2021第二届“祥云杯”网络安全大赛 部分Writeup

原创

塞纳河畔的春水 于 2021-08-23 14:48:59 发布 2102 收藏 5
分类专栏： CTF_Writeup 文章标签： 网络安全

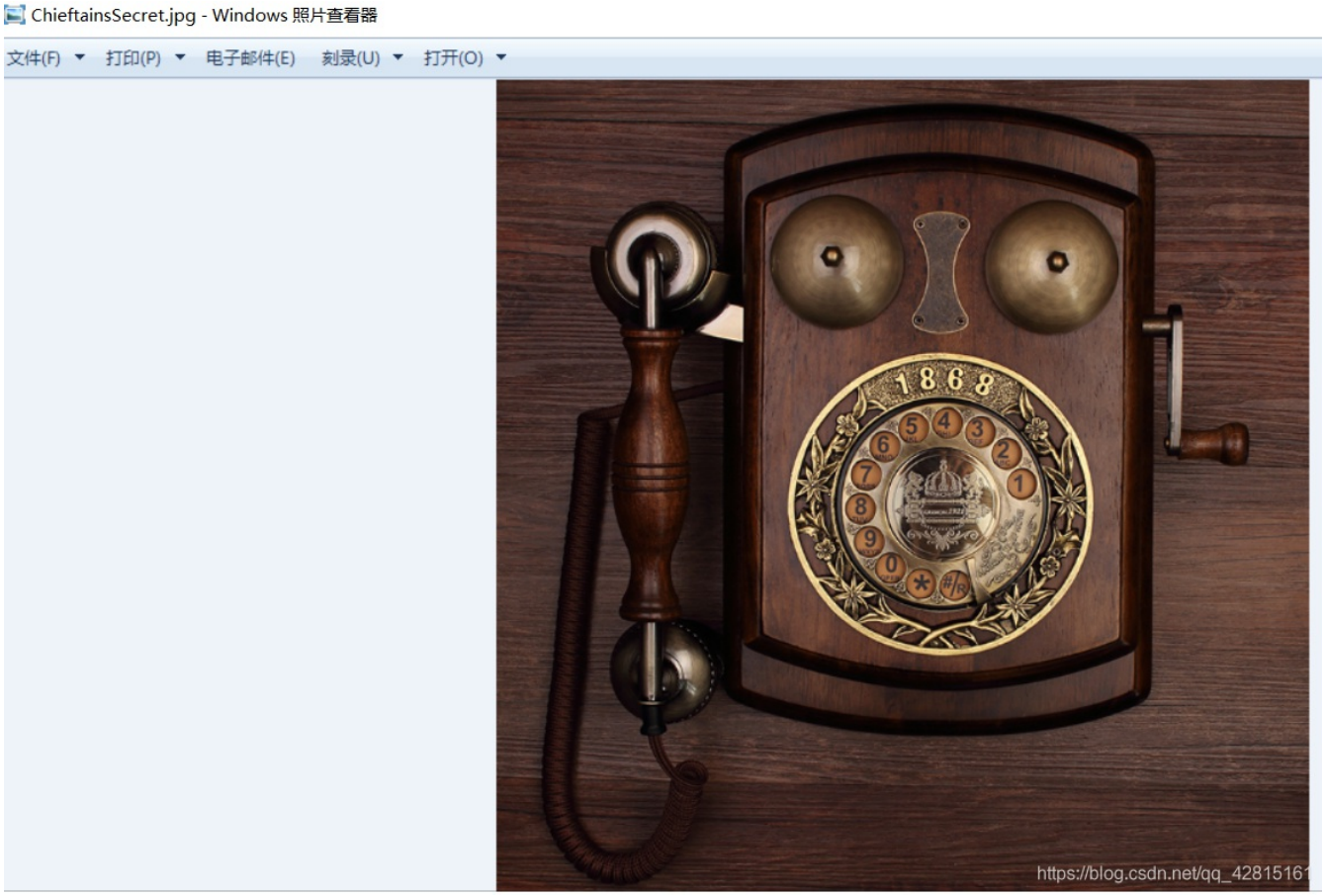
本文链接：https://blog.csdn.net/qq_42815161/article/details/119867158

文章目录

MISC

ChieftainsSecret

> 题目描述：Our agent risked his life to install a mysterious device in the immemorial telephone, can you find out the chieftain's telephone number? Flag format: flag{11 digits}

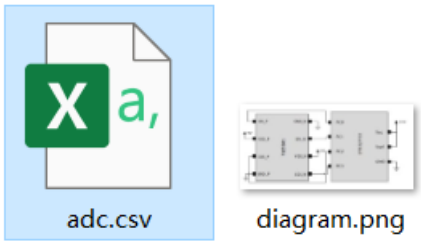


ChieftainsSecret.jpg

题目附件给了一张老式轮盘电话机图片，猜测图片藏了东西，直接上kali分离。

```
foremost ChieftainsSecret.jpg
```

得到一个RAR压缩包，直接解压，里面包含adc.csv和diagram.png两个文件。



adc.csv - Excel

文件 开始 插入 页面布局 公式 数据 审阅 视图 帮助 ACROBAT 团队 百度网盘 操作说明搜索

A1 PC0

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	PC0	PC1	PC2	PC3									
2	3105	3078	4085	2117									
3	3108	3076	4084	2118									
4	3111	3061	4083	2124									
5	3165	3082	4068	2121									
6	3155	3025	4073	2122									
7	3180	3014	4070	2124									
8	3214	3019	4077	2126									
9	3219	2966	4058	2138									
10	3249	2953	4065	2139									
11	3279	2899	4048	2146									
12	3312	2875	4042	2148									
13	3337	2845	4038	2162									
14	3366	2805	4028	2170									
15	3400	2778	4018	2175									
16	3421	2757	3986	2186									
17	3444	2739	3999	2190									
18	3466	2714	3992	2202									
19	3466	2663	3984	2203									
20	3501	2668	3975	2214									
21	3523	2666	3969	2233									
22	3540	2639	3959	2235									
23	3548	2620	3952	2232									
24	3552	2616	3947	2248									
25	3582	2600	3936	2250									
26	3605	2586	3914	2265									
27	3612	2570	3926	2279									

adc.csv

csv文件中包含了2162组PC0、PC1、PC2、PC3数据。

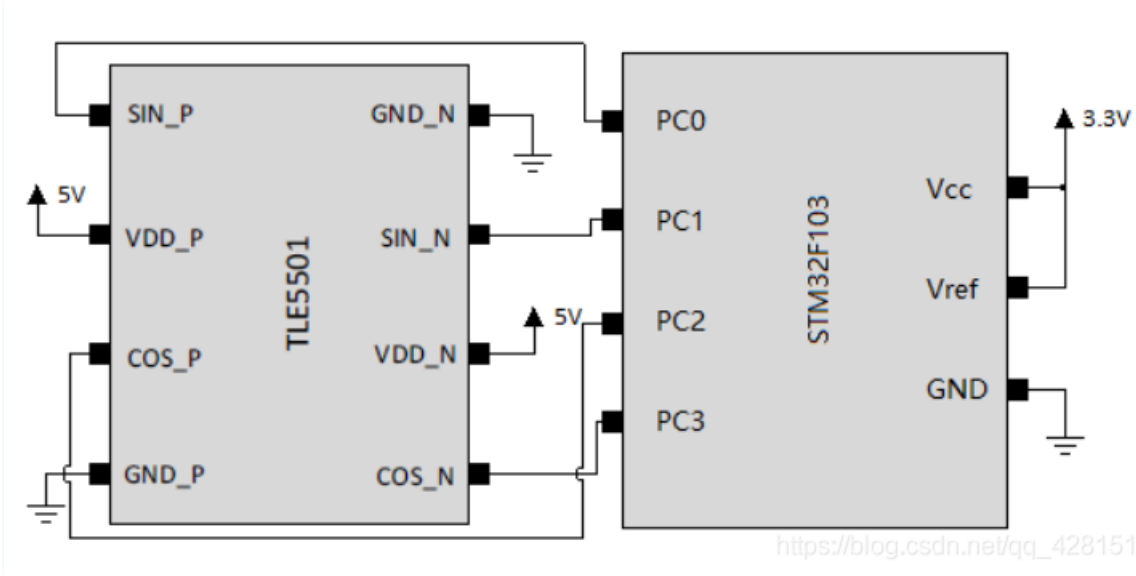
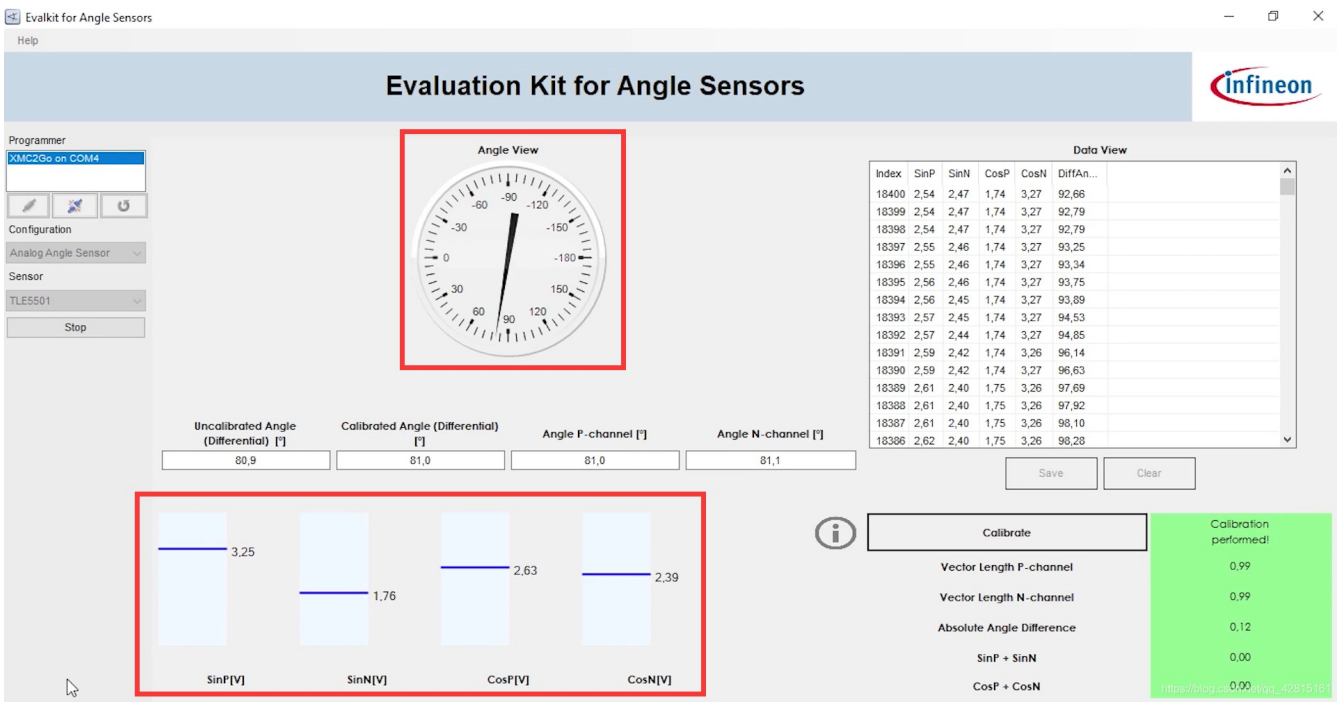


diagram.png

右边是STM32F103单片机，左边是TLE5501芯片。

查阅TLE5501资料：TLE5501 是一种基于 TMR 的 360° 角度传感器，可检测磁场的方向。这是通过测量正弦和正弦角度组件与隧道磁阻 （TMR） 元件实现的。这些原始信号（sin和cos）作为微分输出信号提供，可在微控制器内直接进一步处理。

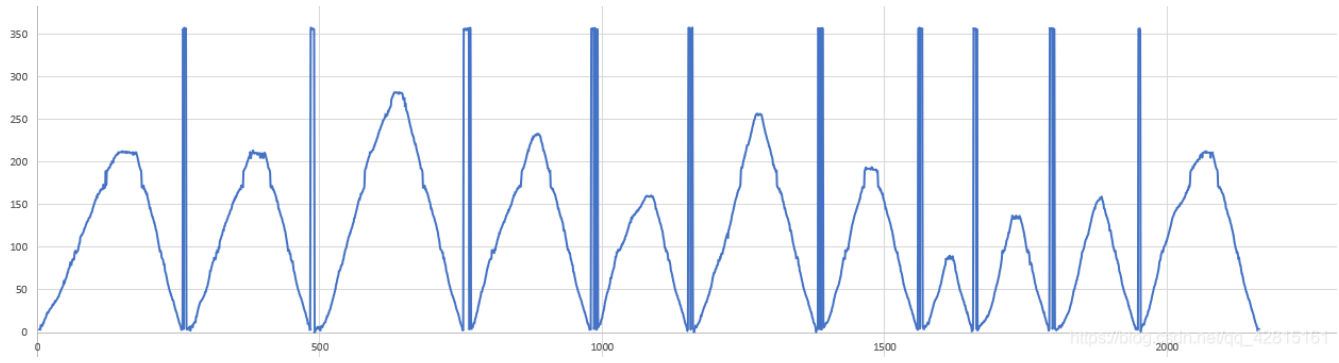


Infineon 公司提供的TLE5501实物测试软件

结合电路图分析，PC0-3各自对应SIN_P、SIN_N、COS_P、COS_N引脚输出，通过计算每一组数据可以得到一个角度。并且题意说明需要十一位电话号码，猜测2162组角度模拟了老式轮盘电话机的拨号过程。

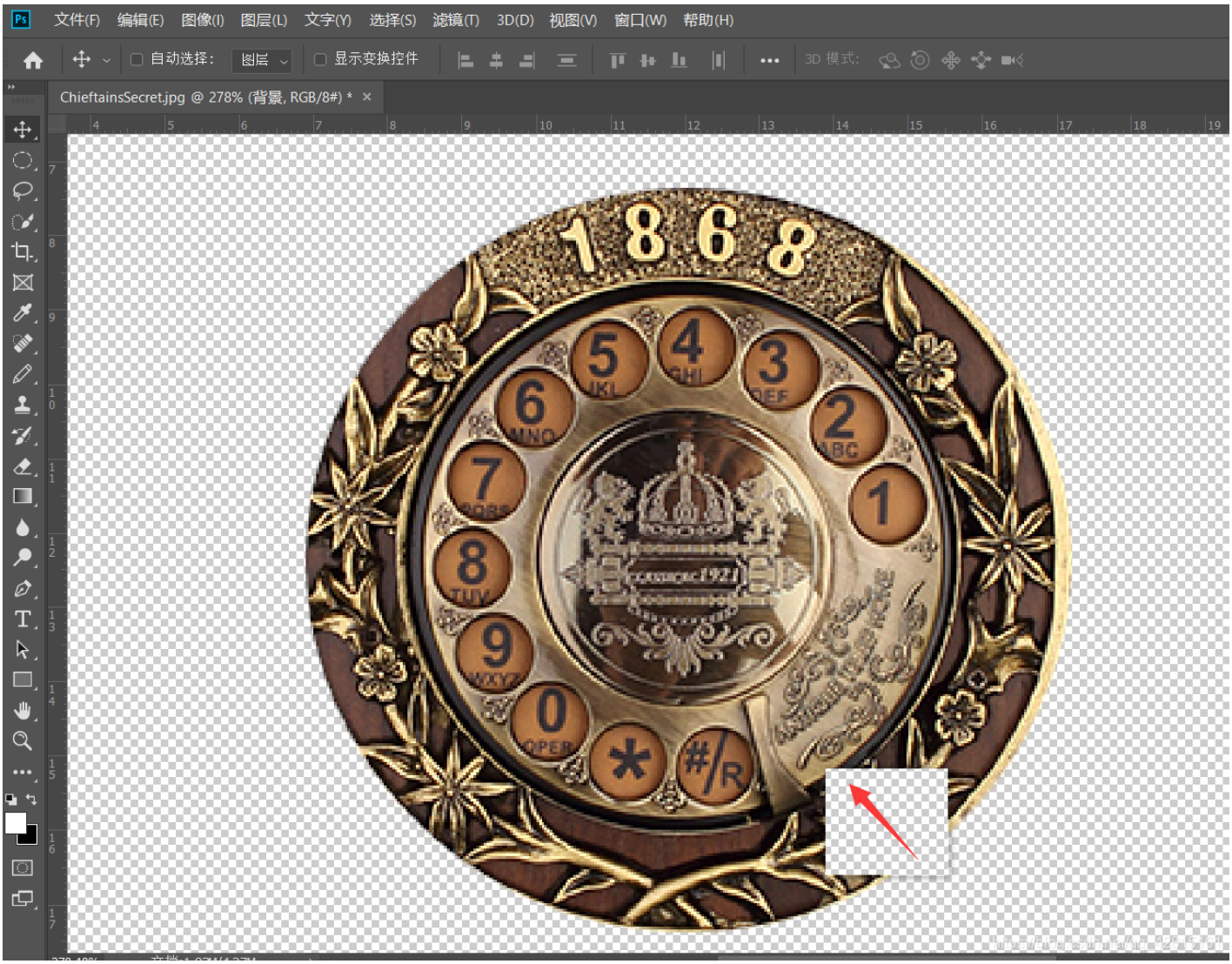
```
计算过程（可能比较笨）：
题目给出sin_N的范围为[2091,4098]，将其近似转化到[-1,1]方便求角度
令sin_N的范围为[sin_min,sin_max]
sin_ave = (sin_min+sin_max) / 2
sin_max_res=(sin_max-sin_ave)
对每个sin_N的值进行转化操作：sin_a = (sin_N-sin_ave) / sin_max_res
cos_N操作同理得到cos_a,进一步计算θ_a
确定θ数值：θ_t = arcsin(sin_a) * 180 / π
确定θ象限：θ_a = cos_a > 0 ? θ_t : θ_t+360
```

如图把2162组角度变化用图表绘制，可以观察到刚好有11个波峰。

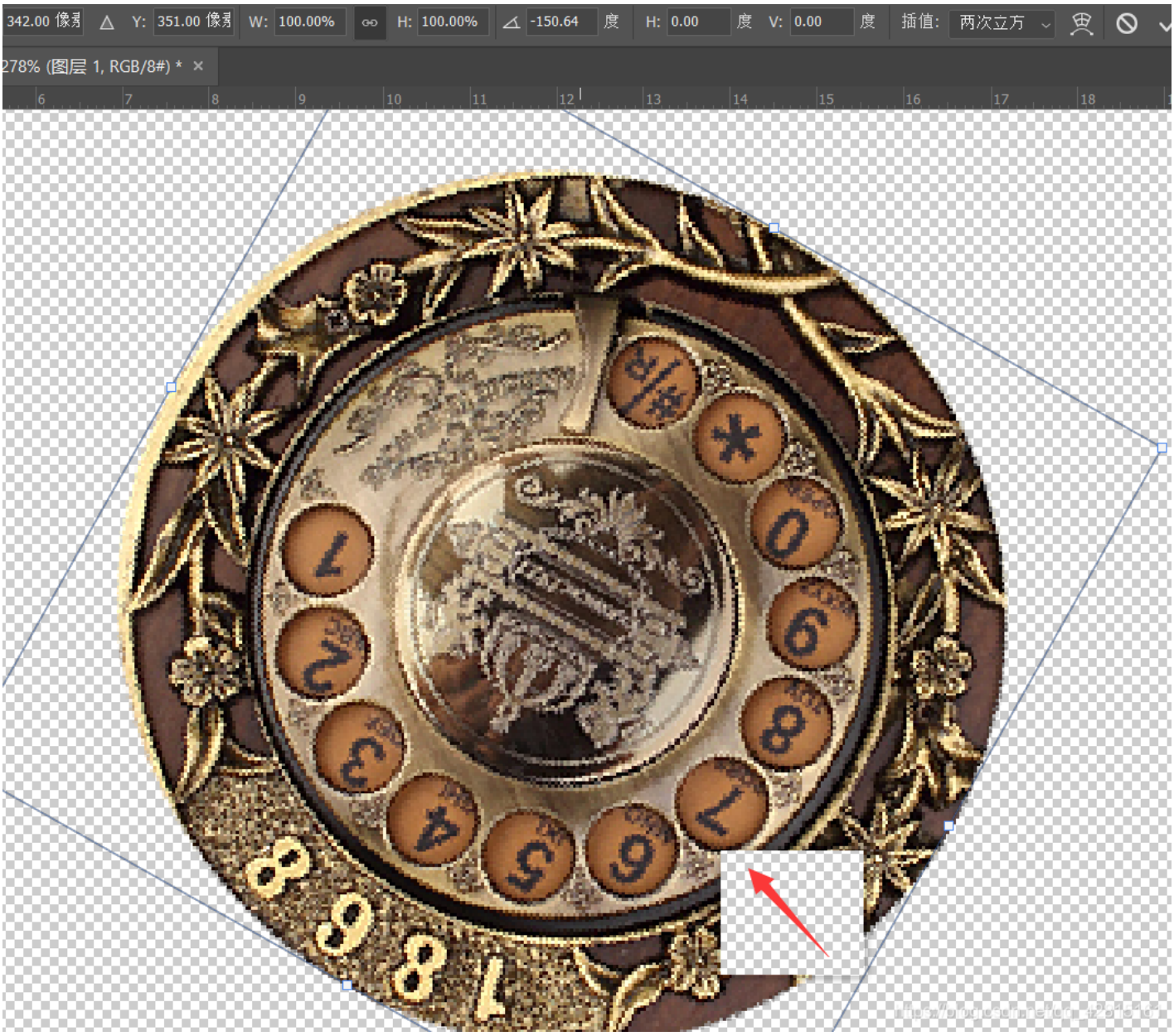


角度变化表

接下来计算角度对应老式电话上的数字，打开PS直接抠图模拟（当然直接算也是可以的/doge）。



假设波峰角度是210°，直接将轮盘转210°，观察到指向数字7，即第一个号码为7。



最终得到十一位数字：flag{77085962457}

鸣雏恋

> 题目描述：在喜欢的人面前,我可不能……丢脸啊……因为我,喜欢鸣人君……

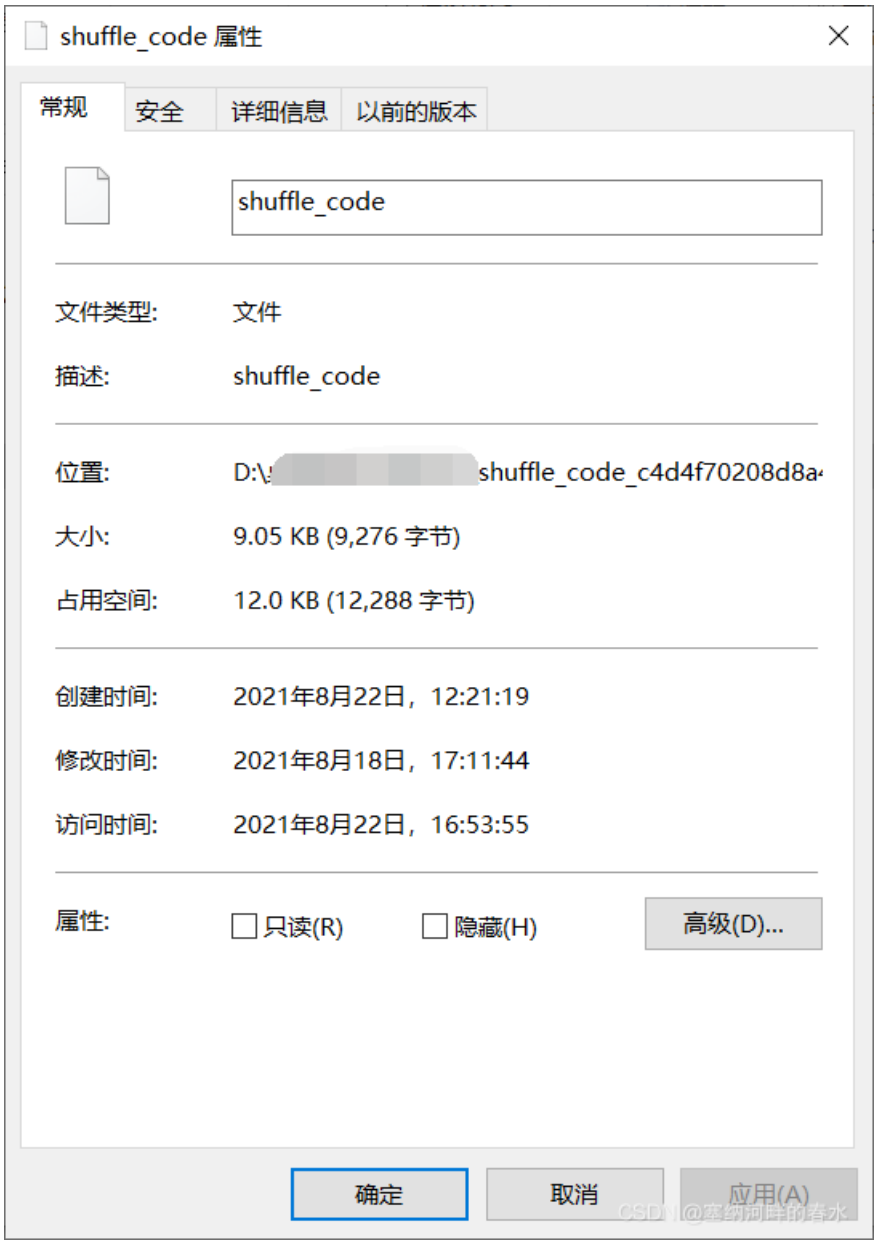
层层取证

> 题目描述：（本题附件较大）

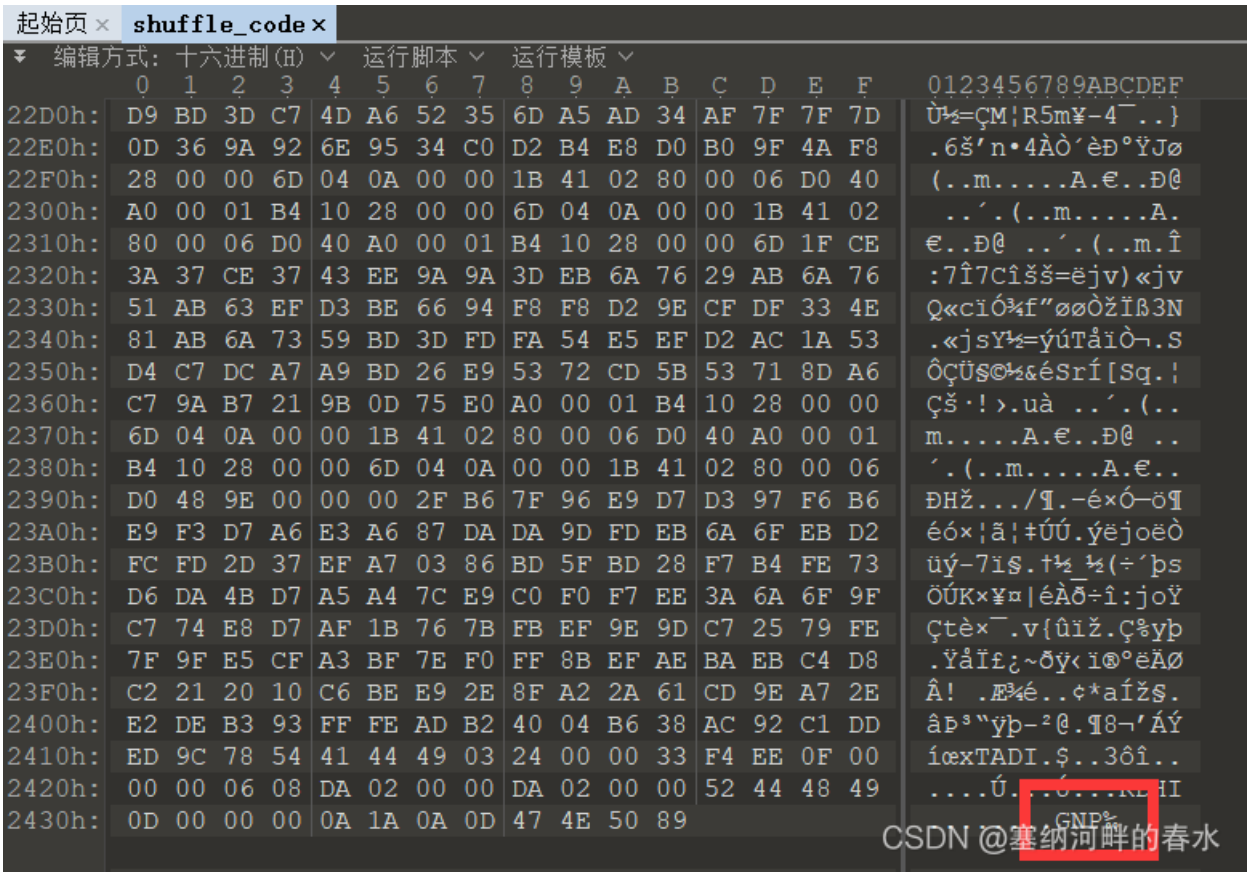
shuffle_code

无题目描述

题目给了一个文件



用010Editor打开查看文件。观察发现，文件末尾有PNG头，猜测是个反过来的PNG。



上Pycharm将文件倒置

```
input_file = "shuffle_code"
output_file = "output.png"
with open(input_file, 'rb') as f1, open(output_file, 'wb') as f2:
    f2.write(f1.read()[::-1])
```

生成一张二维码图片。

扫码后得到行和列两组各29个数据

```
col
426327/1132122/1211132223/3113253/61531113/111312/5323125/2222/11122153/311111/14312121/11231211/2423211/26

row
31121113/12321133/13111112/13112221121/12112232/16113232/11311311/21111231/11111211/711111117/2124112211/61
```

观察到特点521(11)11猜测11代表了11位数字，联想可能为数织，解密后的答案可能为29x29的二维码。打开excel输入数据，开始解密，题目有多解，考虑到二维码的容错率选取任意一解均可。

结合29x29二维码特征发现该点阵行序被打乱，呼应了题目shuffle_code。

上Pycharm爆破，大概一两分钟时间。参考大佬代码

```
data = [[1,1,1,1,1,1,1,0,1,0,1,1,0,0,1,1,0,1,0,0,1,0,1,1,1,1,1,1,1],
[1,0,0,0,0,0,0,1,0,1,0,1,0,0,0,1,0,0,0,0,0,1,1,0,1,0,0,0,0,0,0,1],
[1,0,1,1,1,0,1,0,1,0,1,1,1,1,1,1,0,0,0,0,1,1,0,1,0,1,1,1,0,1],
[1,0,1,1,1,0,1,0,0,1,0,0,1,1,1,0,0,1,1,0,1,0,1,0,1,1,1,0,1],
[1,0,1,1,1,0,1,0,1,0,0,0,0,1,0,0,0,0,0,1,1,0,0,1,0,1,1,1,0,1],
[1,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,1,0,0,1,0,1,0,0,0,0,0,1],
[1,1,1,1,1,1,1,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,1,1,1,1,1,1,1],
[0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[1,0,0,1,1,1,1,1,1,1,0,1,0,0,0,0,0,1,1,1,1,1,1,1,1,0,0,1,0,1,1,1],
[1,1,1,1,0,0,0,0,1,1,0,0,0,0,0,0,1,1,0,0,0,1,0,1,1,1,1,1,0,0,1],
[1,1,0,0,1,0,1,1,0,0,1,1,1,1,0,1,0,1,0,1,1,0,0,1,1,0,1,0,1,0,1],
[0,1,1,0,1,0,0,0,0,0,0,0,1,0,0,0,1,0,1,0,0,1,1,0,1,1,1,1,0,1],
[1,1,0,0,1,0,1,1,0,0,0,1,0,1,0,1,0,0,0,1,0,1,1,1,1,0,1,0,0,1],
[1,1,1,0,1,0,0,0,0,1,0,1,1,0,0,1,0,1,0,1,0,0,0,1,1,1,1,0,0,0],
[0,0,0,0,1,0,1,1,0,0,1,0,1,0,1,1,0,1,1,0,1,1,1,1,0,1,1,0,0,0],
[1,1,1,1,1,1,1,0,1,0,0,0,1,0,1,0,1,0,1,0,0,1,0,1,1,1,1,1,1,0,1],
[0,1,1,1,0,1,1,1,1,1,0,0,1,0,1,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,1],
[1,0,0,1,1,0,0,0,1,1,1,0,1,1,0,1,0,1,0,1,0,1,1,1,1,0,0,1,1,1,0,0],
[1,0,1,1,1,1,1,1,1,0,0,1,0,1,0,1,1,1,0,1,1,0,1,1,1,0,0,0,1,1],
[1,0,1,1,1,1,1,0,0,1,1,0,1,1,0,1,0,0,1,1,0,1,1,1,1,1,1,0,1,1],
[1,1,1,1,1,1,0,1,1,0,0,0,0,1,0,1,1,1,1,1,1,1,1,1,1,1,0,1,0,1],
[0,0,0,0,0,0,0,0,1,0,1,1,0,1,0,1,0,0,0,1,1,0,0,0,1,0,1,0,0],
[1,1,1,1,1,1,1,1,0,1,0,0,1,0,1,0,1,1,1,0,1,1,0,1,0,1,1,0,0,0],
[1,0,0,0,0,0,0,1,0,1,1,1,0,1,0,1,0,1,1,1,0,1,0,0,0,1,0,0,0,0],
[1,0,1,1,1,0,1,0,1,0,0,0,1,1,1,0,0,0,1,1,1,1,1,1,1,1,0,0,1,0],
[1,0,1,1,1,0,1,0,1,0,1,1,0,0,1,1,0,1,1,0,1,0,0,1,0,1,1,0,1],
[1,0,1,1,1,0,1,0,0,1,0,0,0,1,0,1,0,0,0,0,0,0,0,0,1,0,0,1,1],
[1,0,0,0,0,0,1,0,0,0,1,0,0,1,0,0,1,0,0,1,0,1,1,0,1,0,1,0,1],
[1,1,1,1,1,1,1,1,0,1,0,1,0,0,0,0,1,0,0,0,0,1,0,1,1,0,1,0,0,0]]

import pyzbar.pyzbar as pyzbar
from itertools import permutations
from PIL import Image, ImageDraw as draw
import matplotlib.pyplot as plt
from tqdm import tqdm
```

```
shuffle_1 = [9, 11, 13, 15, 17, 19]
shuffle_2 = [10, 12, 14, 16, 18]

head = data[:9]
tail = data[20:]

def body(body_1, body_2): # 获取中间部分的一种排列
    body = []
    for i in range(5):
        body.append(body_1[i])
        body.append(body_2[i])
    body.append(body_1[5])
    return [data[i] for i in body]

def draw_img(data): # 生成二维码图片
    assert len(data) == 29 and len(data[0]) == 29
    img = Image.new('RGB', (31, 31), (255,255,255))
    for i, row in enumerate(data):
        for j, pixel in enumerate(row):
            img.putpixel((j + 1, i + 1), (0,0,0) if pixel == 1 else (255,255,255))
    return img

with tqdm(total=720 * 120) as pbar:
    for body_1 in permutations(shuffle_1):
        for body_2 in permutations(shuffle_2):
            im = draw_img(head + body(body_1, body_2) + tail)
            barcodes = pyzbar.decode(im)
            pbar.update(1)
            if(len(barcodes) == 0):
                continue

            for barcode in barcodes:
                barcodeData = barcode.data.decode("utf-8")
                print(barcodeData)
                plt.imshow(im)
                plt.show()
```

爆破结果：

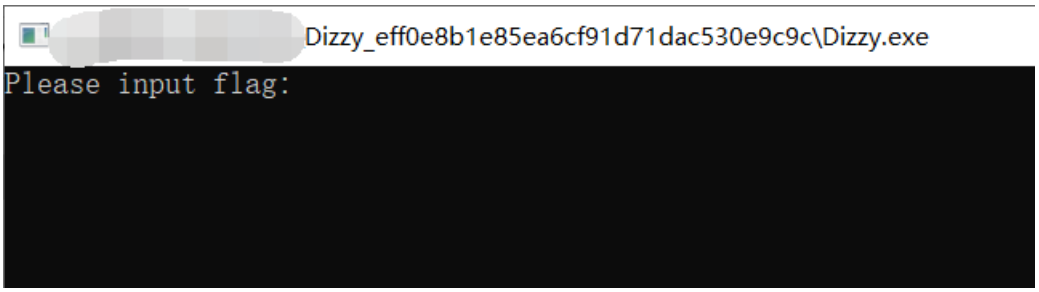
> flag{f31861a9-a753-47d5-8660-a8cada6c599e}

Reverse

Rev_Dizzy

> 题目描述：逆向分析要依赖工具，更要相信自己

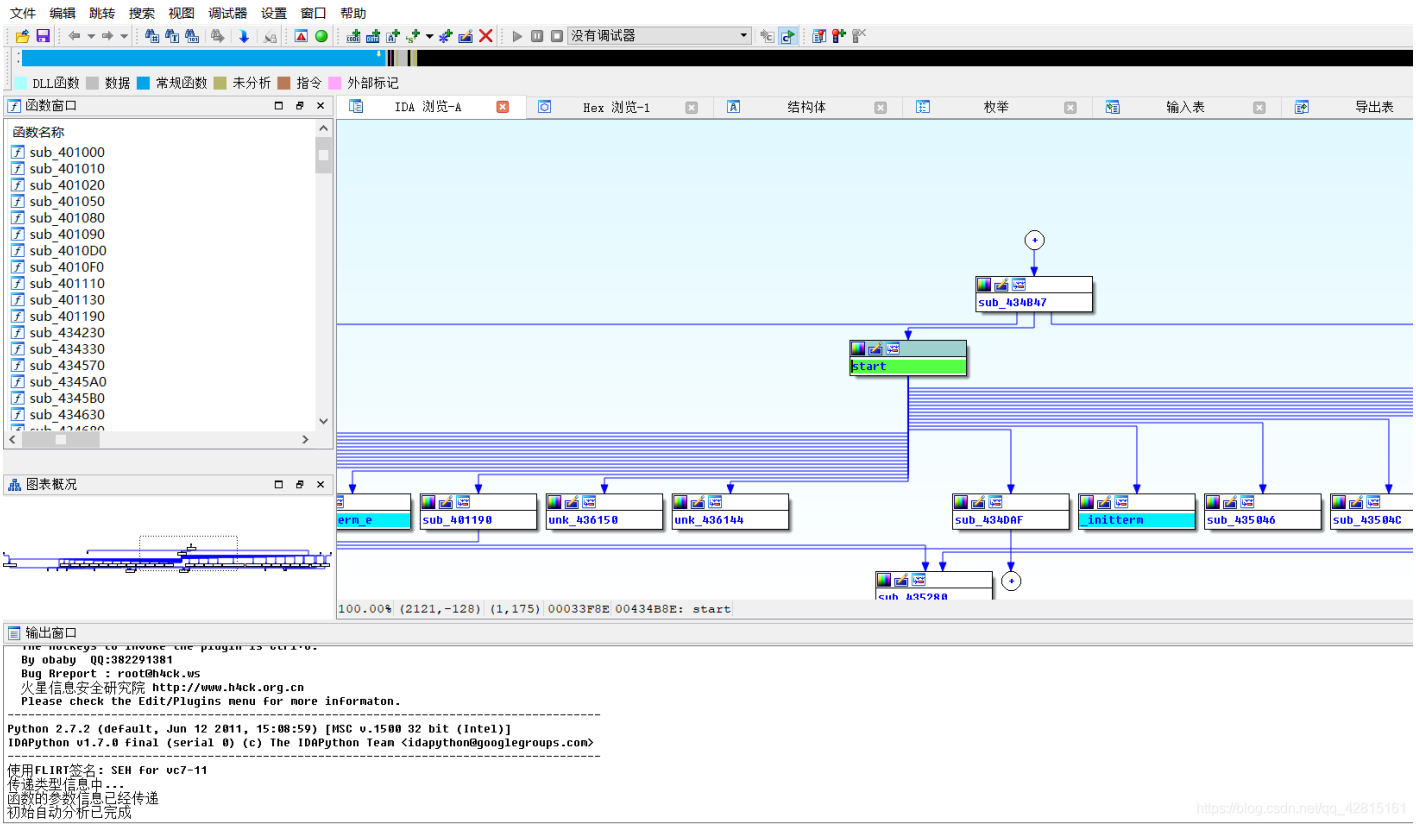
题目只给一个exe附件，Window下直接运行，随意输入后命令窗口直接关闭。



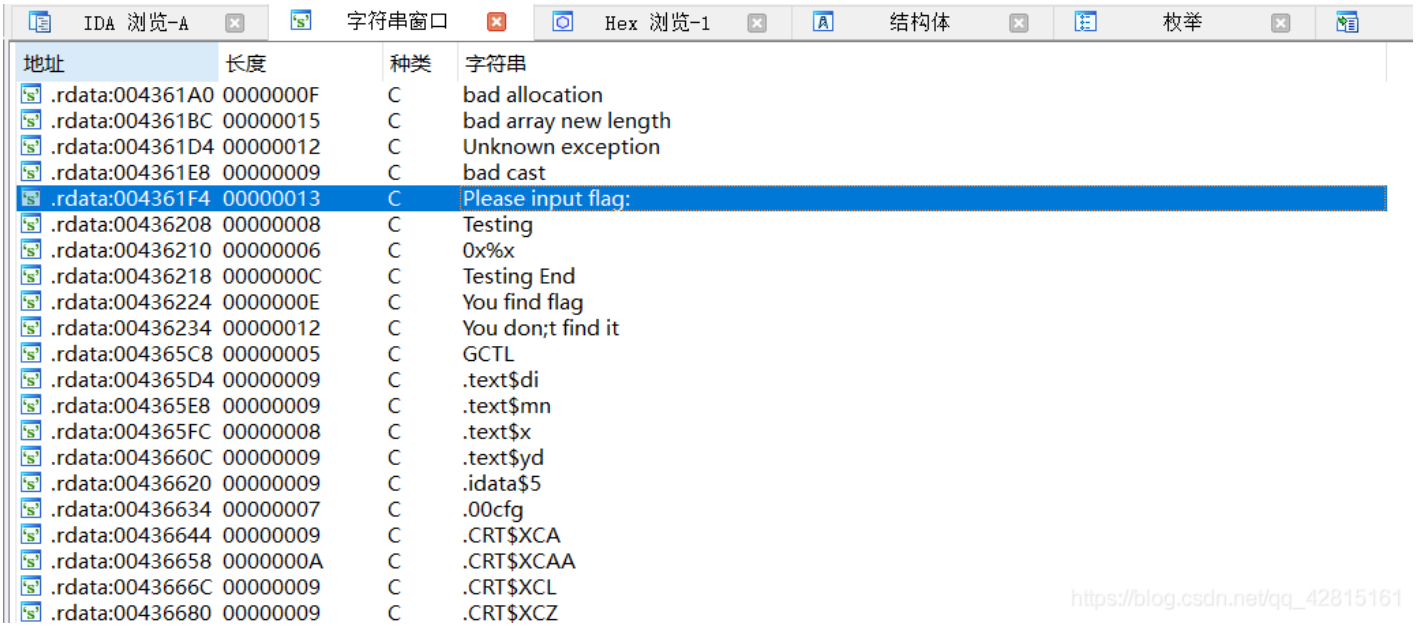
上kali，分析文件格式。

```
file Dizzy.exe
Dizzy.exe: PE32 executable (console) Intel 80386, for MS Windows
```

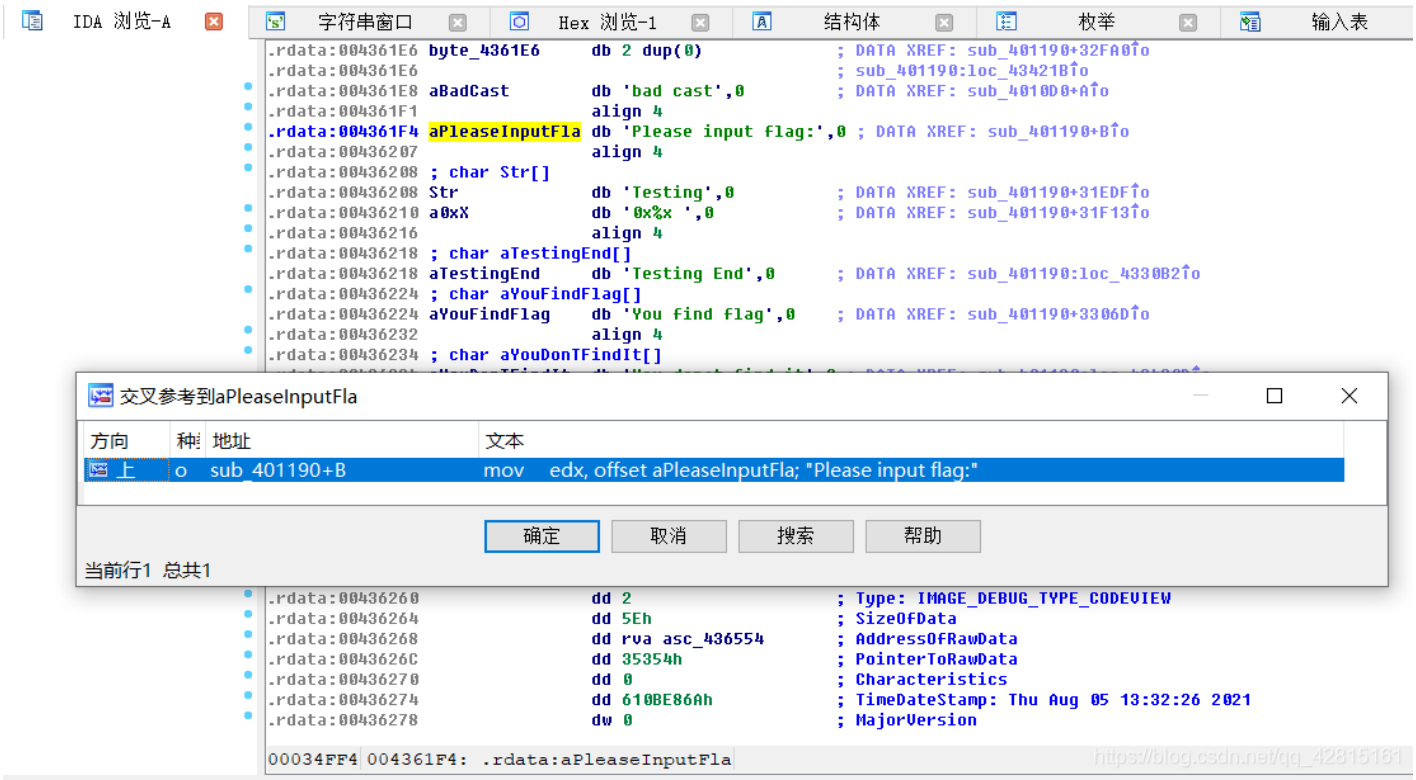
32位，用ida打开



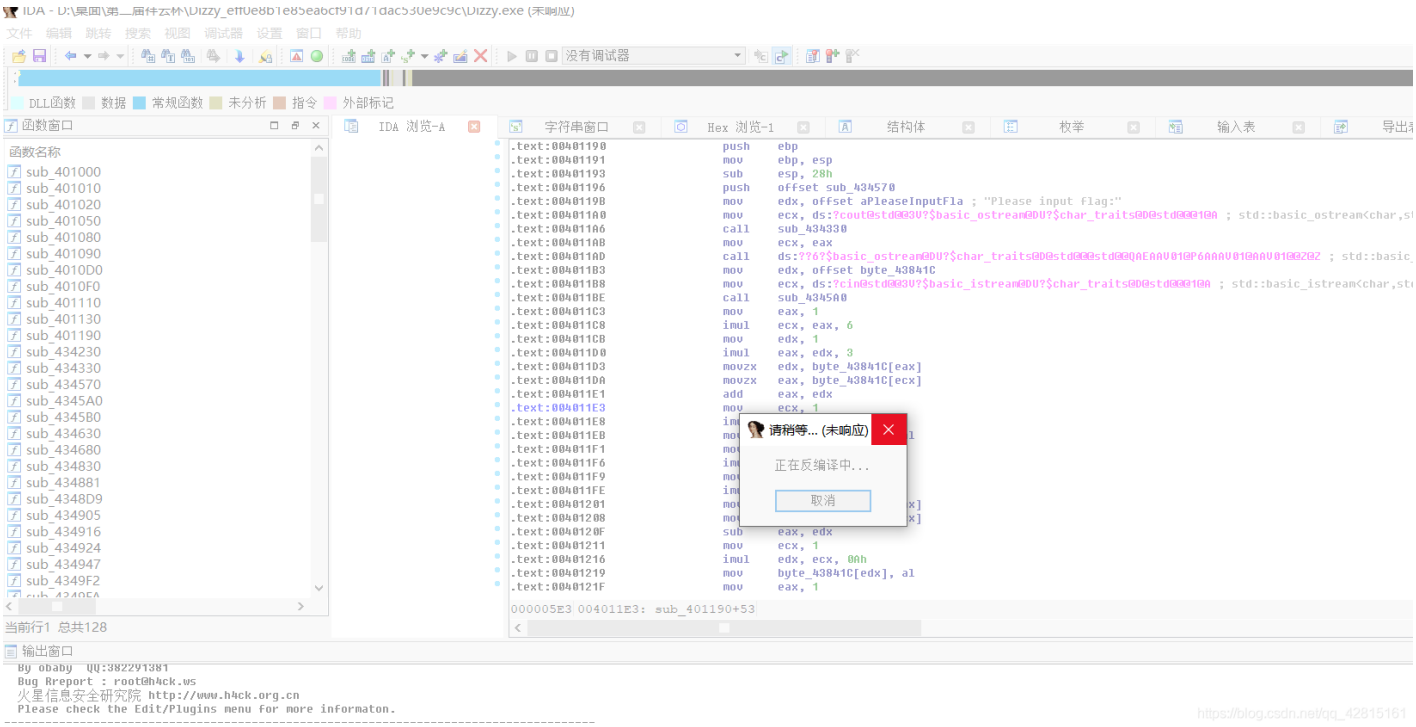
Shift+F12字符串搜索，找到刚才看到的字符串“Please input flag:”



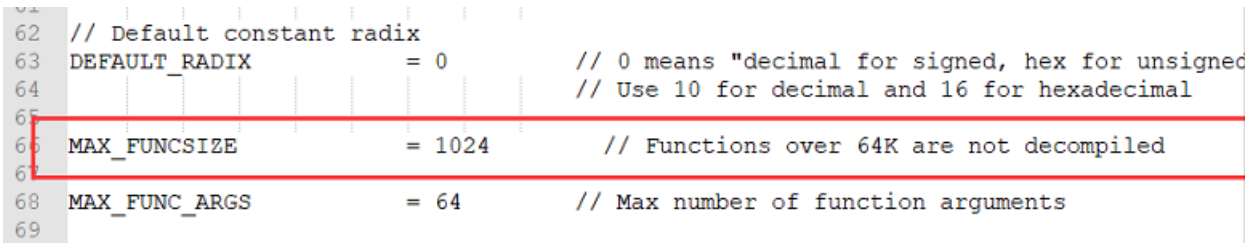
Ctrl+X交叉引用定位到函数位置。



汇编看不明白，F5反编译。



反编译卡死，代码量很大，正常现象。若提示反编译失败，函数太大则修改IDA安装目录\cfg\hexrays.cfg文件，以文本方式打开，修改MAX_FUNCSIZE 的值（可修改为1024）。



反编译成C++结果：（太长了不放代码了Orz）

```
int sub_401190()
{
  int v0; // eax@1
  char v2; // [sp+0h] [bp-28h]@4
  char v3; // [sp+1h] [bp-27h]@4
  char v4; // [sp+2h] [bp-26h]@4
  char v5; // [sp+3h] [bp-25h]@4
  char v6; // [sp+4h] [bp-24h]@4
  char v7; // [sp+5h] [bp-23h]@4
  char v8; // [sp+6h] [bp-22h]@4
  char v9; // [sp+7h] [bp-21h]@4
  char v10; // [sp+8h] [bp-20h]@4
  char v11; // [sp+9h] [bp-1Fh]@4
  char v12; // [sp+Ah] [bp-1Eh]@4
  char v13; // [sp+Bh] [bp-1Dh]@4
  char v14; // [sp+Ch] [bp-1Ch]@4
  char v15; // [sp+Dh] [bp-1Bh]@4
  char v16; // [sp+Eh] [bp-1Ah]@4
  char v17; // [sp+Fh] [bp-19h]@4
  char v18; // [sp+10h] [bp-18h]@4
  char v19; // [sp+11h] [bp-17h]@4
  char v20; // [sp+12h] [bp-16h]@4
  char v21; // [sp+13h] [bp-15h]@4
  char v22; // [sp+14h] [bp-14h]@4
  char v23; // [sp+15h] [bp-13h]@4
  char v24; // [sp+16h] [bp-12h]@4
  char v25; // [sp+17h] [bp-11h]@4
  char v26; // [sp+18h] [bp-10h]@4
  char v27; // [sp+19h] [bp-Fh]@4
  char v28; // [sp+1Ah] [bp-Eh]@4
  char v29; // [sp+1Bh] [bp-Dh]@4
  char v30; // [sp+1Ch] [bp-Ch]@4
  char v31; // [sp+1Dh] [bp-Bh]@4
  char v32; // [sp+1Eh] [bp-Ah]@4
  char v33; // [sp+1Fh] [bp-9h]@4
  int i; // [sp+20h] [bp-8h]@1
  int j; // [sp+24h] [bp-4h]@4

  v0 = sub_434330(std::cout, "Please input flag:");
  std::basic_ostream<char,std::char_traits<char>>::operator<<(v0, sub_434570);
  sub_4345A0(std::cin, byte_43841C);
  byte_43841C[6] += byte_43841C[3];
  byte_43841C[10] -= byte_43841C[27];
  byte_43841C[10] += byte_43841C[6];
  byte_43841C[14] += byte_43841C[3];
  byte_43841C[28] ^= byte_43841C[24];
                .
                .
                .
        此处省略上千次操作
                .
                .
                .
  byte_43841C[21] -= 54;
  byte_43841C[0] += byte_43841C[5];
  byte_43841C[16] += byte_43841C[20];
  puts(byte_4361E6);
  v2 = 39;
  v3 = 60;
```

```
  v4 = -29;
  v5 = -4;
  v6 = 46;
  v7 = 65;
  v8 = 7;
  v9 = 94;
  v10 = 98;
  v11 = -49;
  v12 = -24;
  v13 = -14;
  v14 = -110;
  v15 = -128;
  v16 = -30;
  v17 = 54;
  v18 = -76;
  v19 = -78;
  v20 = 103;
  v21 = 119;
  v22 = 15;
  v23 = -10;
  v24 = 13;
  v25 = -74;
  v26 = -19;
  v27 = 28;
  v28 = 101;
  v29 = -118;
  v30 = 7;
  v31 = 83;
  v32 = -90;
  v33 = 102;
  for ( j = 0; j < 32 && (unsigned __int8)*(&v2 + j) == (unsigned __int8)byte_43841C[j]; ++j )
    ;
  if ( j == 32 )
    puts("You find flag");
  else
    puts("You don;t find it");
  puts(byte_4361E6);
  return 0;
}
```

分析：输入一个长度为32的字符串，字符串内部元素互相作加 '+' 、减 '-' 、异或 '^' 操作共5000次，最后输出的字符串与设定值进行比对。若均相同，则输入字符串为所需flag。可进行相应逆运算解题目（加对应逆运算为减）。

上PyCharm将5000个公式进行相应逆处理：

```
input_file_path = "D:/Dizzy_eff0e8b1e85ea6cf91d71dac530e9c9c/1.txt"
ans = ""
with open(input_file_path, 'r') as f:
    for line in f.readlines():
        if '+' in line:
            ans = line.replace('+', '-') + ans
        else:
            ans = line.replace('-', '+') + ans
    print(ans)
```

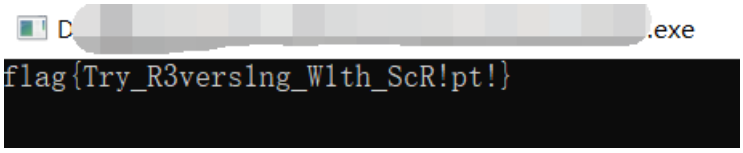
再把代码放进Cpp文件在Visual Studio里面跑一遍。

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
 char flag[32] = {39,60,-29,-4,46,65,7,94,98,-49,-24,-14,-110,-128,-30,54,-76,-78,103,119,15,-10,13,-74,-19
 flag[16] -= flag[20];
 flag[0] -= flag[5];
 flag[21] += 54;
             .
             .
             .
     此处省略上千次操作
             .
             .
             .
 flag[28] ^= flag[24];
 flag[14] -= flag[3];
 flag[10] -= flag[6];
 flag[10] += flag[27];
 flag[6] -= flag[3];

 for (int i = 0; i < 32; i++) {
  cout << flag[i];
 }
 int a;
 cin >> a;
 return 0;
}
```

很快啊，一会儿就跑出来了。



> flag{Try_R3vers1ng_W1th_ScR!pt!}