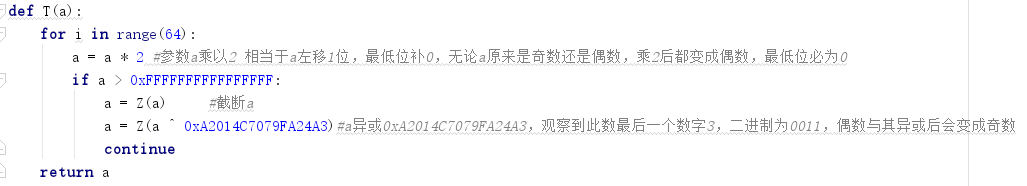
Re

易之蟒蛇

.py文件，pycharm打开



此题目考察的是移位运算和异或运算，题目给出“注意奇偶数”的提示，所以要往这方面靠。



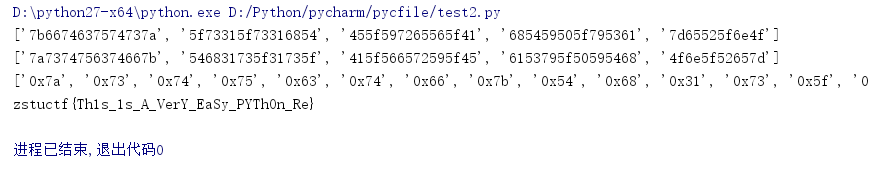
这个函数是重点分析的地方（分析如图上注释）。输入按每8个一组分组后进行运算的结果，也就是data列表。如果在64次循环的最后一次运算中，a与0xA2014C7079FA24A3异或，则a变成奇数，保存在结果列表中。

观察data列表，可以看到有奇数也有偶数，对data列表元素进行逆运算。



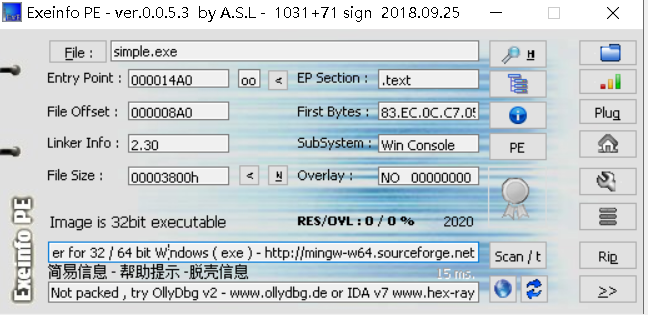


执行结果如下

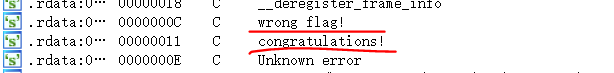


易之异或

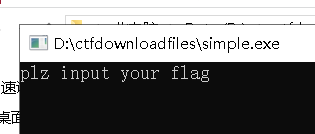
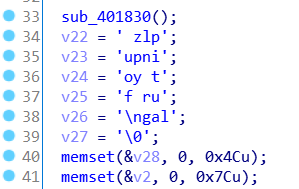
32位程序



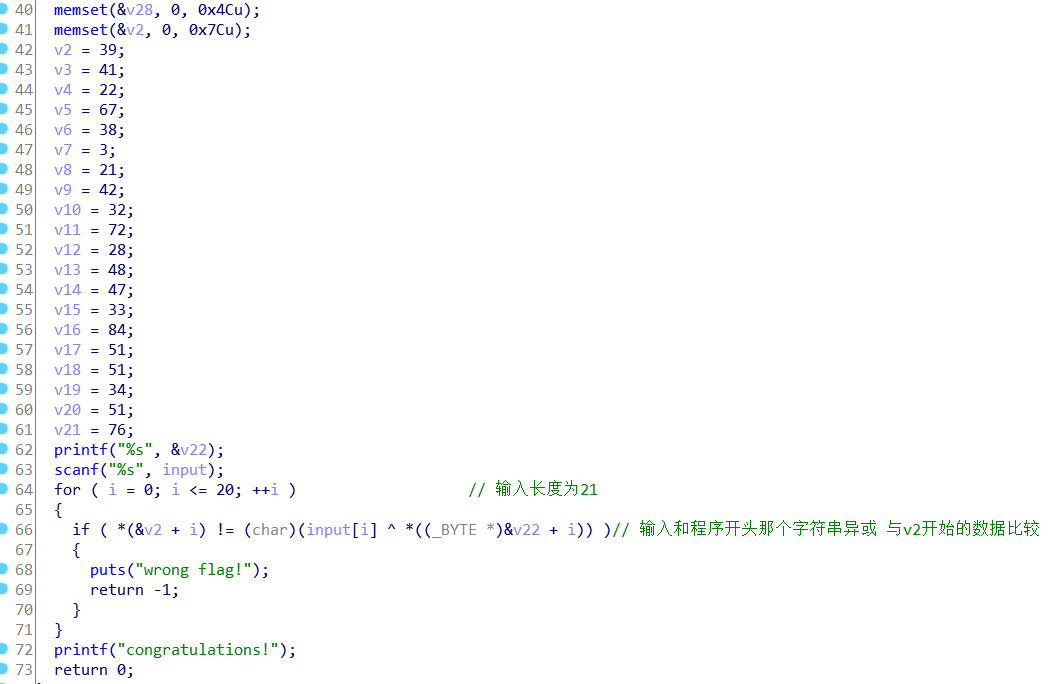
Ida打开，函数窗没发现什么，shift+F12打开字符串窗，发现



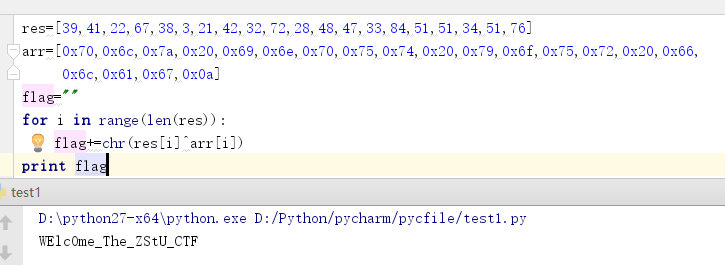
双击，x交叉引用跟过去，来到sub\_4015C0函数（函数名可能不同）。函数开头一段数字，r转换成字符，h可以换回去，发现和运行程序时输出的字符串相同，说明来到了正确的主函数入口



往下是一堆数据和循环判断操作，就是此题目的主逻辑了



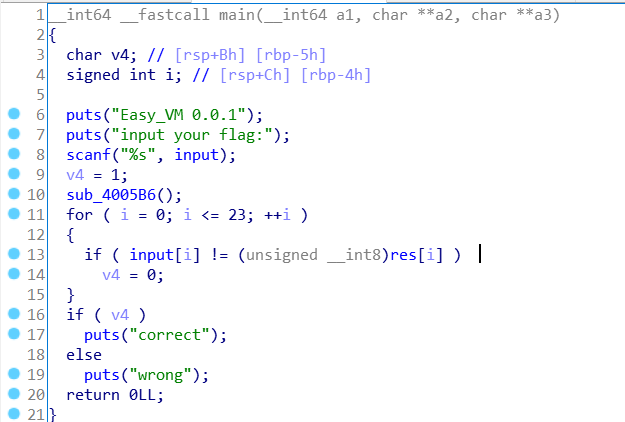
逻辑清晰易懂，直接写脚本，需要注意的是程序开头的字符串在ida中是小端序形式，需要变换一下，脚本如下



提交的时候包上zstuctf{}

易之虚机

从文件名看应该是vm类型的题目，主要是找opcode和主逻辑运算（一般是switch/case结构的运算）。Ida打开，发现main函数，F5反编译



可以看到，最后是对输入和res数组进行比较，而res可以直接双击看到，并不是有意义的内容，所以程序一定对输入在原地址处进行了变化，进入sub\_4005B6函数

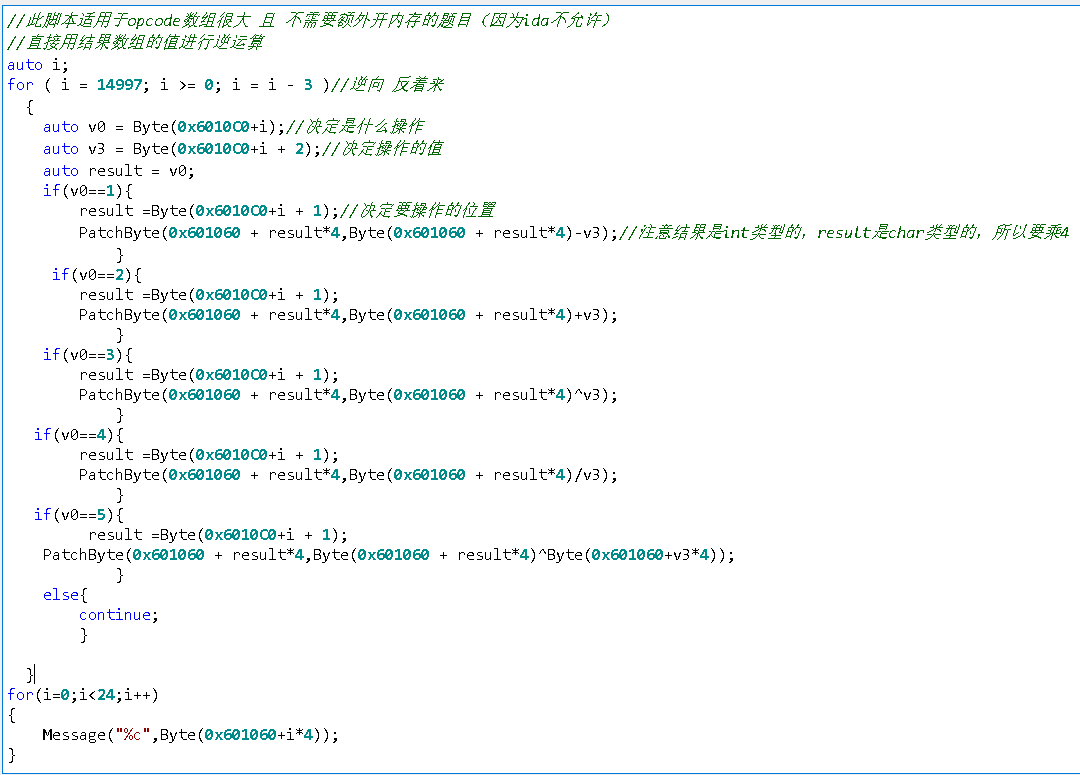


可以看到，输入input的确被修改了。Opcode双击进去看，是大小为15000的char型数组。分析这个循环体，其实是5000次循环，每次循环在opcode数组中找三个数，第一个数，也就是v0，决定了要进行什么运算，第三个数，也就是v3，决定了要运算的值，中间第二个数，也就是result，决定了对input中的什么位置的值进行操作。

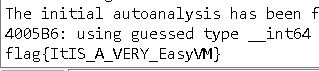
5种运算，并不复杂，可以直接写逆向脚本。对于要比较的res数组，大小不大，可以直接扣下来，但是，opcode数组的大小为15000，不好扣，所以考虑写idc脚本，直接利用ida数据库的值进行逆运算。百度之后发现idc脚本不能扩充空间，就是说不能定义数组。我们发现程序全程对res数组没有进行变换，就是一个比较的作用，它正好就是逆运算的输入。Idc不能定义数组，但可以patch原本地址的内容，也就是PatchByte()函数，原型为：



Idc脚本如下

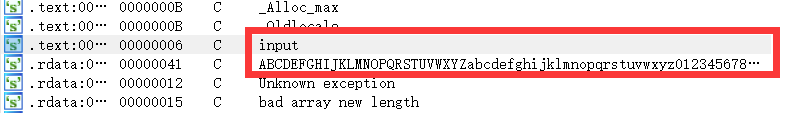


执行脚本结果如下

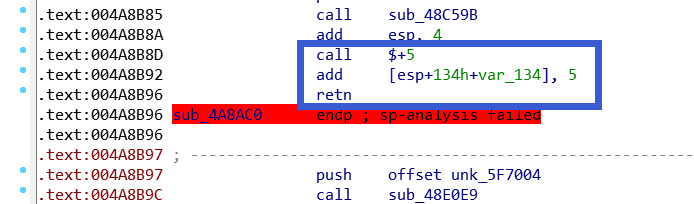


易之迪比

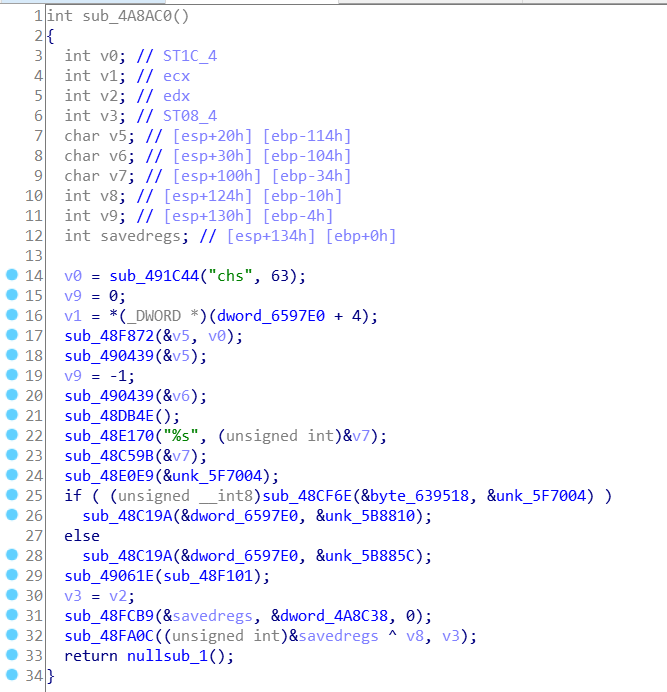
32位程序，无壳，ida打开，函数窗没找到什么有用的东西，打开字符串窗，发现



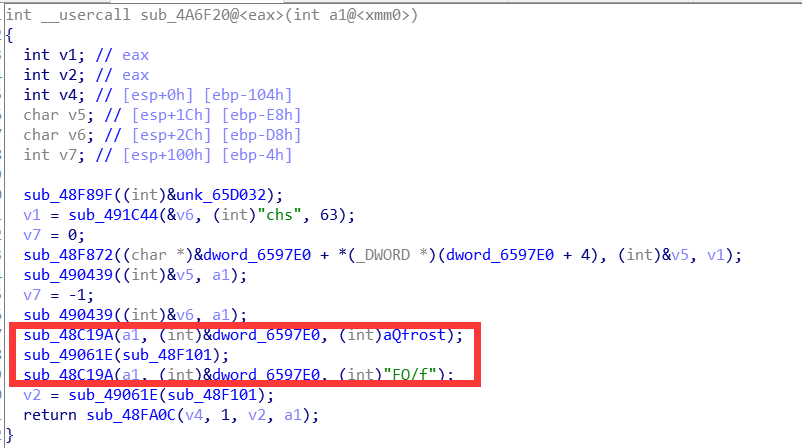
一个input，一个base64的表。双击input过去，x交叉引用，发现位置没变，往上滑，发现一段ida没有分析完全的代码



很明显，蓝框部分加了花指令，nop掉这三条指令，重新create函数，反编译，得到



一堆函数，按着点进去，在sub\_48DB4E函数里看到

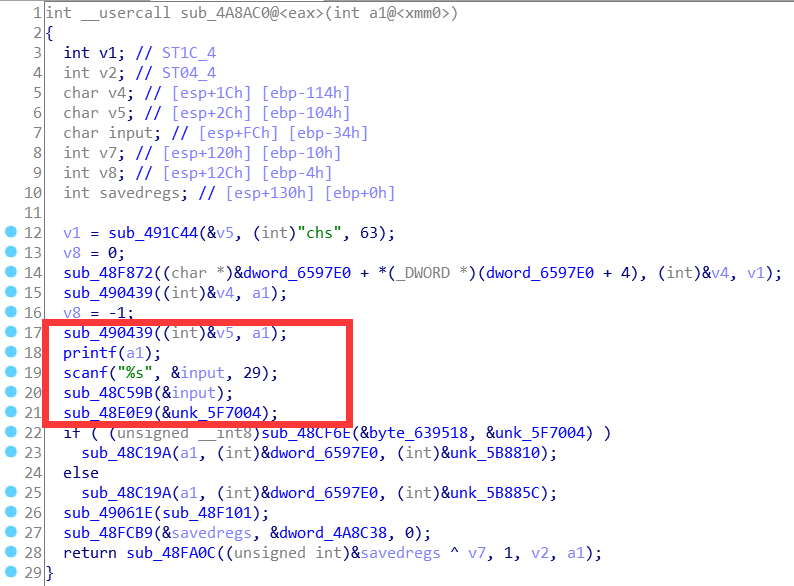


大概是输出程序运行时那些字符串

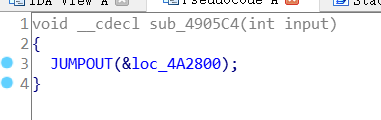


这里提示了应该要动调。不过还没有需要动调的充分理由，继续往下看。

回到sub\_4A8AC0函数，知道了sub\_48DB4E函数就是输出字符串的作用，那么它下面的函数，逻辑上来讲就是输入函数，把函数重命名，得到



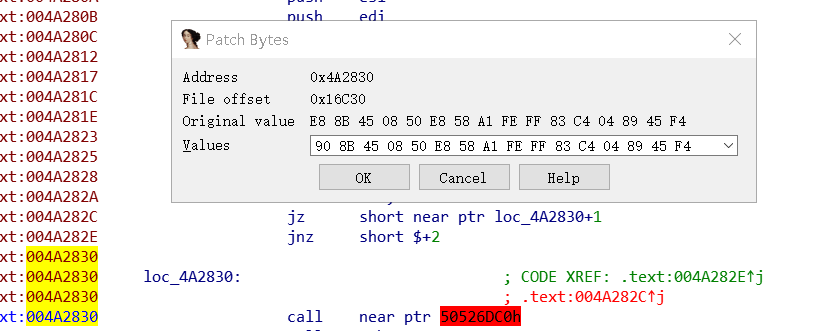
我们的输入要进入sub\_48C59B函数，按照input参数传递的层次一层一层点进去，最终来到



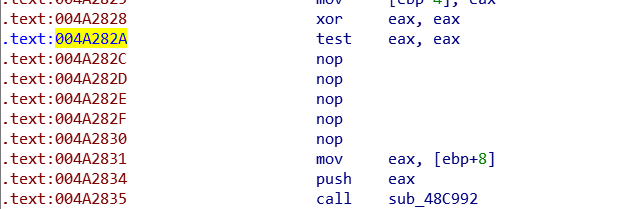
跟过去发现，也是一段ida没有分析完全的代码，加了花指令



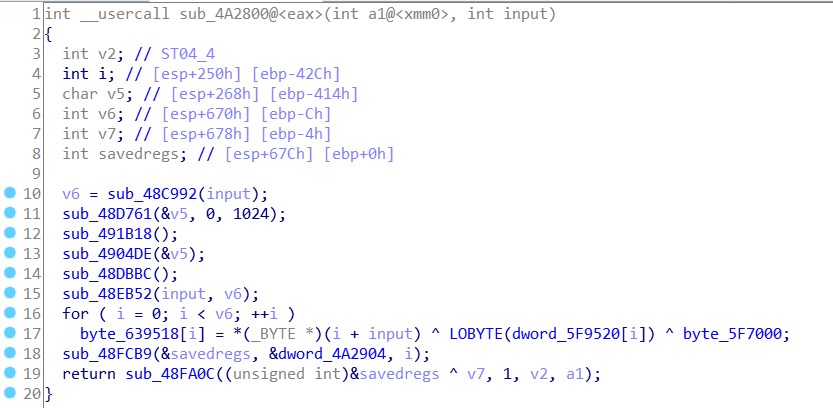
这是比较常见的加花方式，我一般是先把0x004A2830开始的第一个字节改成90，



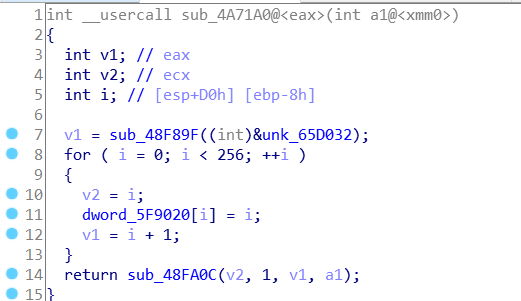
再把jz,jnz两条指令nop掉，得到



创建函数，反编译，得到

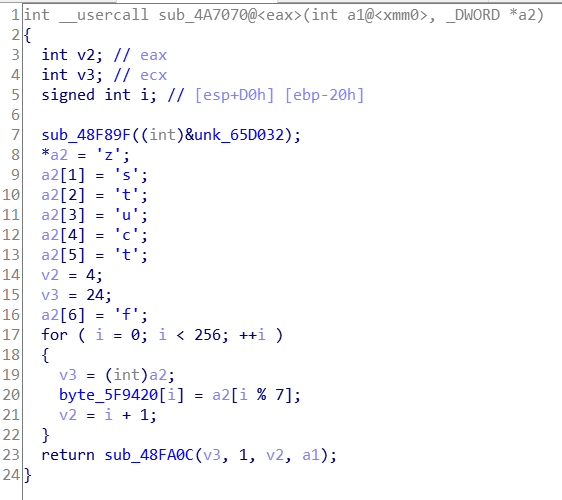


第一个函数大概是计算input的长度。其他挨着点进去，sub\_494B18最终到了

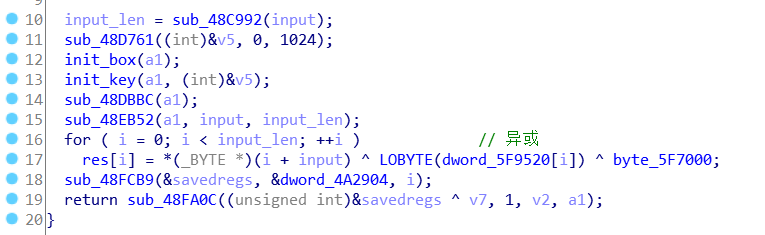


看起来是个盒子，和RC4加密很像，继续看其他函数。

Sub\_4904DE点进去，最后是



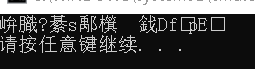
这里几乎可以肯定是RC4加密了，byte\_5F9420放的是密钥zstuctf



其他函数就是RC4的加密算法，我对这个加密算法掌握不是很熟练，这里能识别出来就可以了。现在关键是找密文。

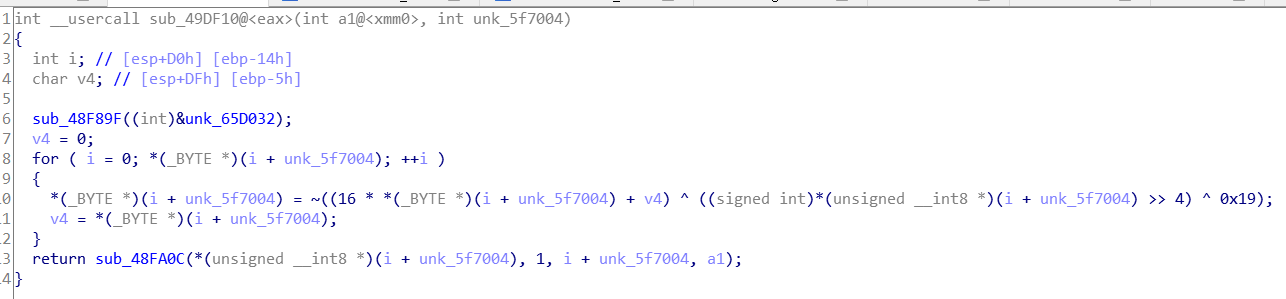
回到sub\_4A8AC0函数，sub\_48E0E9检测了OD调试器，反调试？这块没看懂。

继续往下，sub\_48CF6E是一个比较函数，比较的是我们的输入经RC4加密的结果和unk\_5F7004，后者可以直接扣下来，猜测是密文，直接上脚本

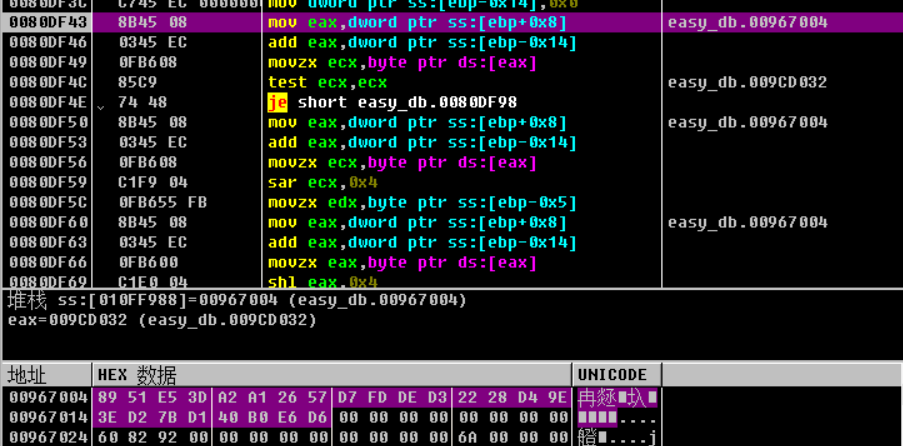


解出来没有意义。想到需要动调的提示，unk\_5F7004的值可能被修改了。

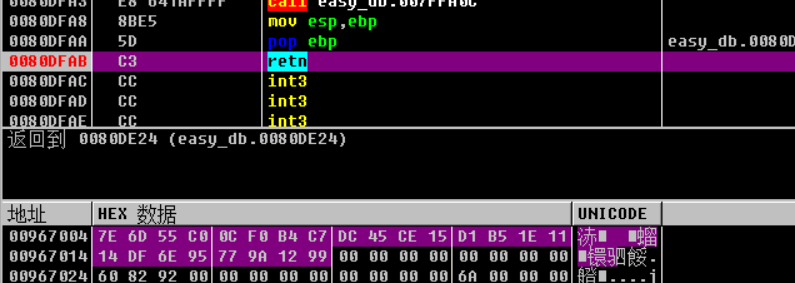
再仔细观察，程序给sub\_48E0E9函数传递了unk\_5F7004，按参数传递层次点进去，来到，确实修改了unk\_5F7004的值



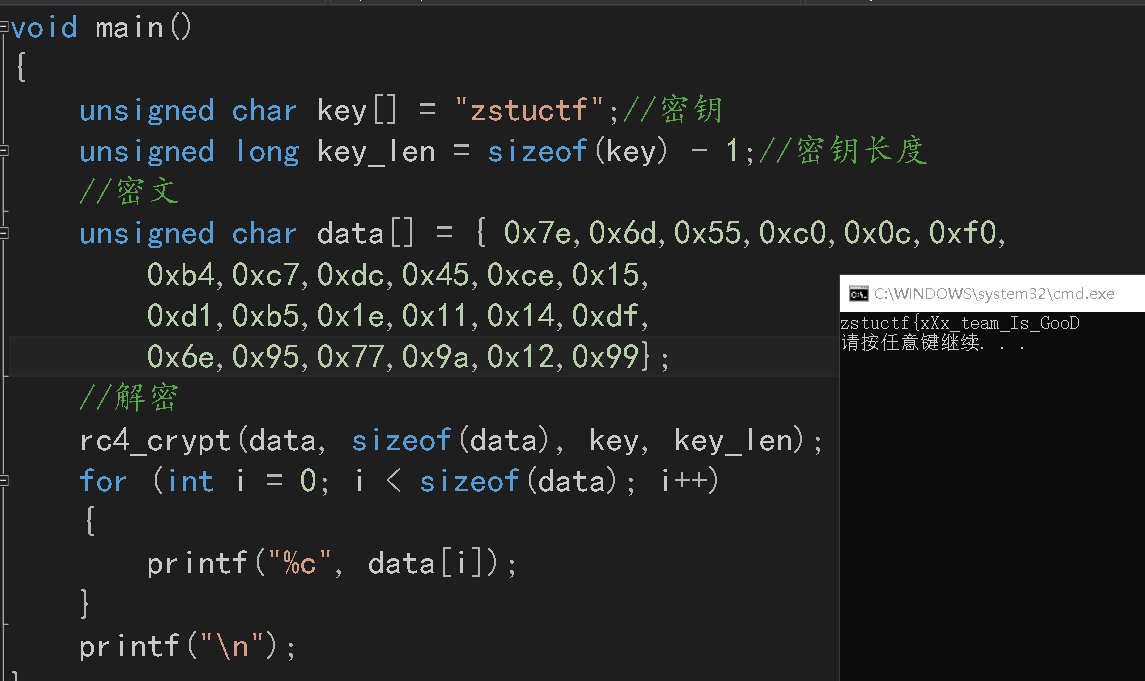
开始动调，由函数名逐渐来到



数据窗中的值是还未被修改过的值，执行到此函数的返回语句时，发现此数据段的值已经改变，也就是真正的密文



上脚本，得到



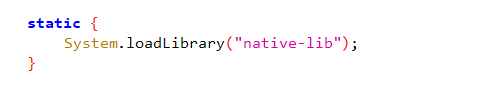
Flag是zstuctf{xXx\_team\_Is\_GooD}

RC4解密脚本有点长，不好放上来，百度也可以找到，有需要的话加我qq：326027901

Misc

输入密码

题目是一个apk，jadx一把梭打开找MainActivity，并没有发现主逻辑，看到

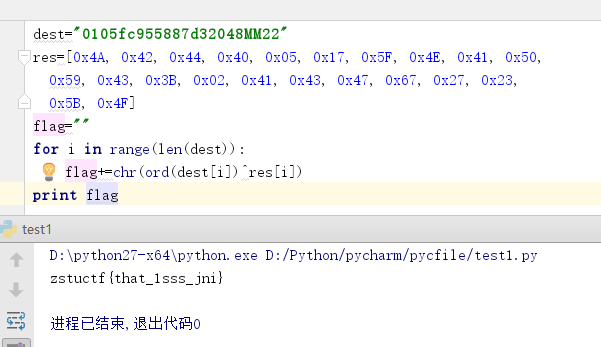




去native-lib里找stringFromeJNI函数。解压或反编译apk都可以，找到libnative-lib.so，32位ida打开，来到



逻辑清晰明白，首先检测输入的长度是否为22，一通操作让dest变成字符串0105fc955887d32048MM22，输入和dest异或，放在dest中，最后dest和unk\_15146比较，unk\_15146已知，写逆脚本。

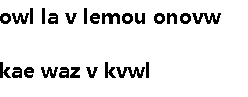


Crypto

替换密码

.txt文件，打开是没有意义的字符串，根据题目名可以得知是替换密码，先试试凯撒，发现不太行，应该不是简单的移位替换，猜测是按频率替换，notepad++可以计算每个字母出现的次数，再按照百度的得到的英文字母出现频率排序表可以一个一个猜，不过我并没有猜出来几个。

整个文档中m出现的次数最多，按照频率表猜测m替换的是e。看到这两行

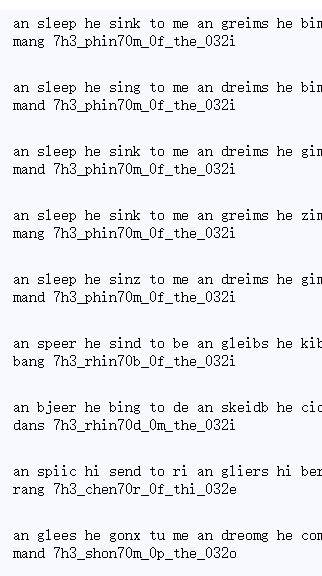


v单独成词，猜测是a，再和



搭配，猜测vw=an

利用在线网址<https://quipqiup.com/>，把整段和密文放上去，clues放vw=an，得到



网站解出来很多，这里只截取了一部分，7代表t，3代表e，0代表o，百度phin70m，也就是phintom,得到





phantom是有意义的单词，不过原来的i要变成a，于是7h3\_phin70m\_0f\_the\_032i就变成了7h3\_phan70m\_0f\_the\_032a，包上ZSTUCTF{}提交成功。