CTF--2016XDCTF全国网络安全大赛之reverse2

原创

0x01 题目介绍

题目名称

reverse2

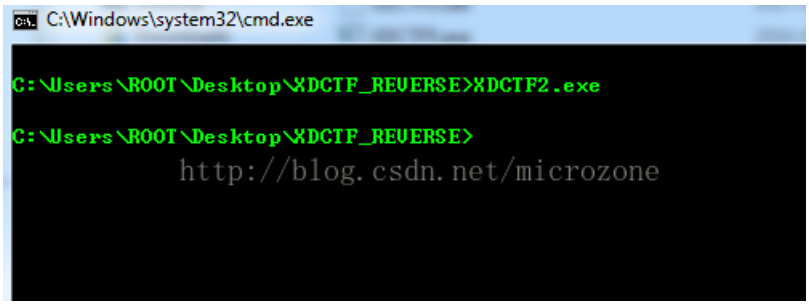
题目描述

File: XDCTF2.exe
Size: 195584 bytes
File Version: 1.0.0.1
Modified: Tuesday, March 07, 2017, 00:11:41
MD5: 1B0EA0BD4B8DA116C7D590F05BE6692A
SHA1: 81174B9D39D467308818B0BB0C99D427ED80EB6C
CRC32: 4D44CF29

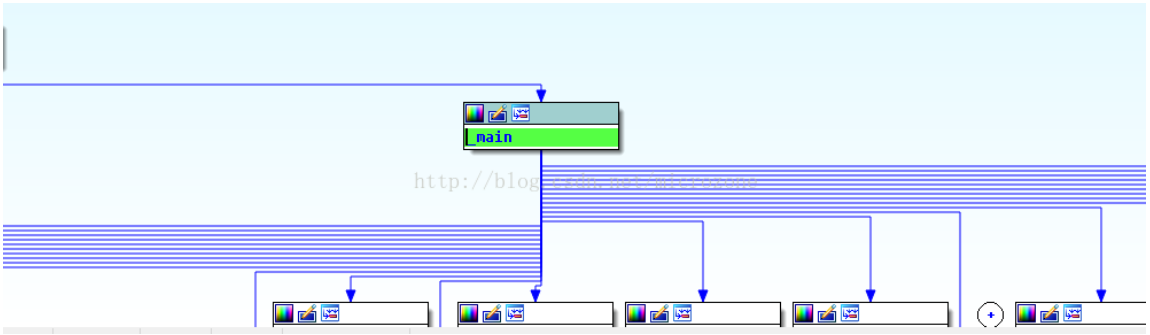
0x02 解题要点

先点击程序运行测试，发现一闪而过。回到cmd下面进行运行。命令行运行程序，什么也没有输出。

于是觉得奇怪，一般的破解逆向程序都有输入输出提示，或者界面等消息。



于是还是先暂时用IDA载入测试一下。找到代码部分，大理一阳指F5



进入main函数。注意红色标记部分，函数入口。

```
21  sub_410HVH();
22  if ( _ioinit() < 0 )
23    _amsg_exit(27);
24  dword_43017C = (int)GetCommandLineA();
25  dword_42F388 = (char *)__crtGetEnvironmentStringsA();
26  if ( _setargv() < 0 )
27    _amsg_exit(8);
28  if ( _setenvp() < 0 )
29    _amsg_exit(9);
30  v0 = _cinit(1);
31  if ( v0 )
32    _amsg_exit(v0);
33  dword_42F700 = (int)envp;
34  v1 = main(argc, (const char **)argv, (const char **)envp);
35  v3 = v1;
36  if ( !v4 )
37    exit(v1);
38  _cexit();
39  return v3;
40}
```


又出现了多线程，注意此处我们用的是静态分析工具，代码没有执行，

不会出现逻辑分支干扰，运行错误。如果是动态分析可能已经跟踪偏了。

```
265         }
266         dword_42F22C = (int)CreateThread(0, 0, StartAddress, 0, 0, 0);
267         if ( argc == 3 )
268         {
269           strcpy(&v39, argv[1]);
270           v28 = (char)dword_42F22C ^ 0x11;
271           for ( k = 0; k < sub_405000(&v122); ++k )
272           {
273             if ( *(&v39 + k) != *(&v122 + k) )
274             {
275               v26 = 0;
276               v25 = 0;
277               while ( v26 < sub_405000(&unk_41E118) )
278               {
279                 v26 += v25;
280                 v25 += 3;
281                 v26 = v25 ^ v25 & v26;
282               }
```

花指令

注意此段参数数量为1，sub_405000经过分析是计算字符串长度。

类似于我们常见的函数strlen

```
155   if ( argc == 1 )
156   {
157     v36 = 0;
158     v35 = 0;
159     while ( v36 < sub_405000(&unk_41E118) )
160     {
161       v36 += v35;
162       v35 += 2;
163       v36 |= v35++;
164       v36 &= v35;
165     }
166     result = 0;
167   }
168   else
169   {
```


这是字符串计算长度函数实体部分，里面用结束\0作为判断标志位。

返回值是result

```
1 int __cdecl sub_405000(int a1)
2 {
3   int result; // eax@2
4   int v2; // [sp+0h] [bp-8h]@1
5   int v3; // [sp+4h] [bp-4h]@1
6
7   v3 = a1;
8   v2 = 0;
9   if ( a1 )
10  {
11    if ( *(_BYTE *)a1 )
12    {
13      while ( *(_BYTE *)v3 )
14      {
15        ++v2;
16        ++v3;
17      }
18      result = v2;
19    }
20    else
21    {
22      result = 0;
23    }
24  }
```

输入参数1运算，输入参数2运算，0x38,0x66文件参数运算

Name运算完成之后会得到一个TEMP字符串

lpBuffer接收返回的路径

```
251   for ( i = 0; i < 10; ++i )
252   {
253     *(&v122 + i) ^= 0x44u;
254     ++*(&v122 + i);
255   }
256   v132 = 0;
```

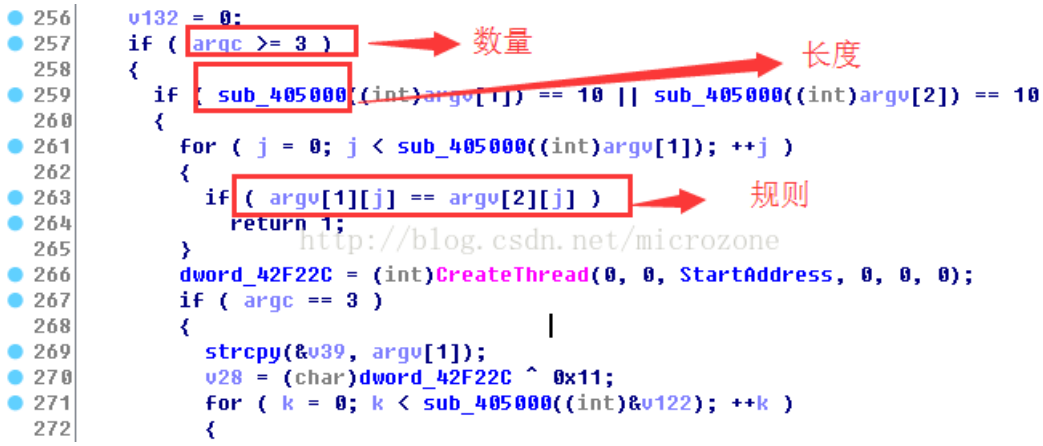
```
288   for ( l = 0; l < 0xA; ++l )
289   {
290     *(&v94 + l) ^= 0x55u;
291     --*(&v94 + l);
292     ++*(&v94 + l);
293   }
294   v104 = 0;
```

byte_42DECC进行异或运算之后得到读写文件标记。

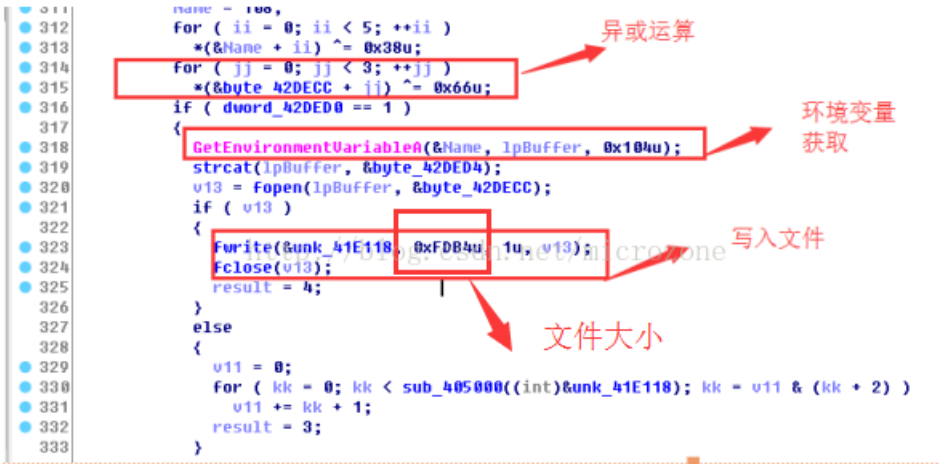
```
312   for ( ii = 0; ii < 5; ++ii )
313     *(&Name + ii) ^= 0x38u;
314   for ( jj = 0; jj < 3; ++jj )
315     *(&byte_42DECC + jj) ^= 0x66u;
316   if ( dword_42DED0 == 1 )
317   {
318     GetEnvironmentVariableA(&Name, lpBuffer, 0x104u);
319     strcat(lpBuffer, &byte_42DED4);
320     v13 = fopen(lpBuffer, &byte_42DECC);
321     if ( v13 )
322     {
323       fwrite(&unk_41E118, 64948u, 1u, v13);
324       fclose(v13);
325       result = 4;
326     }
```

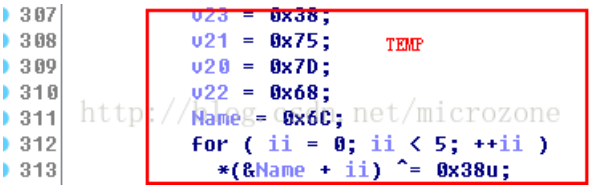
注意不同的逻辑判断，参数数量1,2,3。发现在3的时候有参数数量限制，长度和比较规则。

由此判断输入必须两个参数，长度为10，而且第一个解密之后和第二个相等。



算法主要函数，以及文件写入函数路径TEMP





注意释放文件函数，由于V13必须为真，才能进入数据文件写入调用阶段。

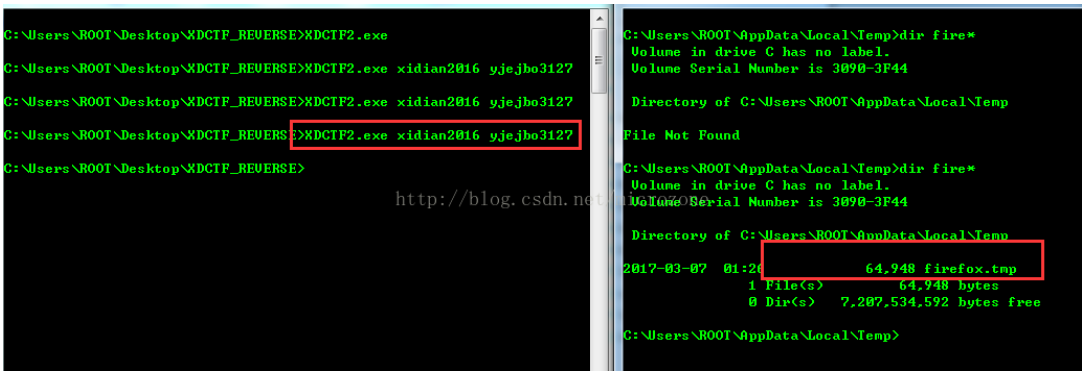
于是网上追溯，分析获取参数1和参数2值为

user:xidian2016

password:yjejbo3127

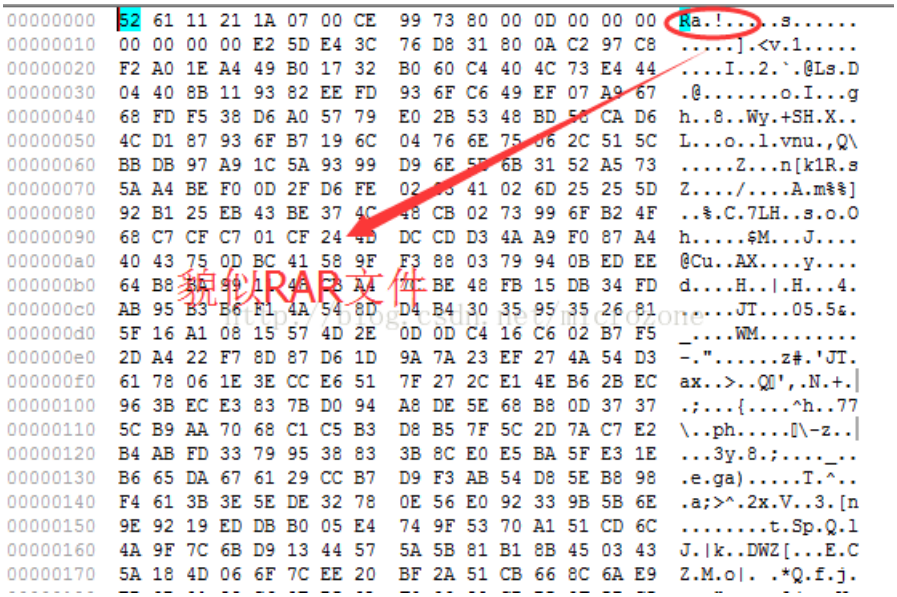
才可以生成文件。

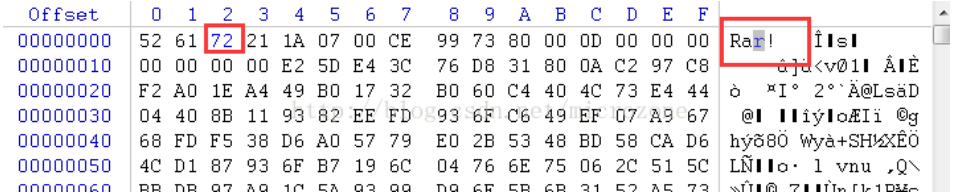
生成 C:\Users\ROOT\AppData\Local\Temp\firefox.tmp文件



转向研究释放的文件，一般思路直接解压或者运行不行或者错误的话，

然后点击十六进制查看，发现类似压缩包文件被破坏，于是采用修复RAR测试



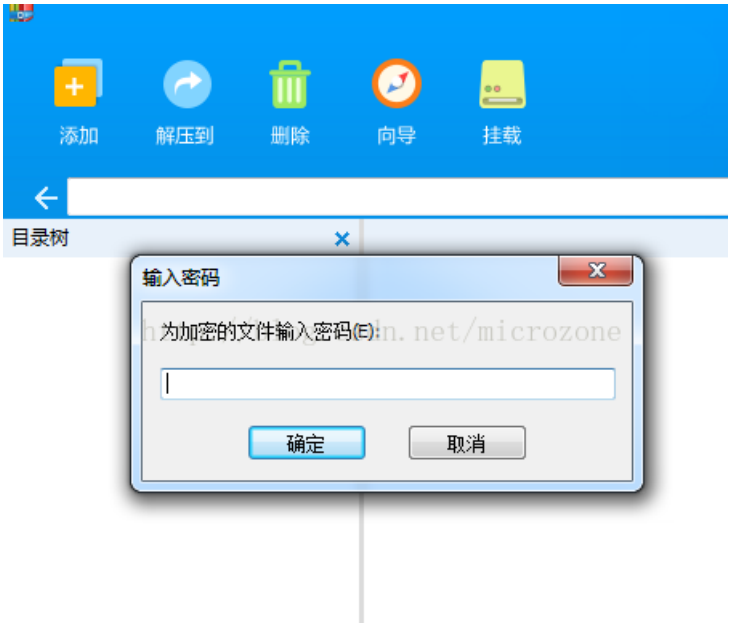


修复之后测试成功，但是解压需要密码。于是猜想前面的用户和密码进行尝试，

尝试了很多次，终于发现解压密码是user+pass拼接而成。于是得到解压后的

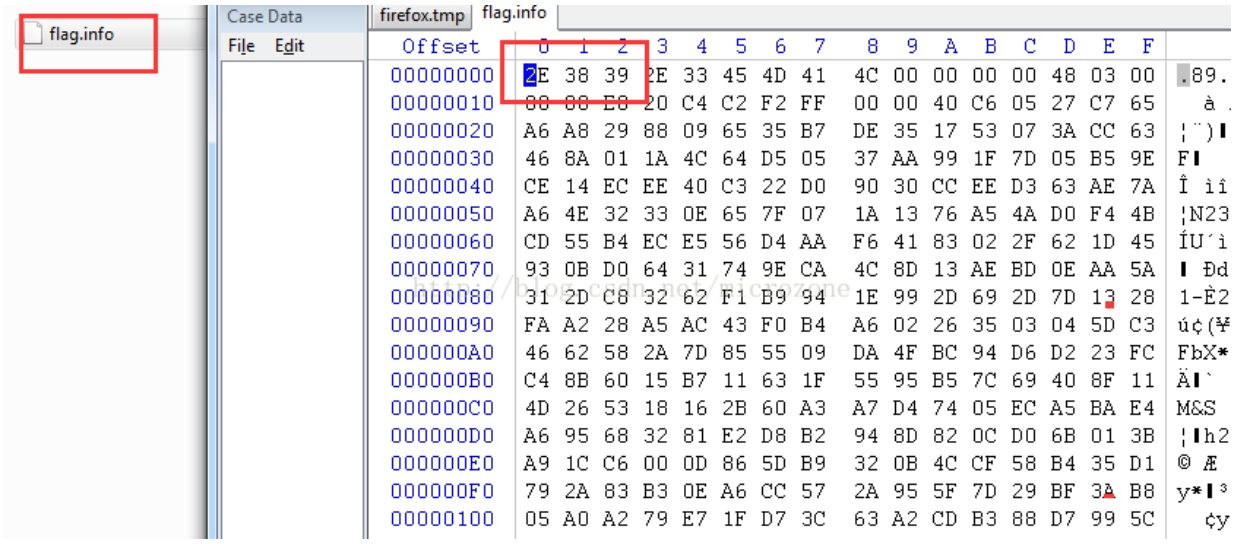
文件内容。

xidian2016yjejbo3127



解压出flag.info文件。我勒个去，还是有文件，不过根据名称应该距离答案很近了。

再继续研究文件类型和内容。



尝试了各种方法，包括修复文件头。还是没有找到，不甘心，想啊想。发现文件的尾部特征，

非常像wav音频文件，可以找个各类文件结构头标志参考。没有想到竟然是倒序存储。

但是却没有完整，于是还是修复，然后写代码实现流倒叙存储。于是乎打开就听到了

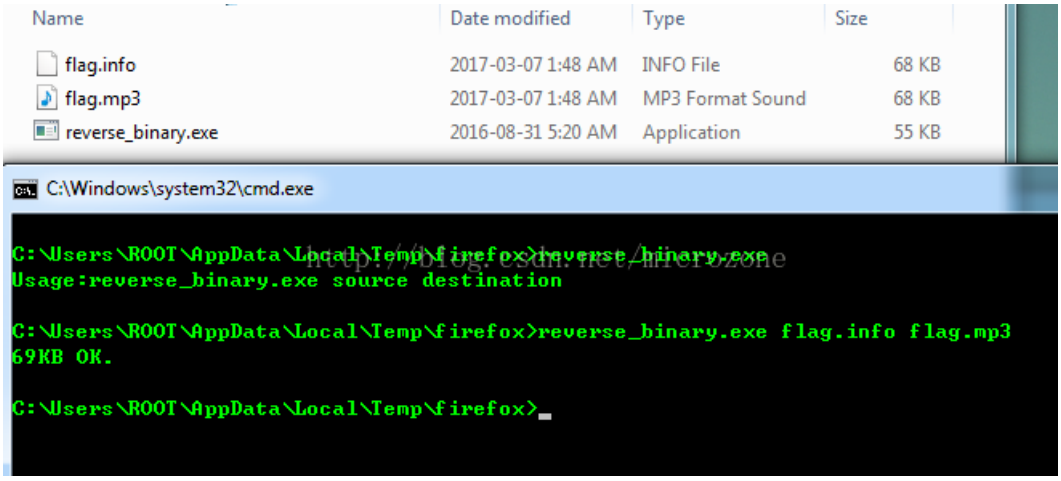
美妙的声音。答案就在音频里面。

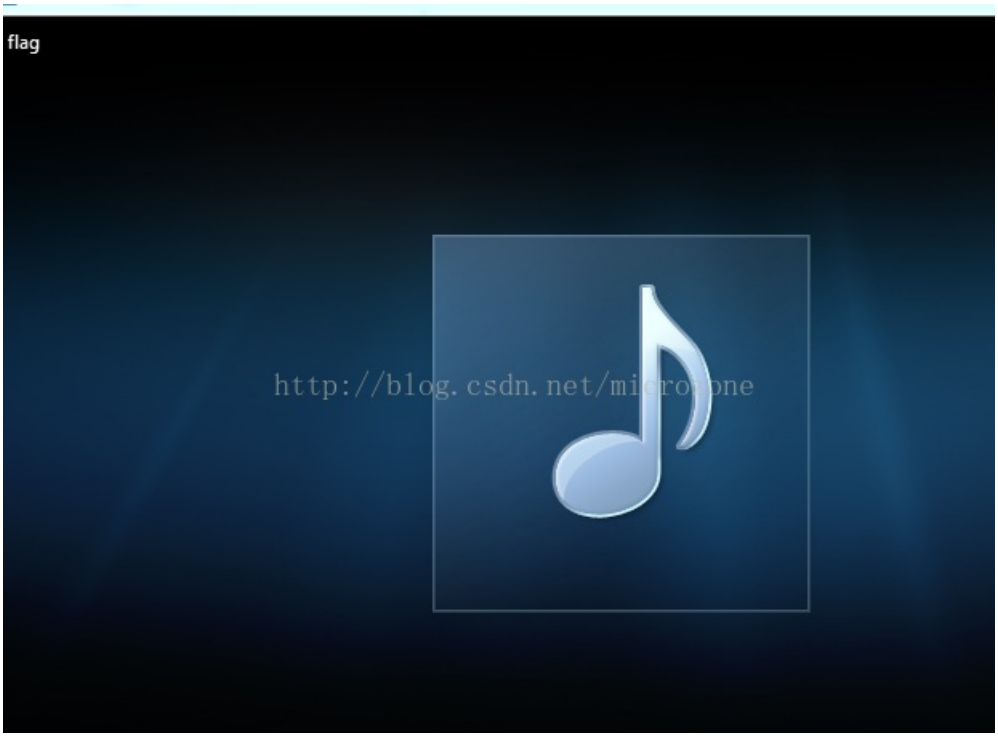
```
00010D00  EF EB EA E7 E4 E2 DF DC  DA D7 D4 D3 D0 CE CB C8  ïëêçäâßÜÚ×ÔÓÐÎËÈ
00010D10  C6 C3 C0 BF BC B9 B7 B4  B1 AF AC A9 A8 A5 A1 A0  ÆÃÀ¿¼¹·´±¯¬©¨¥¡
00010D20  9D 9B 98 95 94 90 8D 8C  89 86 84 81 7E 7C 79 76  ▮▮▮▮ ▮▮▮▮ ~|yv
00010D30  75 72 6F 6D 6A 68 65 62  61 5E 5A 59 56 53 51 4E  uromjheba^ZYVSQN
00010D40  4B 4A 46 43 42 3F 3C 3A  37 35 32 2F 2E 2B 28 26  KJFCB?<:752/.+(&
00010D50  23 20 1E 1B 18 17 14 10  0F 0C 09 07 04 00 05 0F  #
00010D60  01 00 A5 00 00 00 07 00  00 00 6F 66 6E 49 00 00    ¥       ofnI
00010D70  00 00 00 00 00 00 00 C0  C0 F3 FF 00 00 00 00 00        ÀÀóÿ
00010D80  00 00 00 00 00 00 34 30  31 2E 39 31 2E 35 35 66        401.91.55f
00010D90  76 61 4C 03 00 00 0F 00  00 00 45 53 53 54 23 00  vaL         ESST#
00010DA0  00 00 00 00 04 11 44 49                           DI
```

修复好文件，然后自己编码写程序实现二进制流倒叙存储

```
00010D80  00 00 00 00 00 00 34 30  31 2E 39 31 2E 35 35 66        401.91.55f
00010D90  76 61 4C 03 00 00 0F 00  00 00 45 53 53 54 23 00  vaL         ESST#
00010DA0  00 00 00 00 04 33 44 49                           3DI
```

倒序输出结果如下。播放，仔细听答案，多听几遍测试。





输出结果：

XDFLAG{A1B2C3SUCCESSD4F5G6}

0x03 学习总结

此题目没有提示，没有任何输出。先得逆向分析查看程序逻辑和功能。

要识别出干扰的花指令和去除冗余的代码部分。中间出现一些的错误分支。

同时有文件释放写入。此处信息关键，主要注意学习不同文件尾部和

头部特征，以及文件流式操作，最后编码逆序实现成功解题，

拿到音频答案文件。