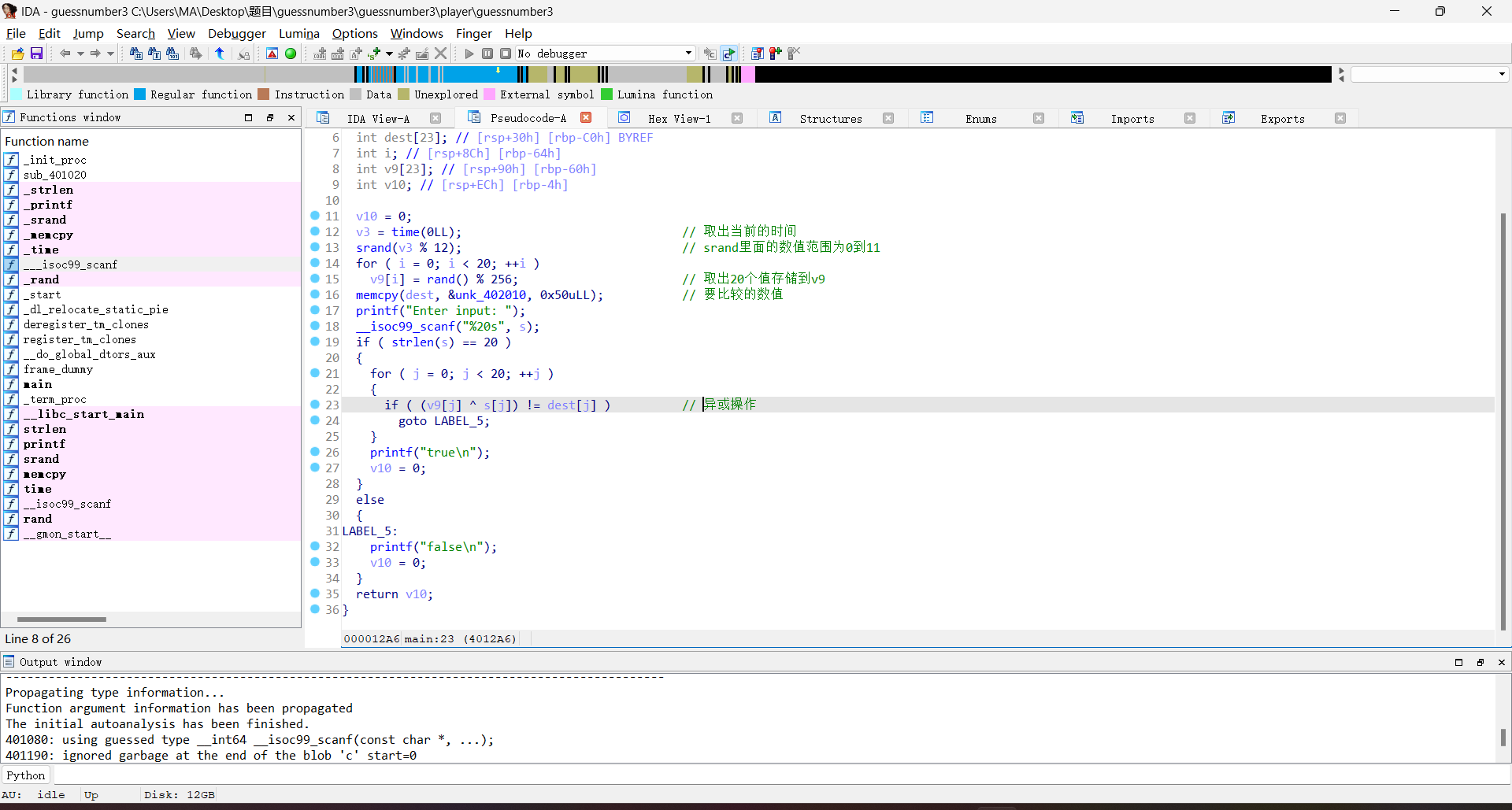
决赛一共出了四道题。其中的guessnumber3是简单题，obob和whichway是中等题，nothing为困难题。obob和whichway是仿照2023年强网杯强网先锋题目考点设计的，nothing涉及到了安卓逆向的脱壳操作，脱壳后与CTF安卓逆向无异。

1. guessnumber3

打开程序直接是主程序



题目逻辑如下

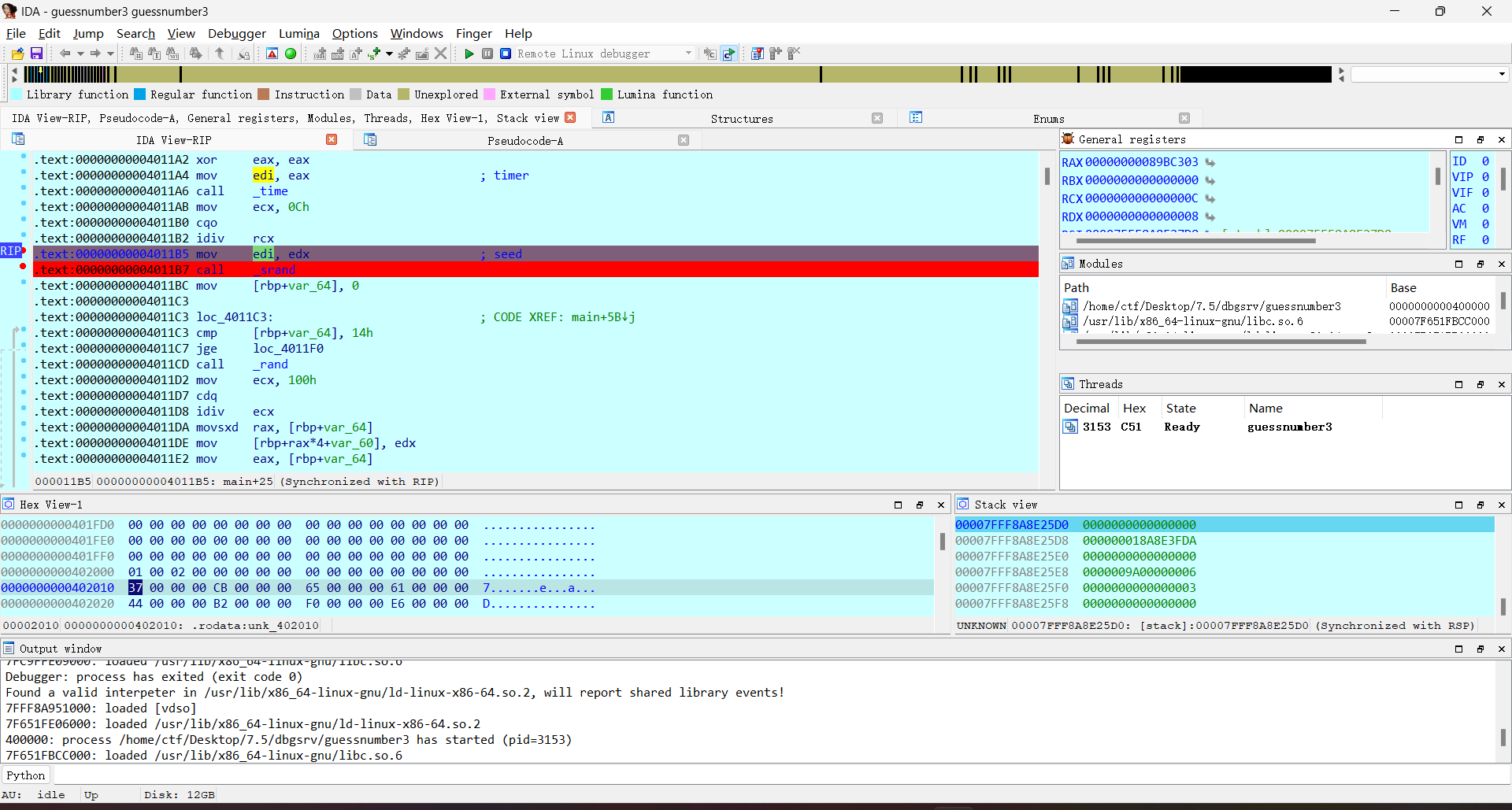


嫌麻烦可以在linux里面写一个c程序，srand分别取0到11爆破出所有值再异或，查看哪个是flag

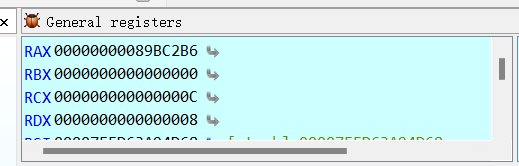
每次调试有1/12的概率调出来的是真flag，还是很良心(凉心 吧

本题是取的srand里面的值是10，即0A，使用ida动态调试

在srand处下断点，再调试到调用srand前面的位置

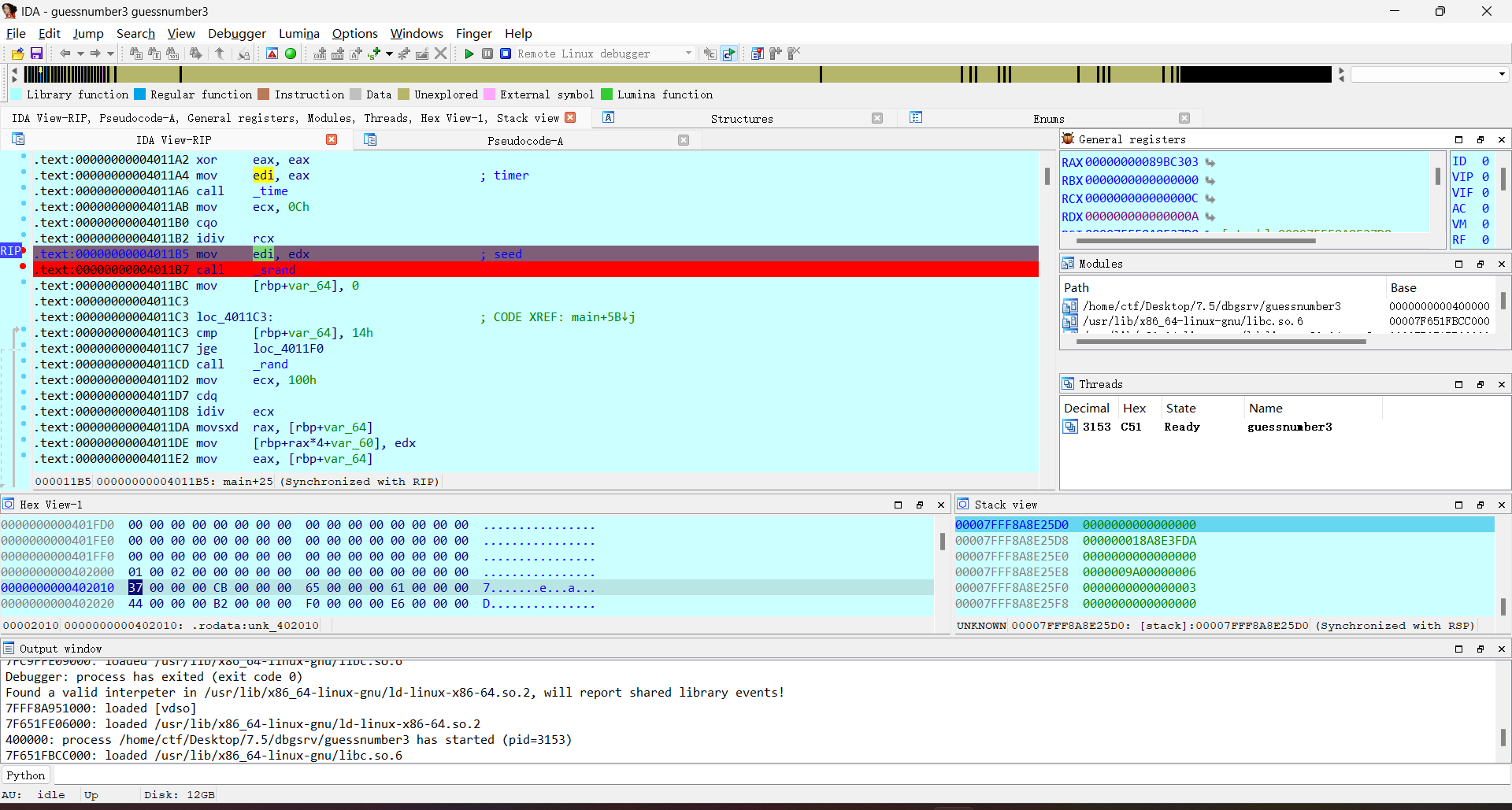


可以看到在call前面的寄存器是edi，就需要修改右上角的RDX的值



双击修改为0A即可

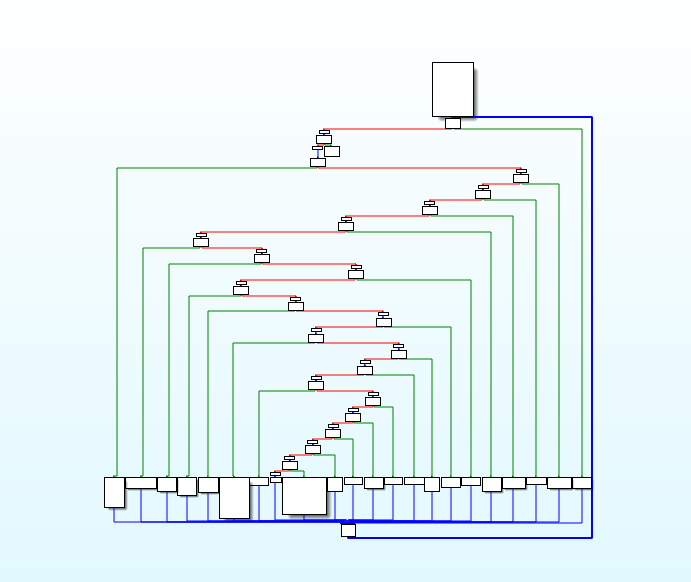
这样调试出的就是正确flag





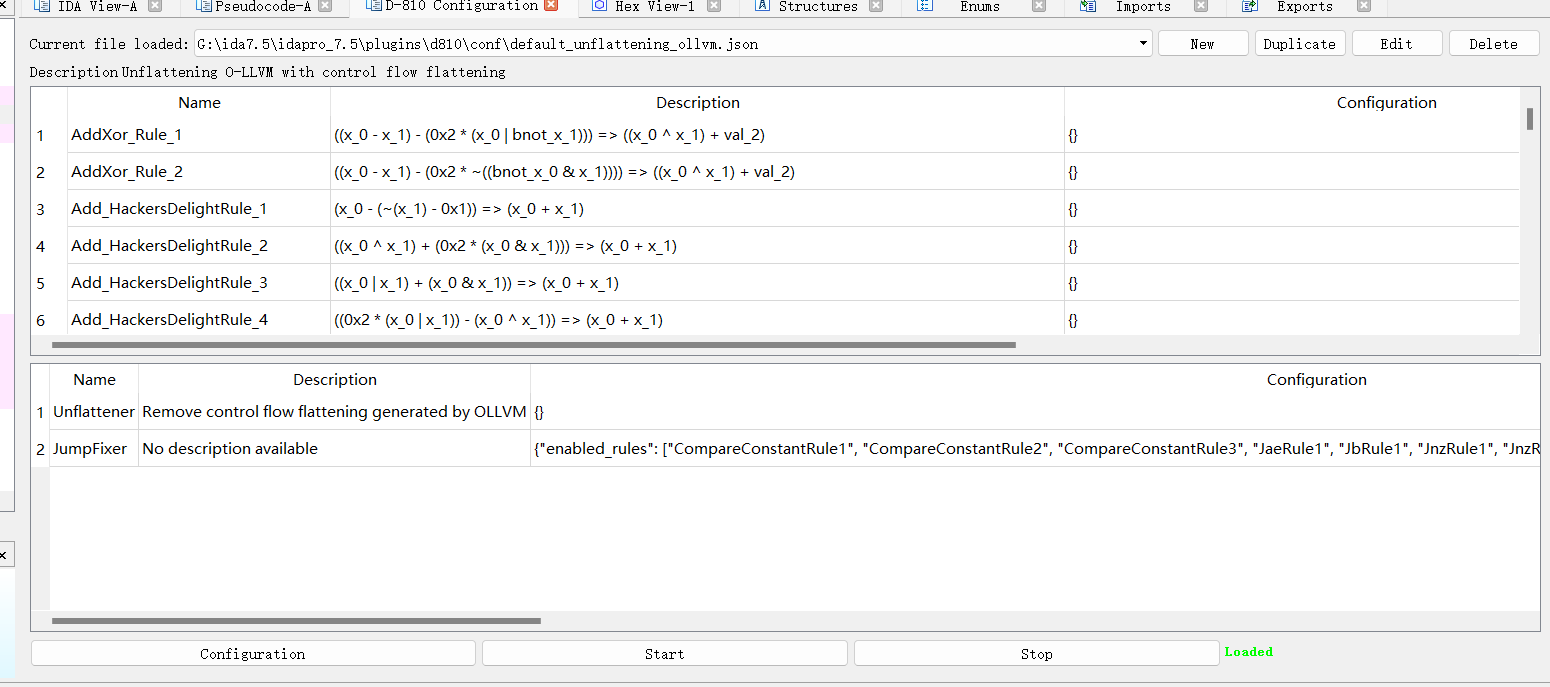
1. obob

在提示过程中给出可以使用D810插件去除控制流混淆，这里安装过程省略

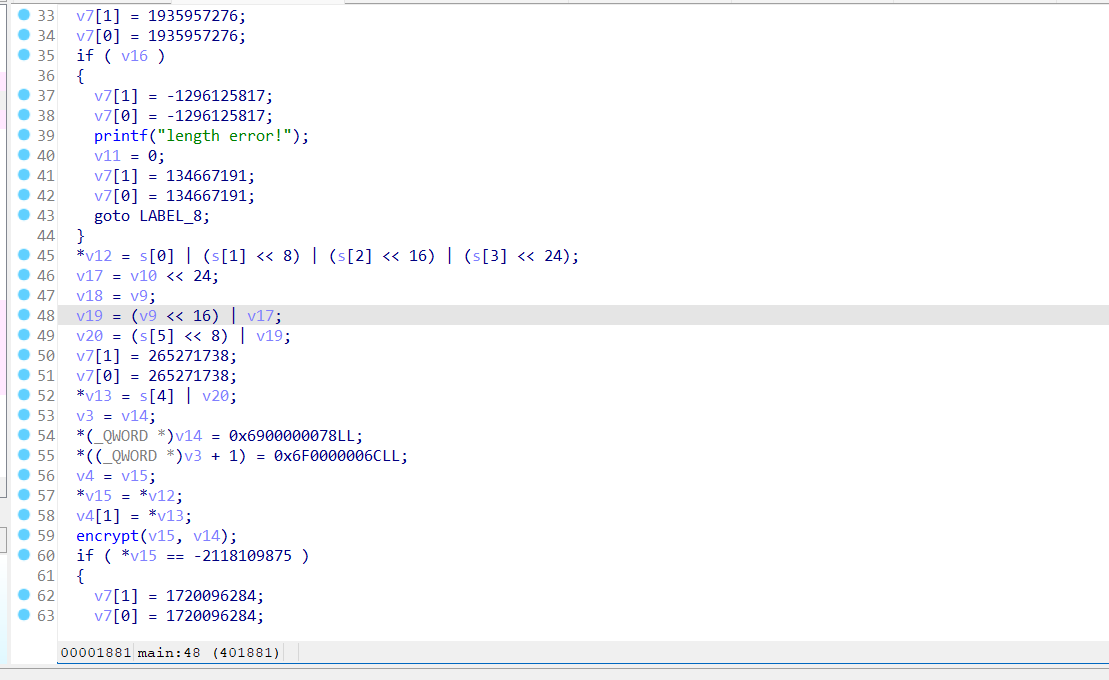


标准的控制流平坦化

Ctrl+shift+D呼出页面，去混淆即可



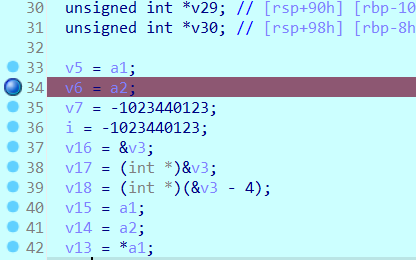
去混淆后的代码可读性已经很高了

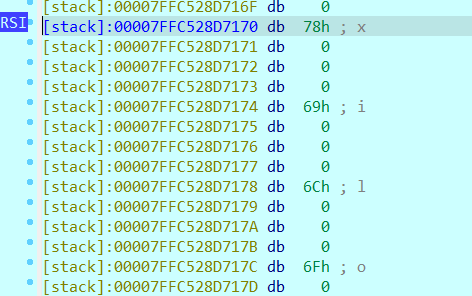


代码逻辑如下



在encrypt里面a2是key的值，下断看一下





取出后注意轮数即可写解密脚本

#include <stdio.h>

#include <stdint.h>

//加密函数

*void* encrypt(uint32\_t\* *v*, uint32\_t\* *k*) {

    uint32\_t v0 = *v*[0], v1 = *v*[1], sum = 0, i;

    uint32\_t delta = 0x9e3779b9;

    uint32\_t k0 = *k*[0], k1 = *k*[1], k2 = *k*[2], k3 = *k*[3];

    for (i = 0; i < 31; i++) {

        sum += delta;

        v0 += ((v1 << 4) + k0) ^ (v1 + sum) ^ ((v1 >> 5) + k1);

        v1 += ((v0 << 4) + k2) ^ (v0 + sum) ^ ((v0 >> 5) + k3);

    }

*v*[0] = v0; *v*[1] = v1;

}

//解密函数

*void* decrypt(uint32\_t\* *v*, uint32\_t\* *k*) {

    uint32\_t v0 = *v*[0], v1 = *v*[1], sum = 0xC6EF3720, i;

    uint32\_t delta = 0x9e3779b9;

    sum=0x9e3779b9\*31;

    uint32\_t k0 = *k*[0], k1 = *k*[1], k2 = *k*[2], k3 = *k*[3];

    for (i = 0; i<31; i++) {

        v1 -= ((v0 << 4) + k2) ^ (v0 + sum) ^ ((v0 >> 5) + k3);

        v0 -= ((v1 << 4) + k0) ^ (v1 + sum) ^ ((v1 >> 5) + k1);

        sum -= delta;

    }

*v*[0] = v0; *v*[1] = v1;

}

*int* main()

{

    // v为要加解密的数据，两个32位无符号整数

    uint32\_t v[2] = {0x626f7a65,0x6165745f };

    // k为加解密密钥，4个32位无符号整数，密钥长度为128位

    uint32\_t k[4] = { 0x78,0x69,0x6c,0x6f };

*int* n = sizeof(v) / sizeof(uint32\_t);

    printf("加密前原始数据:0x%x 0x%x\n", v[0], v[1]);

    encrypt(v, k);

    printf("加密后的数据:0x%x 0x%x\n", v[0], v[1]);

    decrypt(v, k);

    printf("解密后的数据:0x%x 0x%x\n", v[0], v[1]);

    for (*int* i = 0; i < n; i++)

    {

        for (*int* j = 0; j < sizeof(uint32\_t)/sizeof(uint8\_t); j++)

        {

            printf("%c", (v[i] >> (j \* 8)) & 0xFF);

        }

    }

    printf("\n");

    return 0;

}

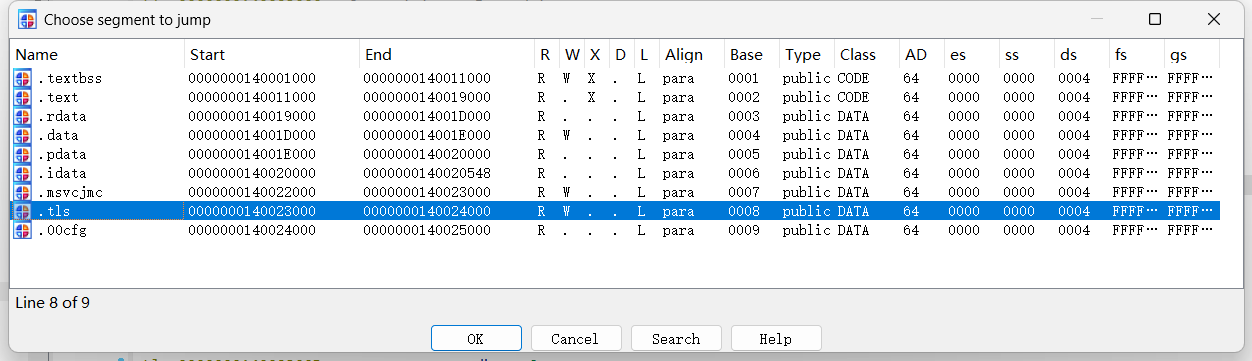
//ezob\_tea

1. whichway

[【逆向工程核心原理：TLS回调函数】-CSDN博客](https://blog.csdn.net/qq_42363151/article/details/130686318)

本题使用了callback回调函数考点，如果你问 我怎么知道这题考点是什么呢，这就需要做题来积累了，正常调试调不过去得不到答案，那么就要考虑其他考点了

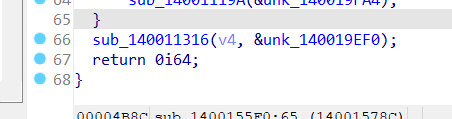
Ctrl+s找到tls，再交叉引用跳转到TlsCallback\_0\_0中



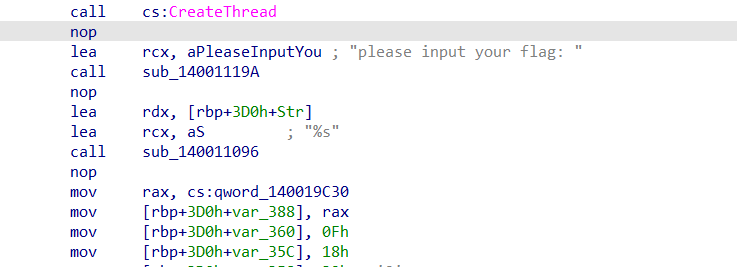


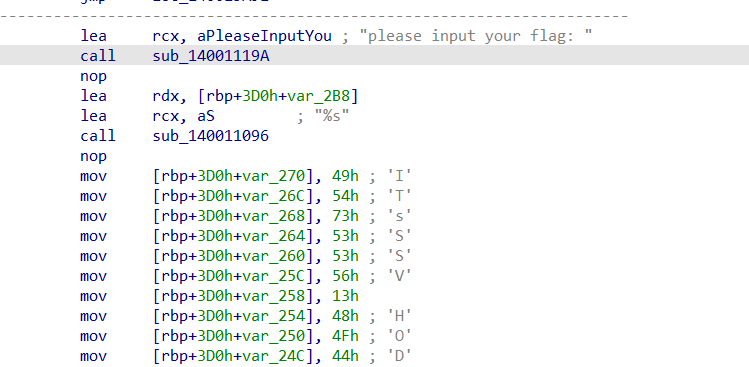
这个函数是会先检测程序是否被调试，如果没有被调试就会对一些位置nop掉，可以下断点动态调试查看具体位置是哪里，这里不再演示。

其实就是把最后的return 0 nop掉了，因为后面还有代码



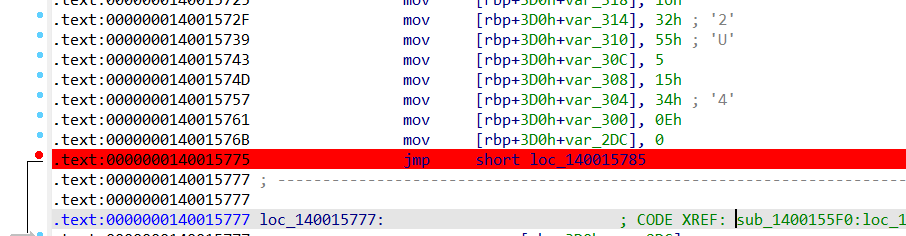
这个时候看汇编就很有用了



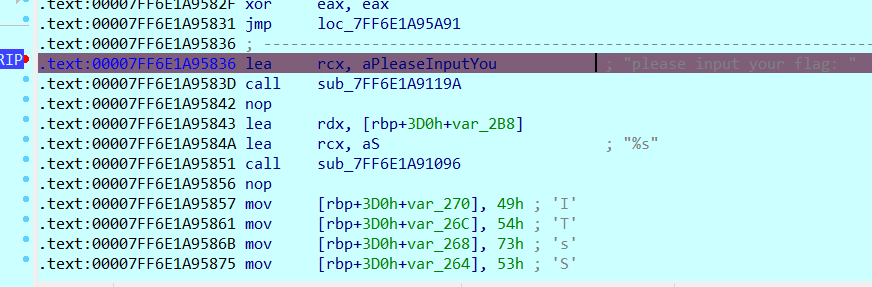


可以看到有两个位置可以输入flag

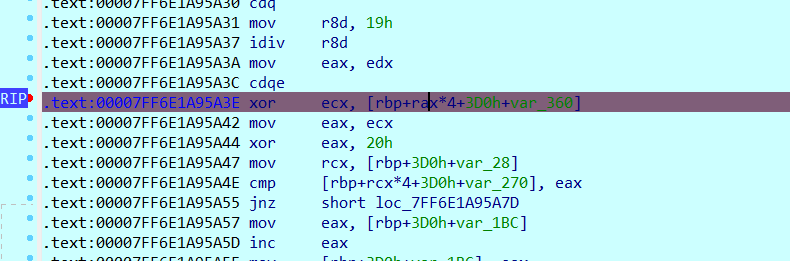
经过调试可以知道前一个是fakeflag，后面是真的，在前面mov完后可以直接跳到后面的inputflag部分



这里下断点，运行到这里后直接setip到下面位置



后向下运行，在下面位置可以找到异或的值



先异或[rbp+rax\*4+3D0h+var\_360]，再异或0x20

后可直接写脚本

arr1=[*0x*0000000F, *0x*00000018, *0x*00000032, *0x*00000014, *0x*00000008, *0x*00000019, *0x*0000005B, *0x*00000000, *0x*00000007, *0x*0000000C, *0x*0000001A, *0x*00000000, *0x*00000032, *0x*00000052, *0x*00000036, *0x*00000012, *0x*00000032, *0x*00000018, *0x*00000016, *0x*00000032, *0x*00000055, *0x*00000005, *0x*00000015, *0x*00000034, *0x*0000000E]

arr2=[*0x*00000049, *0x*00000054, *0x*00000073, *0x*00000053, *0x*00000053, *0x*00000056, *0x*00000013, *0x*00000048, *0x*0000004F, *0x*00000044, *0x*00000065, *0x*00000059, *0x*0000007D, *0x*00000007, *0x*00000049, *0x*00000054, *0x*0000007B, *0x*00000056, *0x*00000052, *0x*0000004D, *0x*00000001, *0x*0000004D, *0x*00000050, *0x*0000004B, *0x*0000005A, *0x*0000005D, *0x*0000004D, *0x*00000077, *0x*0000006B, *0x*0000005F, *0x*00000058, *0x*00000002, *0x*00000001, *0x*00000006, *0x*0000000D, *0x*0000001B, *0x*0000005D]

flag=''

for i in range(len(arr2)):

    flag+=chr(arr1[i%len(arr1)]^arr2[i]^32)

print(flag)

1. nothing

本题是按照防ak题目出的，

在java层与native层均有加密部分

使用360加壳时未使用检测环境，来保证可以使用LSPosed插件进行脱壳。

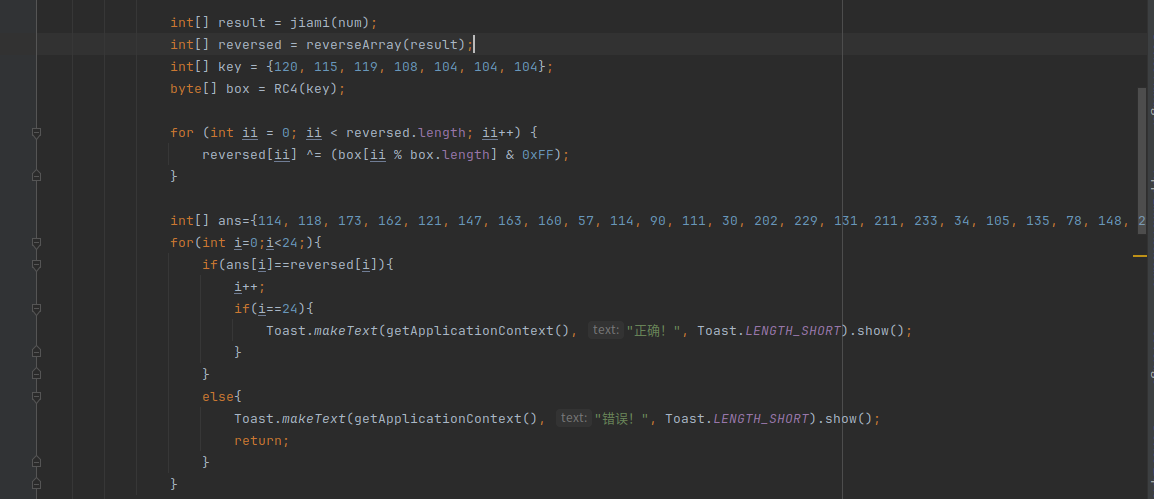
注:本题需要真机 root后安装面具以及LSPosed插件，如果有其他方法脱壳亦可

使用FunDex进行脱壳



选中后再启动Fundex选中app，再启动app就会在/data/data/com……/里面生成dex

后通过adb取出使用jadx或用MT管理器(需要会员)直接查看代码即可



源码部分java层为异或操作，应该无法使用jeb动态调试，所以需要自写代码进行正向模拟生成box

Native层是xxtea加密，可以直接打开lib里面进行查看

再进行解密即可