




# Bugku-杂项部分题目WP

原创

晚安這個未知的世界  于 2020-10-11 12:50:01 发布  1530  收藏 3

分类专栏: [Bugku 杂项](#) 文章标签: [信息安全](#) [加密解密](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: [https://blog.csdn.net/weixin\\_47598409/article/details/109010457](https://blog.csdn.net/weixin_47598409/article/details/109010457)

版权



[Bugku 杂项 专栏收录该内容](#)

1 篇文章 0 订阅

订阅专栏

## 前言

一个新手学习CTFer的成长之路必定要通过大量的刷题, 下面是bugku部分题目wp

## Bugku-MISC-论剑

Challenge

870 Solves



# 论剑 100

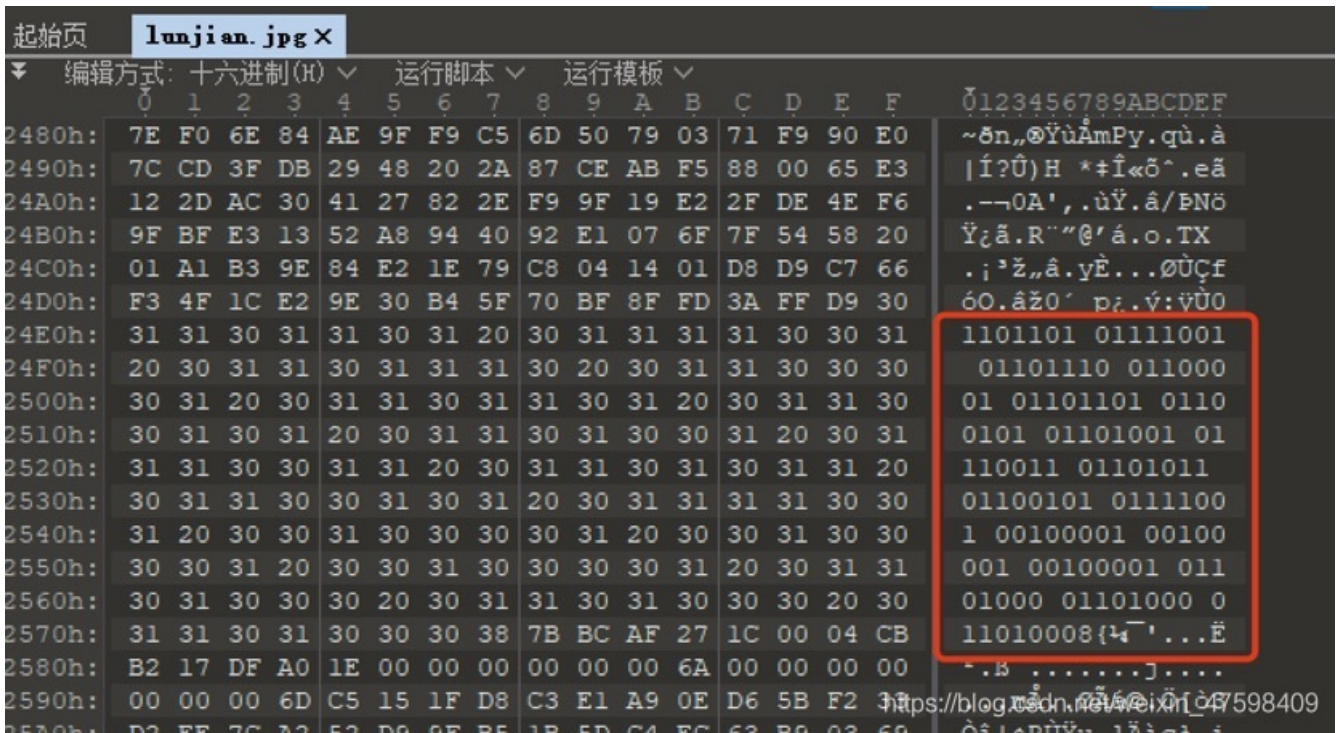
剑客

十年磨一剑, 霜刃未曾试。  
今日把示君, 谁有不平事。

lunjian.jpg

Flag

Submit



```

1 01101101 01111001 01101110 01100001 01101101 01100101 01101001 01110011 01101011 01100101 01111001
00100001 00100001 00100001 01101000 01101000 01101000

```

```

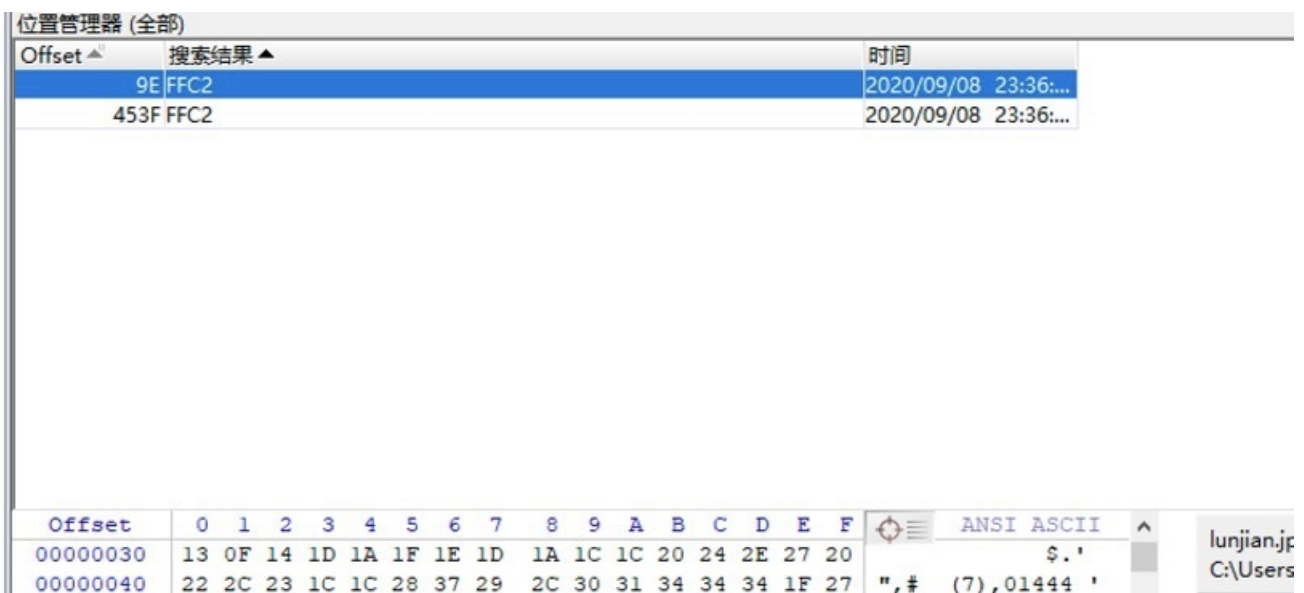
1 mynameiskey!!!hhh

```

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

把图片放进010发现有一串二进制先转ASCII码获得一个关键词，但不知道是什么东西

然后通过搜索FFC2来修改JPEG图片的高度，把00改为01



00000050	39 3D 38 32 3C 2E 33 34	32 FF DB 00 43 01 09 09	9=82<.342yŪ C	文件大小 缺省编辑 状态: 撤消级数 反向撤消 创建时间
00000060	09 0C 0B 0C 18 0D 0D 18	32 21 1C 21 32 32 32 32	2! !2222	
00000070	32 32 32 32 32 32 32 32	32 32 32 32 32 32 32 32	2222222222222222	
00000080	32 32 32 32 32 32 32 32	32 32 32 32 32 32 32 32	2222222222222222	
00000090	32 32 32 32 32 32 32 32	32 32 32 32 32 32 FF C2	22222222222222yÄ	
000000A0	00 11 08 01 42 03 0D 03	01 22 00 02 11 01 03 11	B "	
000000B0	01 FF C4 00 1B 00 01 00	02 03 01 01 00 00 00 00	yÄ	
000000C0	00 00 00 00 00 00 00 04	05 02 03 06 01 07 FF C4	yÄ	
000000D0	00 16 01 01 01 01 00 00	00 00 00 00 00 00 00 00		
000000E0	00 00 00 00 01 02 FF DA	00 0C 03 01 00 02 10 03	https://blog.csdn.net/weixin_47598409	

# CTF\_论剑场

Not flag{666C61677B6D795F6E61656461216768686868}

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI ASCII
00002460	53	48	19	9A	77	F3	D6	AF	92	E0	47	49	BD	30	40	34	SH šwóC' àGI:0@4
00002470	9C	B8	CE	3D	0B	B2	16	93	78	D6	CF	BC	D3	F6	CE	9F	α,î= "xŌI4ŌŌîŸ
00002480	7E	F0	6E	84	AE	9F	F9	C5	6D	50	79	03	71	F9	90	E0	~šn,,šŸùĂmPy qù à
00002490	7C	CD	3F	DB	29	48	20	2A	87	CE	AB	F5	88	00	65	E3	í?Ū)H *+î«š^ eā
000024A0	12	2D	AC	30	41	27	82	2E	F9	9F	19	E2	2F	DE	4E	F6	--ŌA',.ùŸ á/ŲNš
000024B0	9F	BF	E3	13	52	A8	94	40	92	E1	07	6F	7F	54	58	20	Ÿčā R"@"á o TX
000024C0	01	A1	B3	9E	84	E2	1E	79	C8	04	14	01	D8	D9	C7	66	¡ž,,ā yÈ ØŪčf
000024D0	F3	4F	1C	E2	9E	30	B4	5F	70	BF	8F	FD	3A	FF	D9	30	óC āžŌ' pč ý:yŪŌ
000024E0	31	31	30	31	31	30	31	20	30	31	31	31	31	30	30	31	1101101 01111001
000024F0	20	30	31	31	30	31	31	31	30	20	30	31	31	30	30	30	01101110 011000
00002500	30	31	20	30	31	31	30	31	31	30	31	20	30	31	31	30	01 01101101 0110
00002510	30	31	30	31	20	30	31	31	30	31	30	30	31	20	30	31	0101 01101001 01
00002520	31	31	30	30	31	31	20	30	31	31	30	31	30	31	20		110011 01101011
00002530	30	31	31	30	30	31	30	31	20	30	31	31	31	31	30	30	01100101 0111100
00002540	31	20	30	30	31	30	30	30	30	31	20	30	30	31	30	30	1 00100001 00100
00002550	30	30	31	20	30	30	31	30	30	30	30	31	20	30	31	31	001 00100001 011
00002560	30	31	30	30	30	20	30	30	31	30	31	30	30	30	20	30	01000 01101000 0
00002570	31	31	30	31	30	30	30	30	38	7B	BC	AF	27	1C	00	04	11010008(4' Ě
00002580	B2	17	DF	A0	1E	00	00	00	00	00	00	6A	00	00	00	00	" B j
00002590	00	00	00	6D	C5	15	1F	D8	C3	E1	A9	0E	D6	5B	F2	33	mĂ ŌĂăĊ Ō[ò3
000025A0	D2	FF	7C	A2	52	D9	9F	B5	1B	5D	C4	FC	63	B9	03	69	Ōi!cRŪŸu lĂi c' i

这里猜出应该是一个7z压缩包文件头，应该被出题人改了文件头  
7z的文件头是377ABCAF271C，然后修改为正确头部后并且修改文件后缀名为7z





发现了有一个图片，但是这个压缩包被加密了，但是回想一下，开始的时候用二进制转ASCII码有一串东西，猜一猜是不是密码？

好家伙，果然是密码

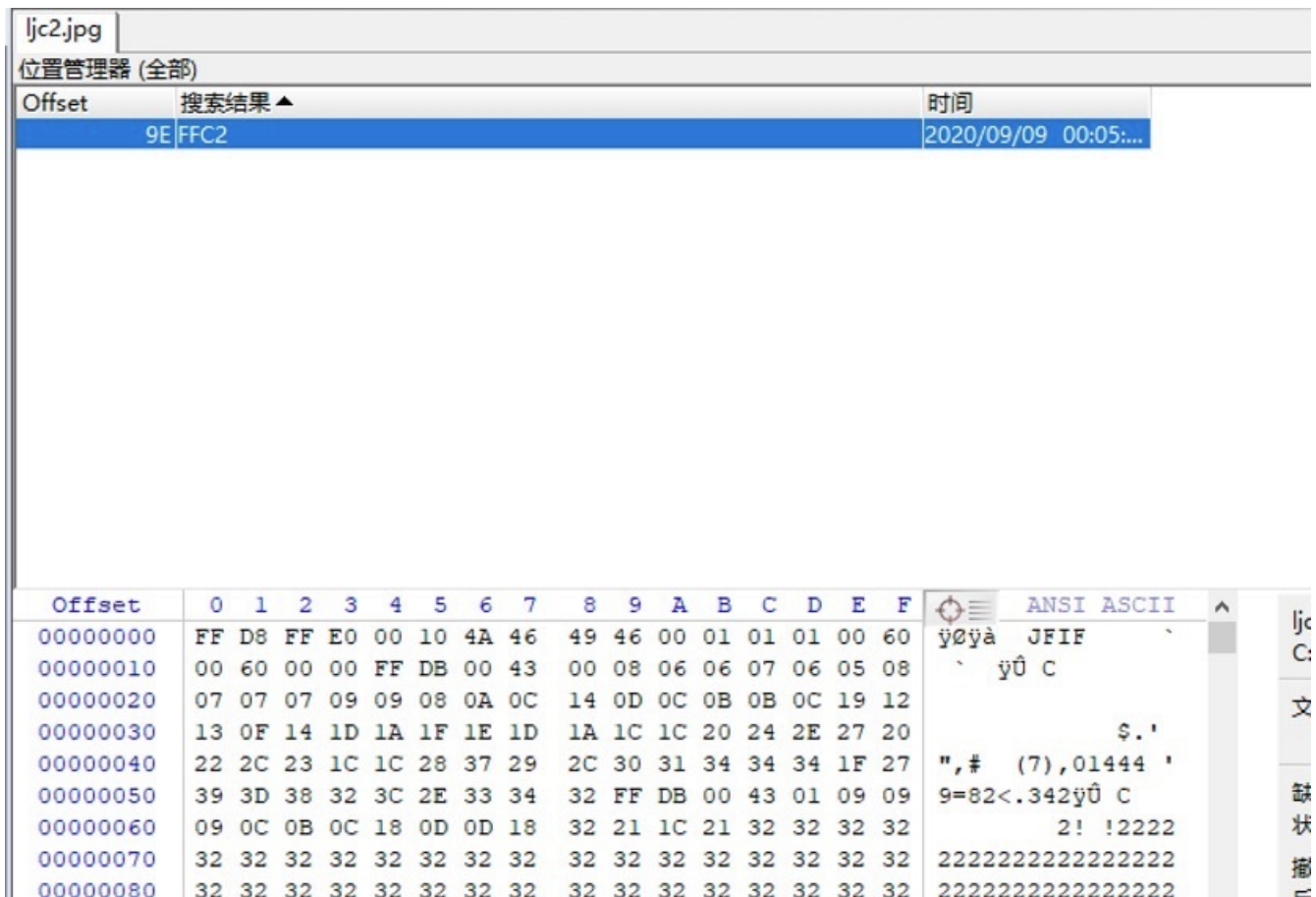
lunjian

共享 查看

> lunjian



但是这张图片跟刚才的那个也是图片高度被修改了，继续用winhex或者010修改红色框的00改为01，保存





```
00000090 32 32 32 32 32 32 32 32 32 32 32 32 FF C2 2222222222222222ÿÄ
000000A0 00 11 08 01 3E 02 6E 03 01 22 00 02 11 01 03 11  n "
000000B0 01 FF C4 00 1B 00 01 00 02 03 01 01 00 00 00 00  yÄ
000000C0 00 00 00 00 00 00 00 03 04 02 05 06 01 07 FF C4  yÄ
000000D0 00 16 01 01 01 01 00 00 00 00 00 00 00 00 00 00  https://blog.csdn.net/weixin_47598409
000000E0 00 00 00 00 01 02 FF DA 00 0C 03 01 00 02 10 03  vñ
```

# CTF\_论剑场

Not flag{66C61677B6D795F6E616D655F482121487D} hhhh

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

然后把两张这样的图片的flag数据合在一起

flag{666C61677B6D795F6E616D655F482121487D}

然后去提交，发现不正确，再仔细看了一下flag里面的的是十六进制的数值，我们去转换一下ASCII码

## 16进制到ASCII字符串

```
1 666C61677B6D795F6E616D655F482121487D
```

🗑️ 清空    ↕ 交换位置    📄 示例    **➡ 转换**    📄 保存结果    📄 复制结果

```
1 flag{666C61677B6D795F6E616D655F482121487D} hhhh
```

```
flag{my_name_n:!!n}
```

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

flag就这样出来了

```
flag{my_name_H!!H}
```

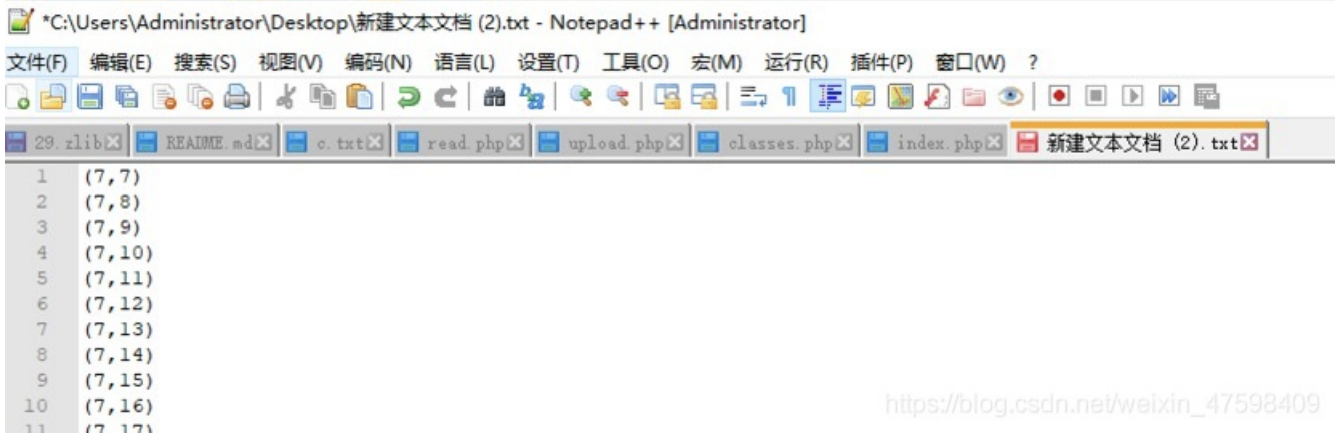
## Bugku-MISC-图穷匕见

用winhex打开，结果发现下面有一大串16进制数据，FF D9是JPG文件尾

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI	ASCII
000052D0	06	A2	A7	2F	54	21	11	68	42	15	02	10	85	00	84	21	c\$/T!	hB ... !!
000052E0	00	84	21	50	24	84	28	1A	10	84	02	10	85	40	84	21	..!P\$..( .. ..@..!	
000052F0	00	84	21	07	FF	D9	32	38	33	37	32	63	33	37	32	39	..! yÙ20072c0720	
00005300	30	61	32	38	33	37	32	63	33	38	32	39	30	61	32	38	0a28372c38290a28	
00005310	33	37	32	63	33	39	32	39	30	61	32	38	33	37	32	63	372c39290a28372c	
00005320	33	31	33	30	32	39	30	61	32	38	33	37	32	63	33	31	3130290a28372c31	
00005330	33	31	32	39	30	61	32	38	33	37	32	63	33	31	33	32	31290a28372c3132	
00005340	32	39	30	61	32	38	33	37	32	63	33	31	33	33	32	39	290a28372c313329	
00005350	30	61	32	38	33	37	32	63	33	31	33	34	32	39	30	61	0a28372c3134290a	
00005360	32	38	33	37	32	63	33	31	33	35	32	39	30	61	32	38	28372c3135290a28	
00005370	33	37	32	63	33	31	33	36	32	39	30	61	32	38	33	37	372c3136290a2837	
00005380	32	63	33	31	33	37	32	39	30	61	32	38	33	37	32	63	2c3137290a28372c	
00005390	33	31	33	38	32	39	30	61	32	38	33	37	32	63	33	31	3138290a28372c31	
000053A0	33	39	32	39	30	61	32	38	33	37	32	63	33	32	33	30	39290a28372c3230	
000053B0	32	39	30	61	32	38	33	37	32	63	33	32	33	31	32	39	290a28372c323129	
000053C0	30	61	32	38	33	37	32	63	33	32	33	32	32	39	30	61	0a28372c3232290a	

然后复制这些数据

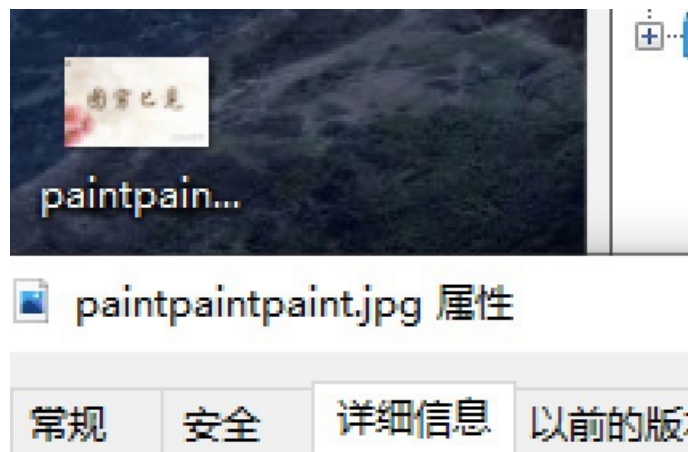
用Notepad++打开，并用这个软件里面的一个插件进行转换 HEX—》ASCII码



[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

看上去有点像坐标点

再看一下图片属性信息

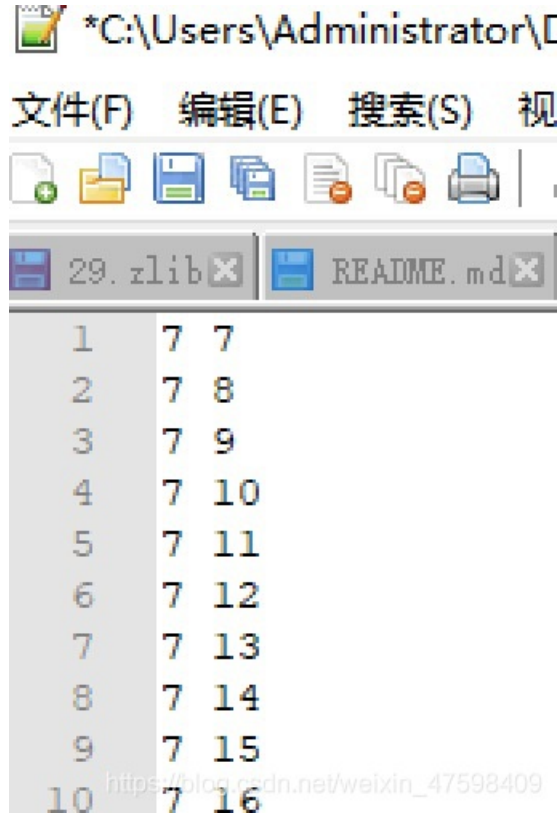


属性	值
说明	
标题	图穷flag见
主题	会画图吗?
分级	☆☆☆☆☆
标记	
备注	气氛搞起来!
来源	
作者	<a href="https://blog.csdn.net/qq_398409">https://blog.csdn.net/qq_398409</a> 出题人已跑路~

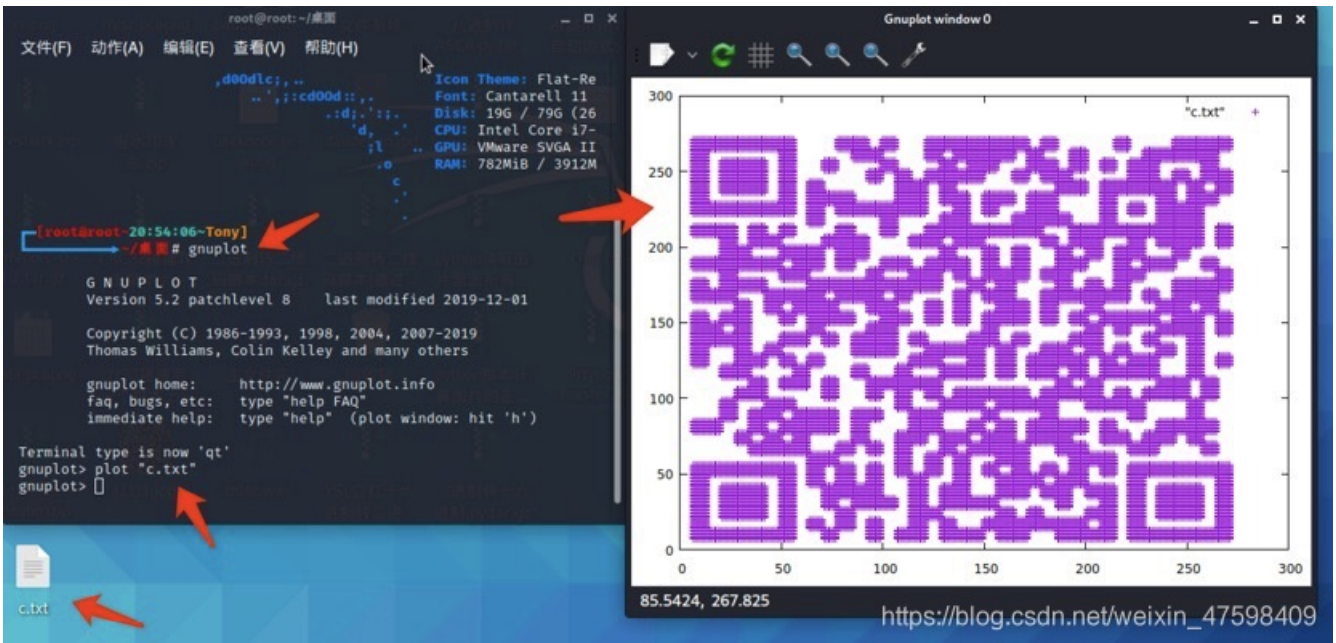
会画图吗???



进行一番的替换，把括号和逗号都替换掉



Linux系统里面有一个专门画图工具，根据坐标数值画图——gnuplot



扫描二维码得出flag

flag{40fc0a979f759c8892f4dc045e28b820}

## Bugku-MISC-Convert









常规 安全 详细信息 以前的版本

属性	值
说明	
标题	
主题	ZmxhZ3swMWEyNwVhM2ZkNjM0OwMz
分级	☆☆☆☆☆

https://blog.csdn.net/weixin\_47598409

Base64解密获得flag

文字加密解密 MD5加密/解密 URL加密 JS加/解密 JS混淆加密压缩 ESCAPE加/解密 **BASE64** 散列哈希 迅雷, 快车, 旋风URL加解密

flag{01a25ea3fd6349c6e635a1d0196e75fb}	ZmxhZ3swMWEyNwVhM2ZkNjM0OwMzZTYzNWExZDAxOTZlZmYnO=
--	--

flag{01a25ea3fd6349c6e635a1d0196e75fb}

### Bugku-MISC-听首音乐

Challenge 1541 Solves ×

# 听首音乐

## 150

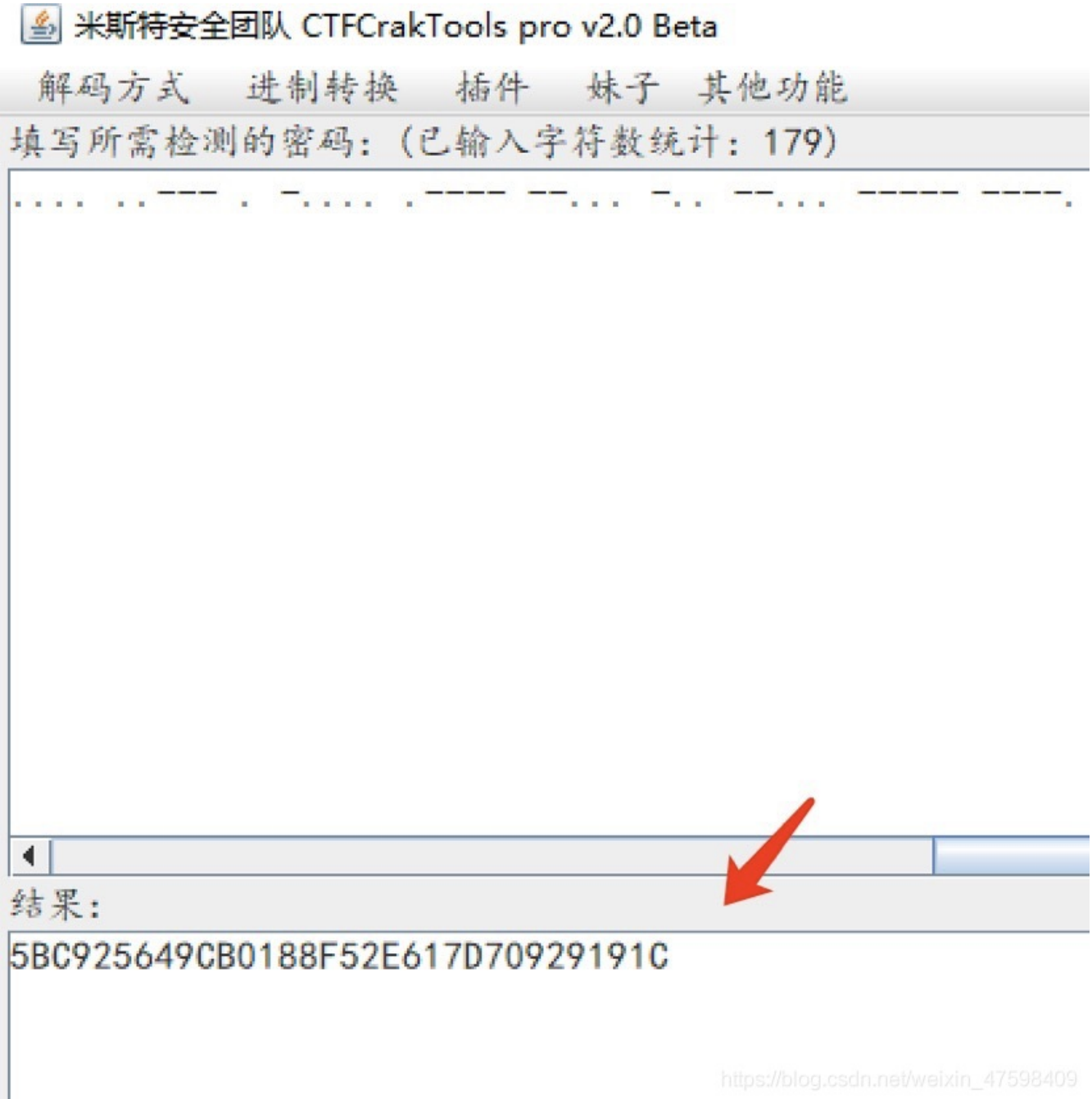
听首音乐放松放松吧~

下载地址: 链接: <http://pan.baidu.com/s/1gfvezBI> 密码: y6gh

Flag

Submit

直接用Audacity打开就可以发现是摩斯密码



The screenshot shows the CTFCrakTools pro v2.0 Beta application window. At the top, there is a title bar with the application name and a menu bar with options: 解码方式 (Decode Method), 进制转换 (Base Conversion), 插件 (Plugins), 妹子 (Girls), and 其他功能 (Other Functions). Below the menu bar, a text field contains the instruction: 填写所需检测的密码: (已输入字符数统计: 179). The main area of the application is mostly empty, with some faint, illegible text visible. At the bottom, there is a results section labeled 结果: (Results:). A red arrow points to the result text: 5BC925649CB0188F52E617D70929191C. A URL is visible in the bottom right corner: [https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409).

直接解码获得flag

5BC925649CB0188F52E617D70929191C

**Bugku-MISC-好多数值**

Challenge

1140 Solves



# 好多数值

## 150

flag格式 flag{}

1.txt

Flag

Submit

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

← → × [ctf.bugku.com/files/093d4073de2c7bfac7466fd166c5d990/1.txt](https://ctf.bugku.com/files/093d4073de2c7bfac7466fd166c5d990/1.txt)

应用 百度 淘宝 京东 天猫 苏宁易购 Gmail YouTube

```
255, 255, 255
255, 255, 255
255, 255, 255
255, 255, 255
255, 255, 255
255, 255, 255
255, 255, 255
255, 255, 255
255, 255, 255
255, 255, 255
255, 255, 255
255, 255, 255
255, 255, 255
255, 255, 255
255, 255, 255
255, 255, 255
255, 255, 255
255, 255, 255
255, 255, 255
255, 255, 255
```

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

我猜这应该是RGB数值

借用大佬的一个脚本“根据RGB数值自动因式分解画图.py”

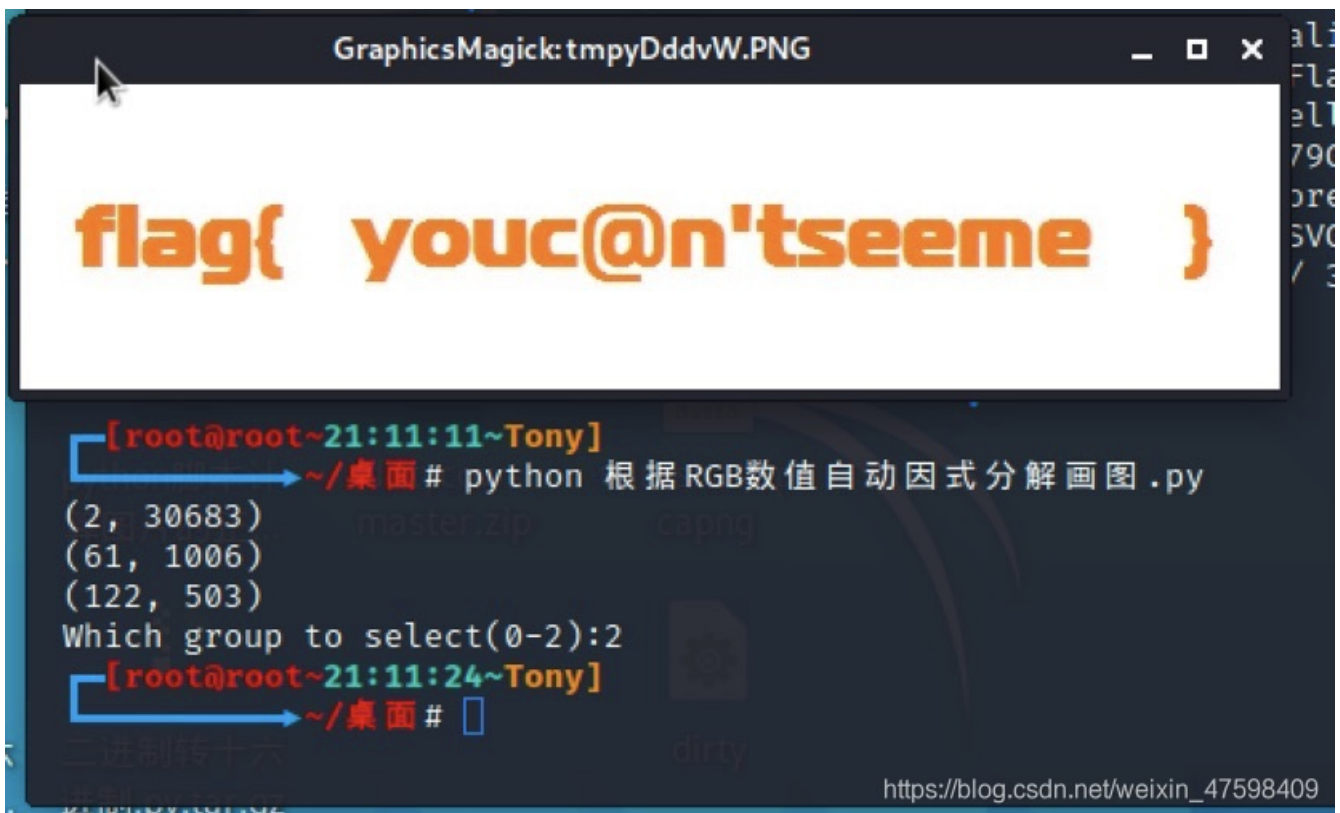


网上好多大佬解这题用的脚本是已经定义好X和Y，而我找的大佬脚本是自动进行因式分解，自动生成，可用性应该高一点

```
from PIL import Image

def Crack(n):#yinshufenjie
    flag = []
    for each in range(2,int(n **0.5)+1):
        if(n % each == 0):
            print(each,int(n/each))
            flag += [(each,int(n/each))]
    if len(flag) == 1:return flag[0]
    else:
        choice = input("Which group to select(0-%s):"%(len(flag)-1))
        return flag[int(choice)]

def Paint(X,Y,listrgb):#Draw according to string List
    pic = Image.new("RGB", (X, Y))
    i=0
    for x in range (0,X):
        for y in range (0,Y):
            temp = listrgb[i].split(',')
            pic.putpixel([x,y],(int(temp[0]),int(temp[1]),int(temp[2])))
            i = i+1
    pic.show()
    pic.save(r"/root/flag%s.png"%(X))
listrgb = open(r"/root/1.txt").readlines()
X,Y = Crack(len(listrgb))
Paint(X,Y,listrgb)
Paint(Y,X,listrgb)
```



flag{youc@n'tseeme}

## Bugku-MISC-很普通的数独(ISCCCTF)

# 很普通的数独(ISCCTF)

## 150

zip

Flag

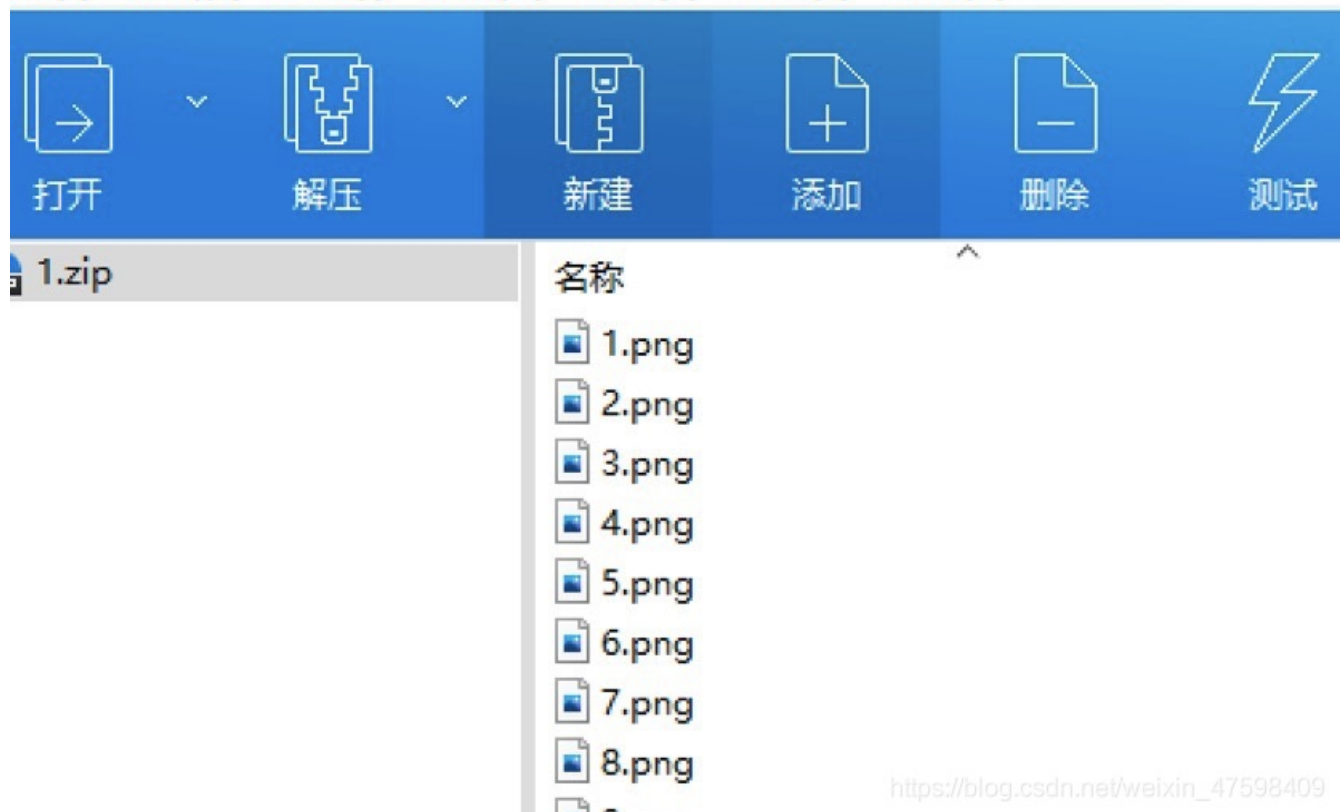
Submit

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

附件有一个zip里面有25张图

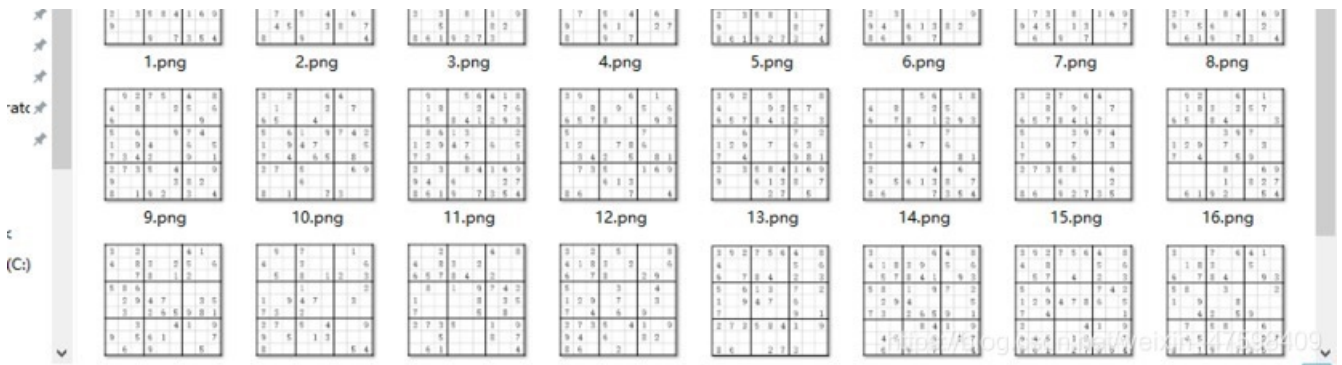
1.zip - Bandizip 7.04 (Professional)

文件(F) 编辑(E) 查找(I) 选项(O) 视图(V) 工具(T) 帮助(H)



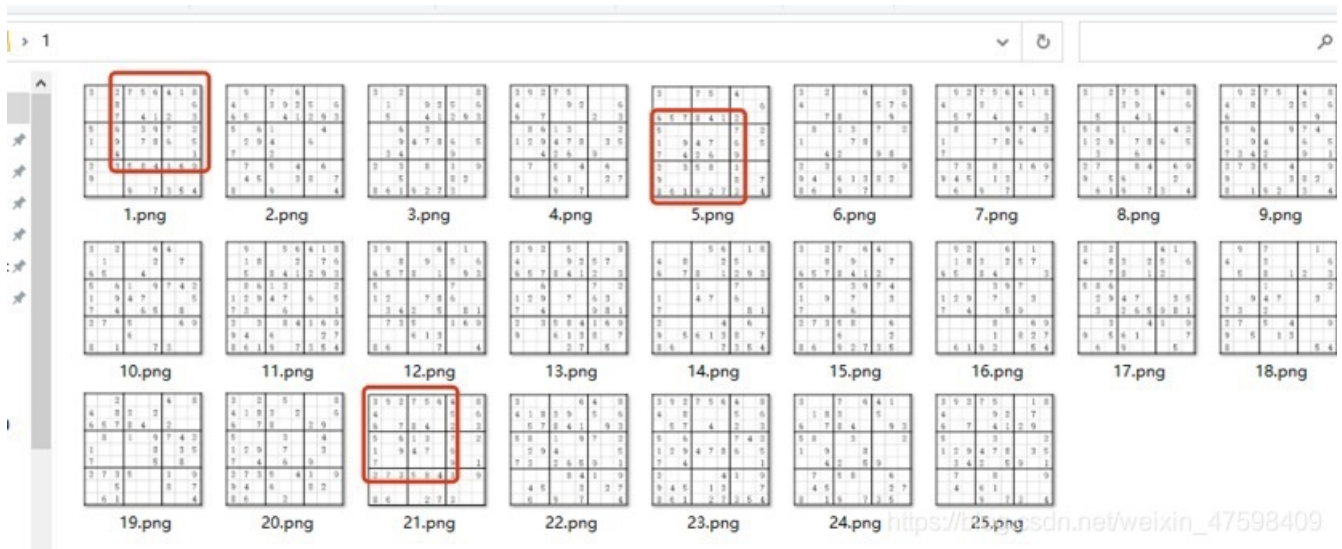
[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)





全是数独图片

然后仔细看了一下1.png 5.png 21.png，好像是二维码的定位符，但是顺序乱了



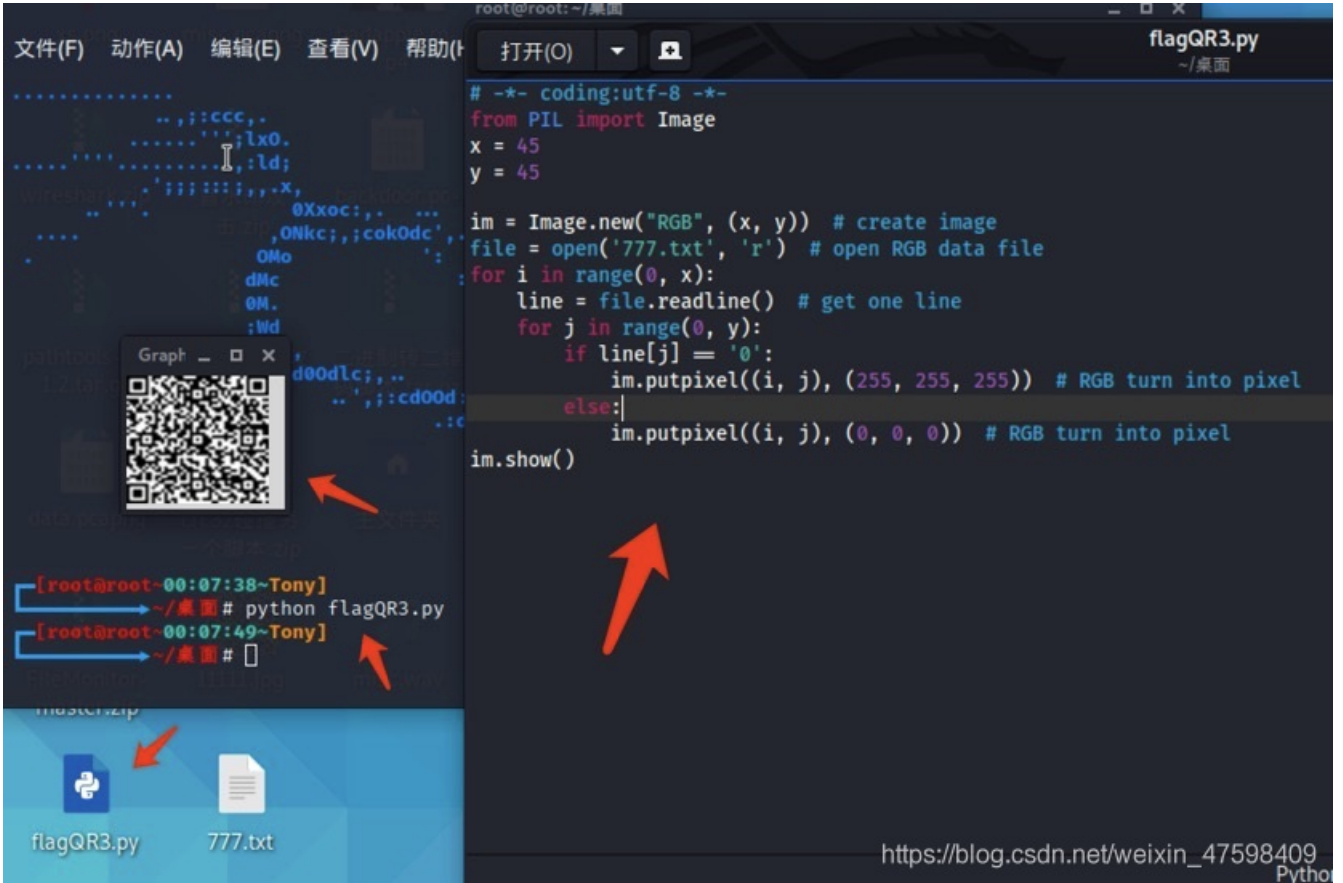
然后把三张图片顺序调一下，5调到21，21调到1，1调到5

调完后，用记事本将信息记录下来，有数字的用1代替，没有的用0代替。

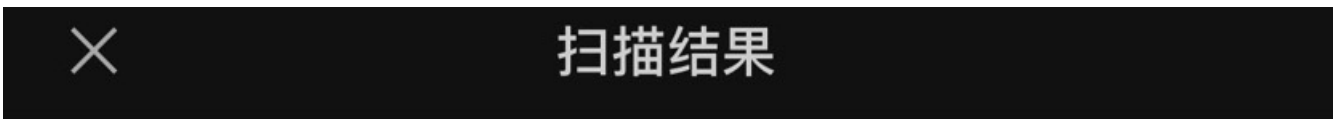


111111010101000101000011111000010111111  
100000101100111101010011101100011001001000001  
101110101110011111010011111101000101001011101  
101110101101100010001010000011110001101011101  
10111010001110010000111110111111011101011101  
100000101100100000011000100001110100001000001  
11111110101010101010101010101010101110111111  
00000000001100110100100011010011001110000000  
11001110010010000111111100100101000000101111  
10100100101111111101110101011110101101001100  
100000111100100100000110001101001101010001010  
001100010011010001010011000100000010110010000  
010110101010001111110100011101001110101101111  
100011000100011100111011101101100101101110001  
001100110100000000010010000111100101101011010  
10100000101101011111001101111101001110100011  
110111110111011001101100010100001110000100000  
110101000010101000011101101101110101101001100  
010011111110001011111010001000011011101101100  
011001011001010101100011110101001100001010010  
010111111111010111111101101101111111111100  
011110001100000100001000101000100100100011110  
111110101110011100111010110100110100101010010  
110010001011101011101000111100000011100010000  
10101111101110011110111111100001010111110010  
110100011000111000100111101101111101000100010  
111101111110001001000011010110001111110111110  
011001010101000110010100010001000101101010001  
011101110101101101100100001101101000111101001  
110110001001101100010101101111110100101100110  
000011100111000000000100001010101111100010010  
111010010011110011101110010100001011111010010  
101001100010111111110100000100001010101010100  
000010011001001101110101001111100101111101101  
000010111101110001101011000001000101110100110  
011110011010100010100000011011000001110010000  
10011010010000110111111101100101110111110011  
000000001111110101101000101011100100100011010  
111111100011111011011010101101110011101011110  
100000101110101101101000111110010001100010001  
10111010101110000111111101101001000111111011  
101110100110111101101000001001101100011101101  
101110100000011101100001101010110010010010001  
100000101011001011111011001011000011010110000  
111111101010101001111011110101101110000101101

然后用py脚本把以上01转换为二维码



扫描二维码得出了一串base64编码



Vm0xd1NtUXlWa1pPVldoVFfIUSlNjRlJVVGtOamJGWnlWM

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

Vm0xd1NtUXlWa1pPVldoVFfIUSlNjRlJVVGtOamJGWnlWMjFHVlUxV1ZqTldNakZlWvcxS1IxTnNhRmhoTVZweVdWUkdXbvZHwkhOWGJGcHBWa1paZWxac1pEUmhNVXBYVW14V2FHVnFRVGs9

经过多次base64解码得出flag



flag{y0ud1any1s1}

Bugku-MISC-PEN\_AND\_APPLE

Challenge

306 Solves



# PEN\_AND\_APPLE

## 150

狗师傅平日里比较害羞，但是又想追女神，于是他隐藏了一段信息在这段自拍中，这句话是他最想对女神说的话:) 你能找到信息，并帮助狗师傅表白成功么:) 视屏在这儿:)

题目来源：第七季极客大挑战

test.mp4

Flag

Submit

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

这题原题的附件是一个rar压缩包里面有一个视频的，但是这个网站给的附件是一个test.mp4文件，应该被人修改了，所以就没做出来，不过可以在网上看到大概的解题思路

这题的flag为SYC{Hei\_hei\_hei}

[Bugku-MISC-color](#)



Challenge

680 Solves



# color 150

你见过彩虹吗?

来源：第七届山东省大学生网络安全技能大赛

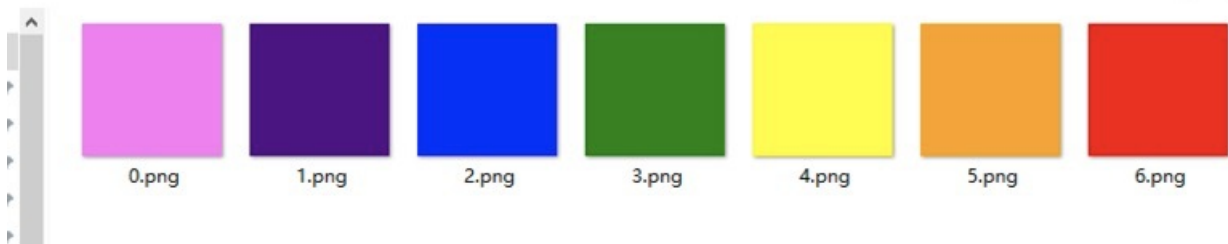
81cf84c2-0901...

Flag

Submit

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

> 81cf84c2-0901-46d8-80c7-f0844e7e5253



有7张图片，看过每张图片的属性信息没可疑线索，用stegsolve打开

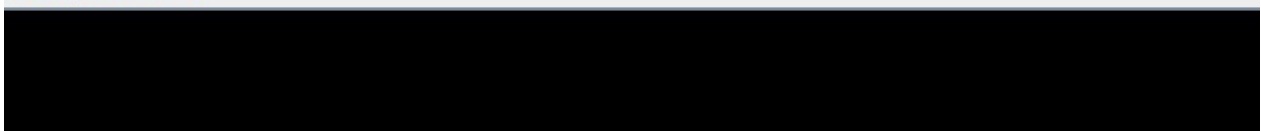


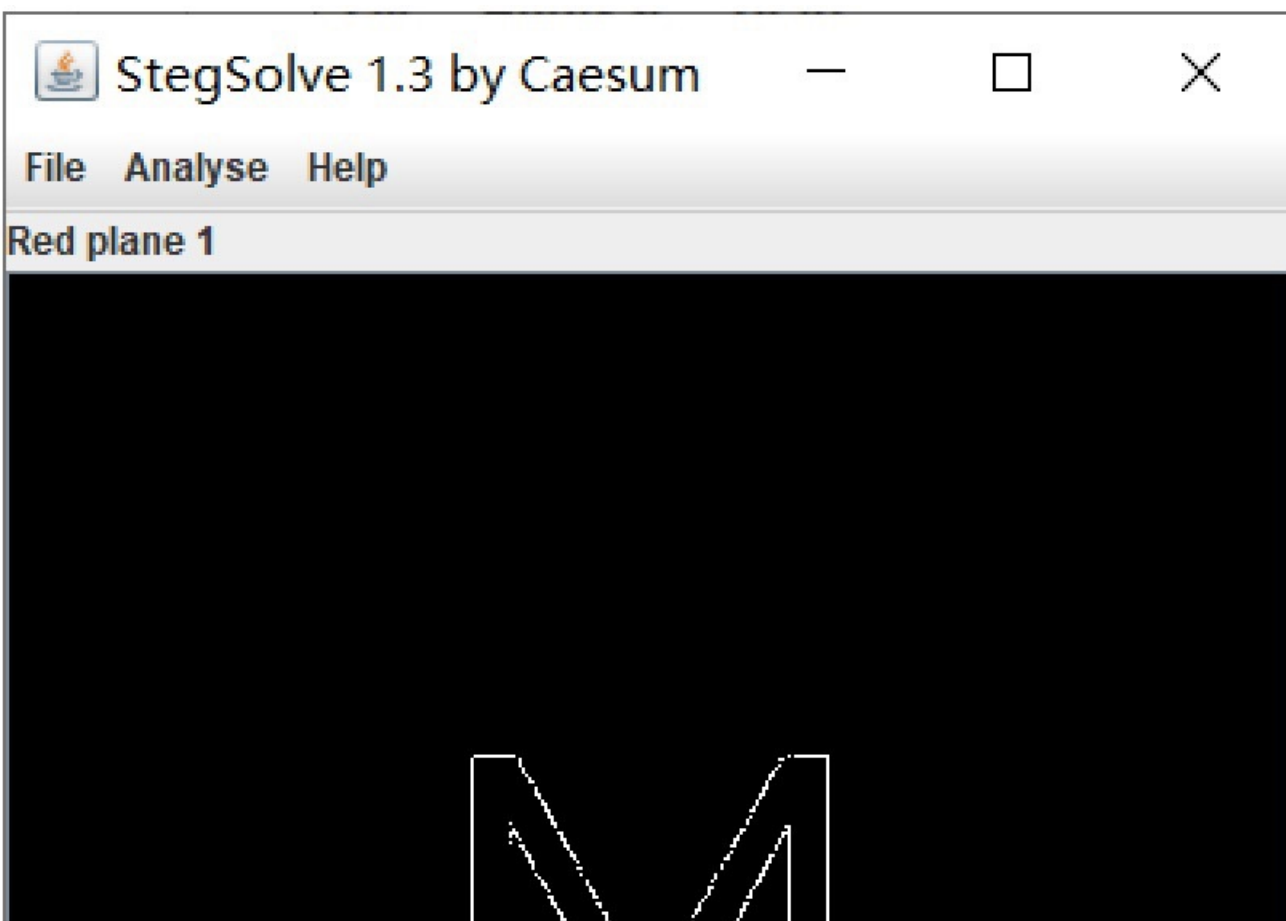
StegSolve 1.3 by Caesum

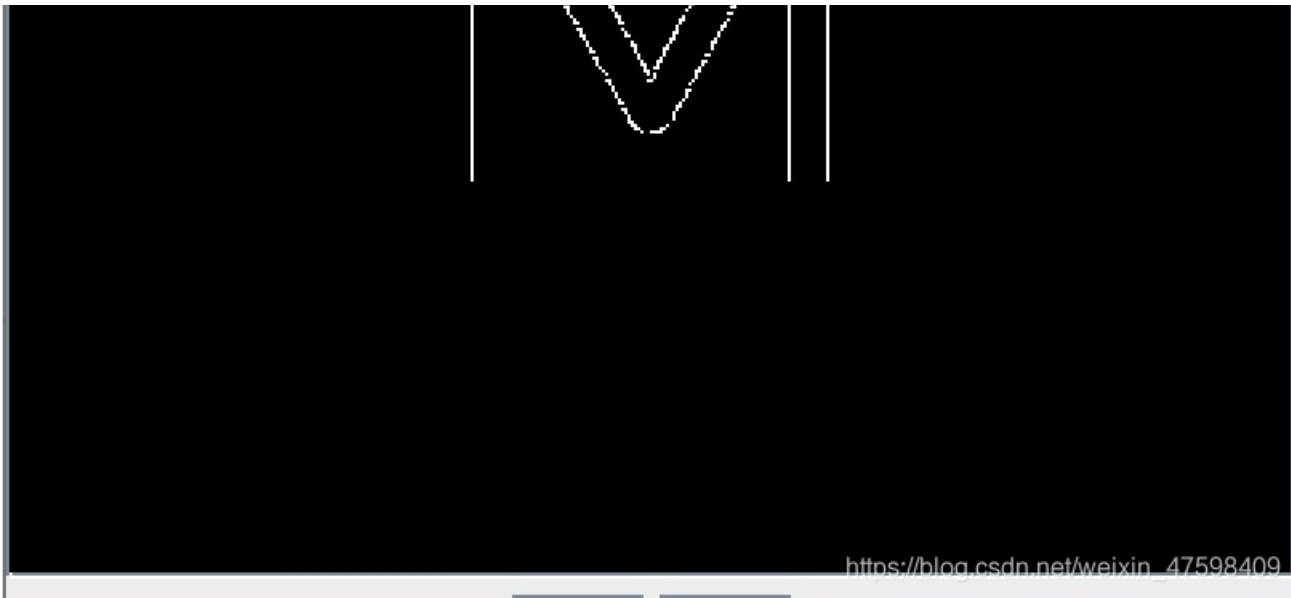


File Analyse Help

Red plane 3







StegSolve 1.3 by Caesum

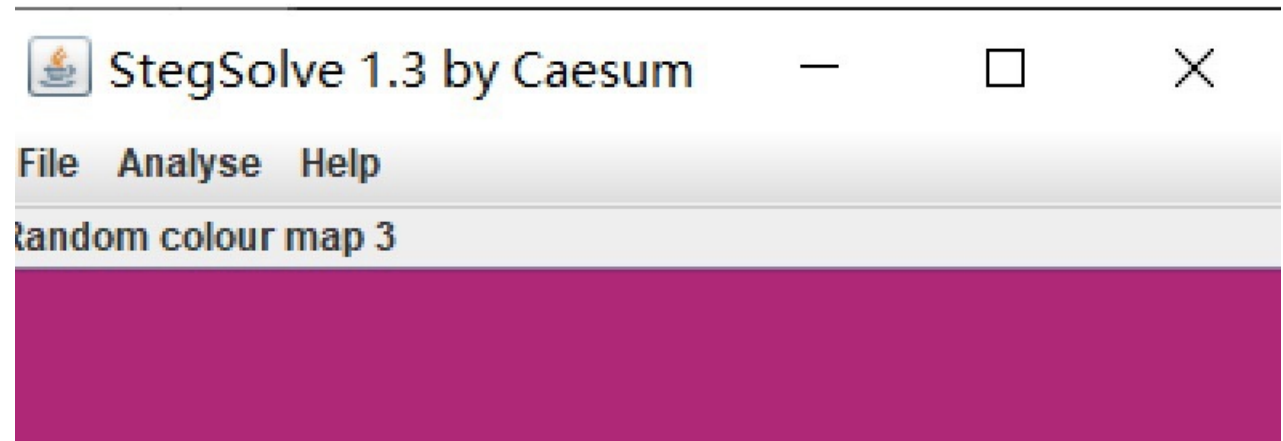


File Analyse Help

Red plane 0

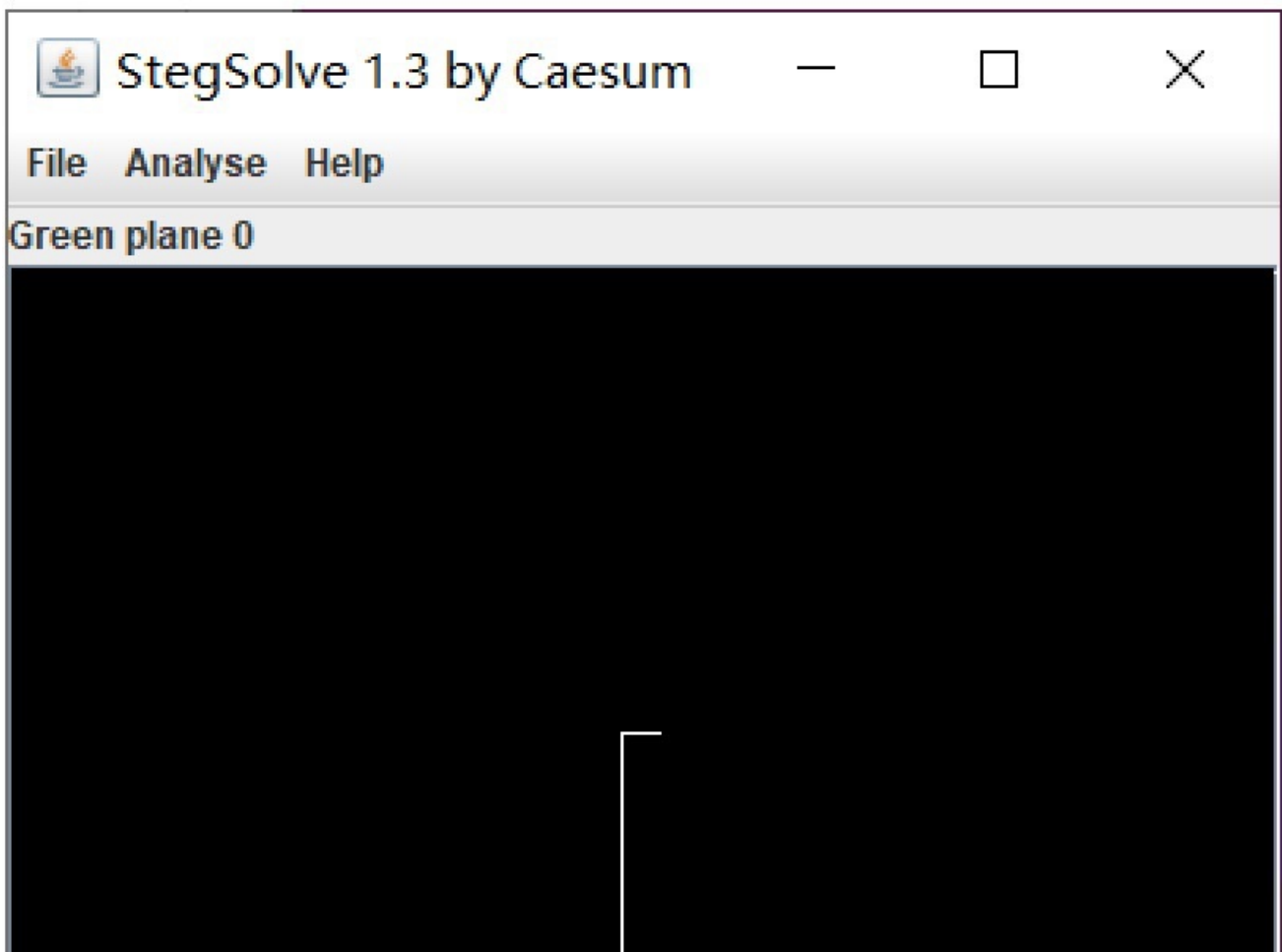


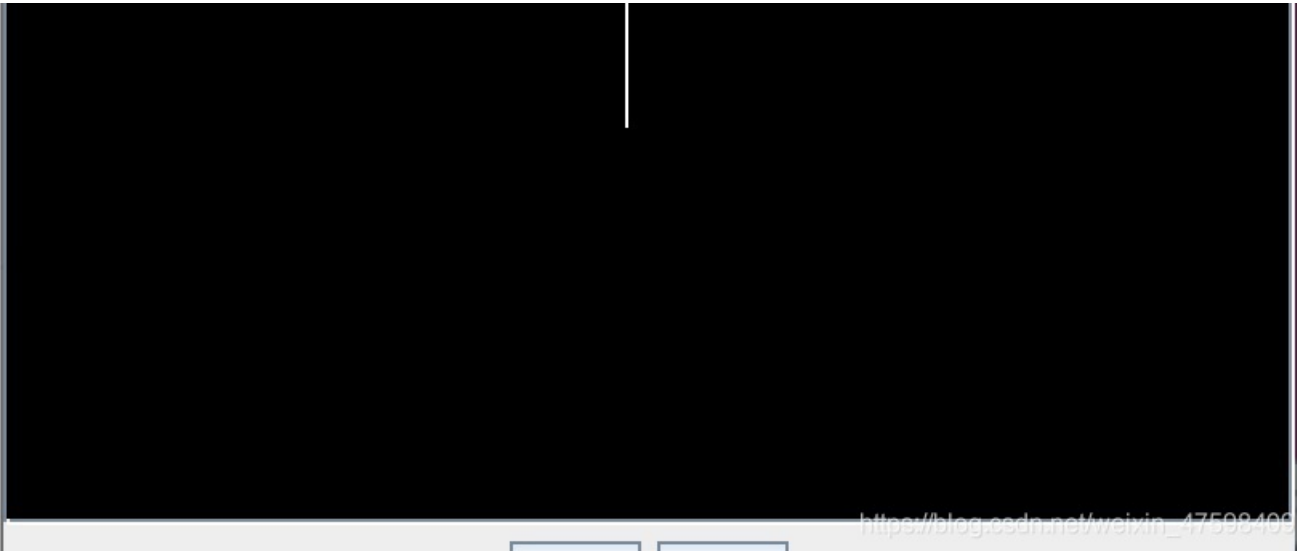







[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

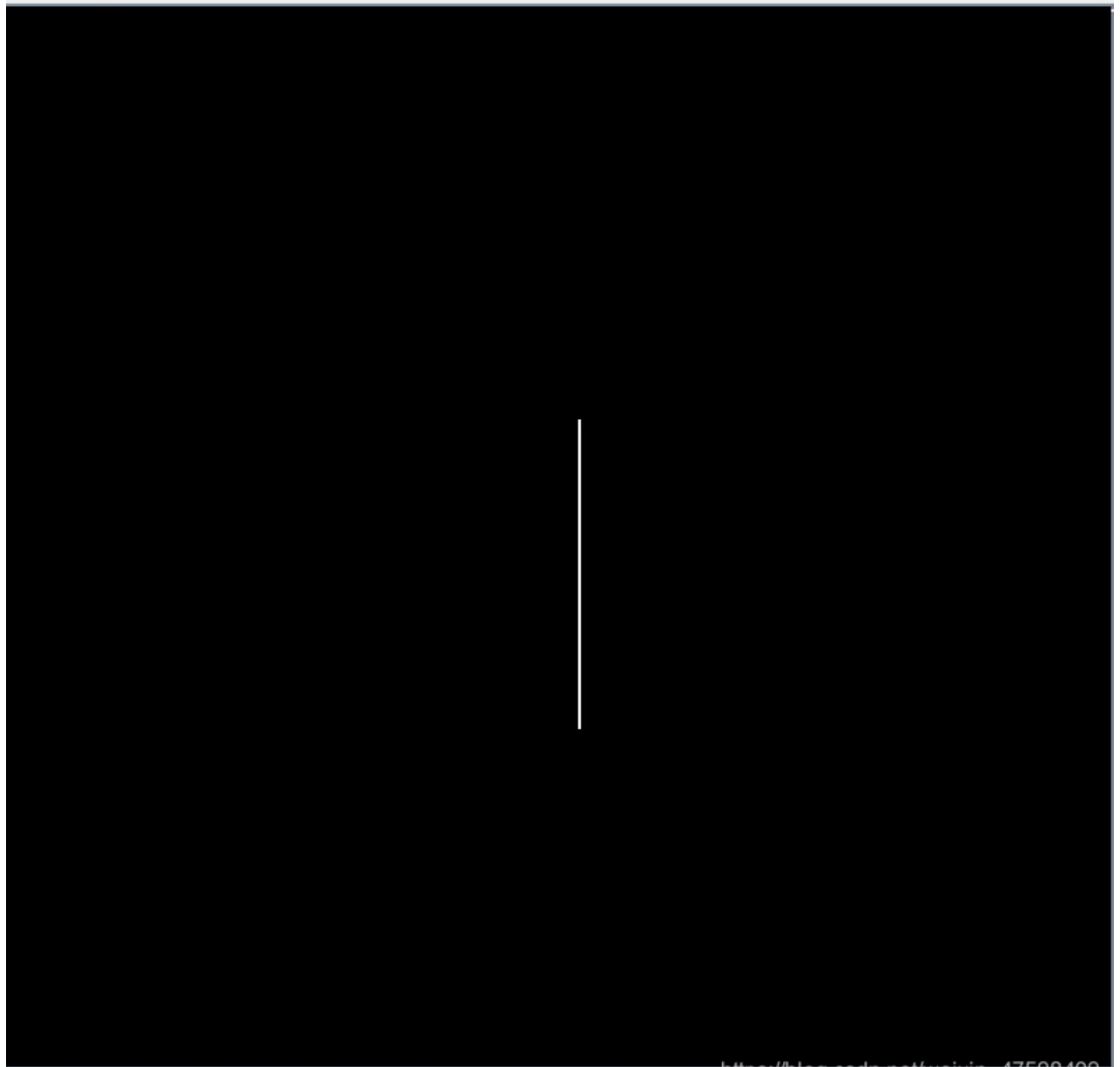




 StegSolve 1.3 by Caesum — □ ×

File Analyse Help

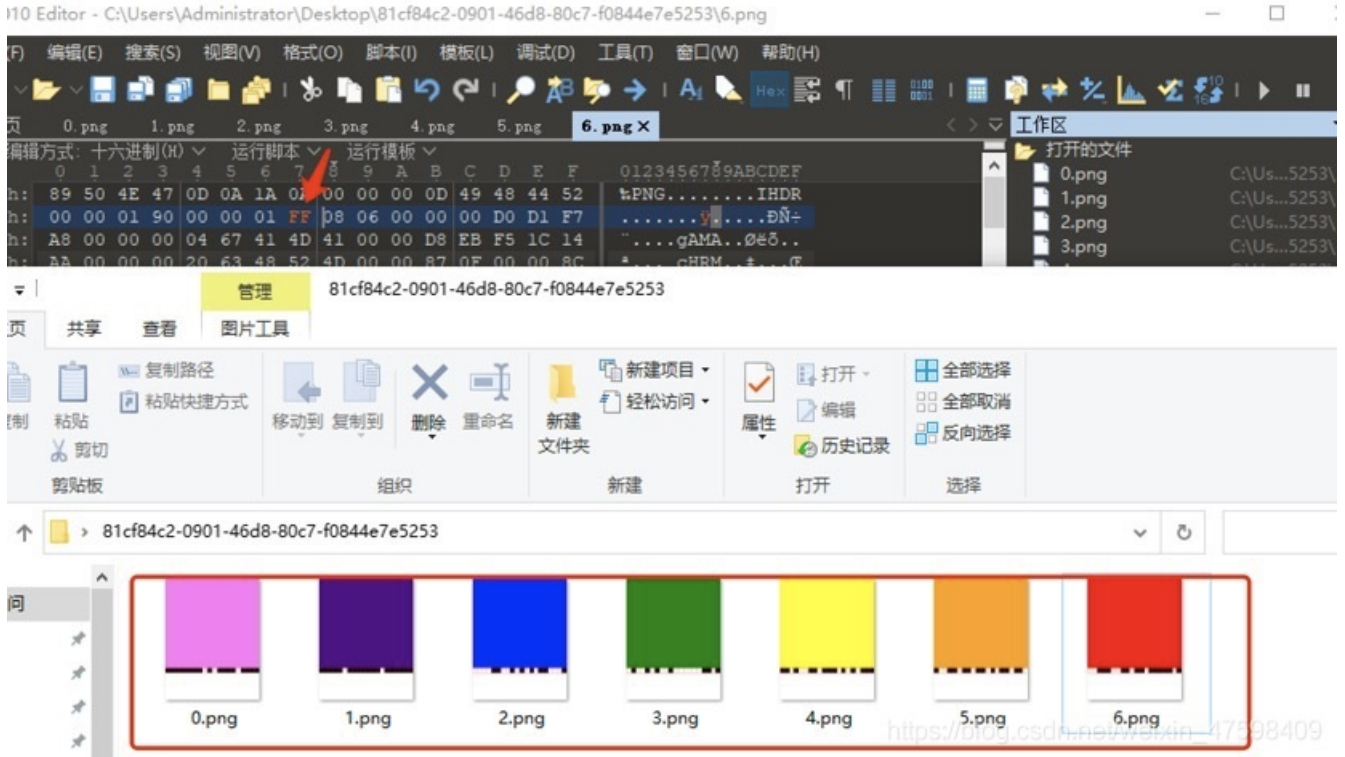
green plane 1



根据7张图片给出的英文字母组成Make Me Tall——使我变高???

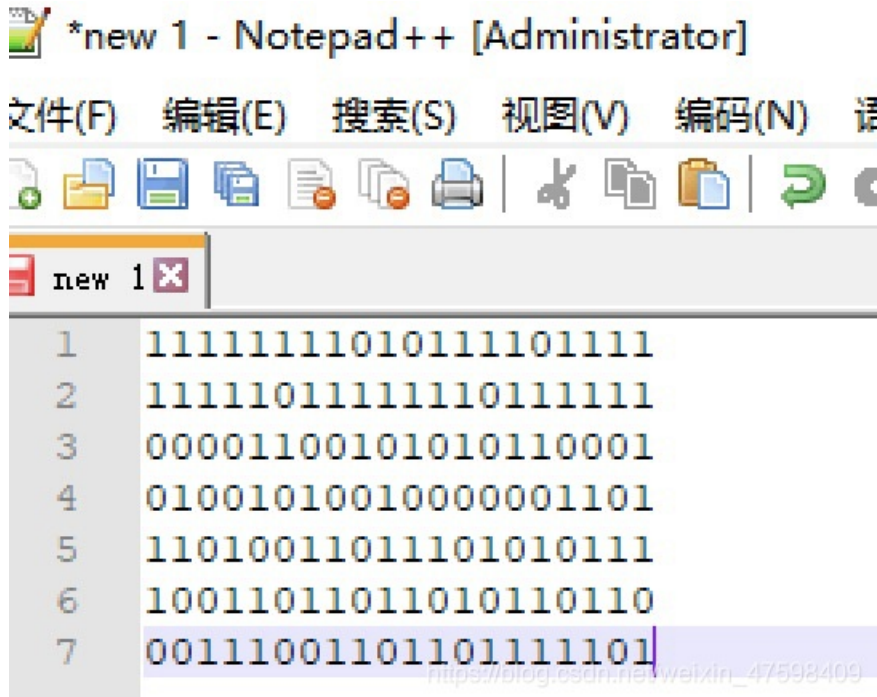
还以为这个是一个flag，结果提交不了

那么意思是让我变高 那么我们就在010或者winhex里面把它们都变高看看



然后发现每个图片最底层都是黑白相间的方块，看起来像是二进制。

使用记事本将黑白块用01记录下来。



仔细观察这几个二进制，发现第一列连起来1100110转换刚好对应的ASCII码是f

然后用大佬们的一个脚本跑一下，把每一列打竖的二进制进行转换



```
c1 = '11111111010111101111'
c2 = '11111011111110111111'
c3 = '00001100101010110001'
c4 = '01001010010000001101'
c5 = '11010011011101010111'
c6 = '10011011011010110110'
c7 = '00111001101101111101'

flag = ''

for i in range(0,20):
    c = c1[i]+c2[i]+c3[i]+c4[i]+c5[i]+c6[i]+c7[i]
    flag += chr(int(c,2))

print flag
```

```
[root@root~23:57:17~Tony]
└─┬─> ~/桌面# python aaa.py
flag{Png1n7erEs7iof}
└─┬─> [root@root~23:57:20~Tony]
└─┬─> ~/桌面#
```

flag{Png1n7erEs7iof}

**Bugku-MISC-怀疑人生**

# 怀疑人生

## 150

zip

Flag

Submit

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

下载附件是一个zip文件自己添加一下后缀名为zip并解压

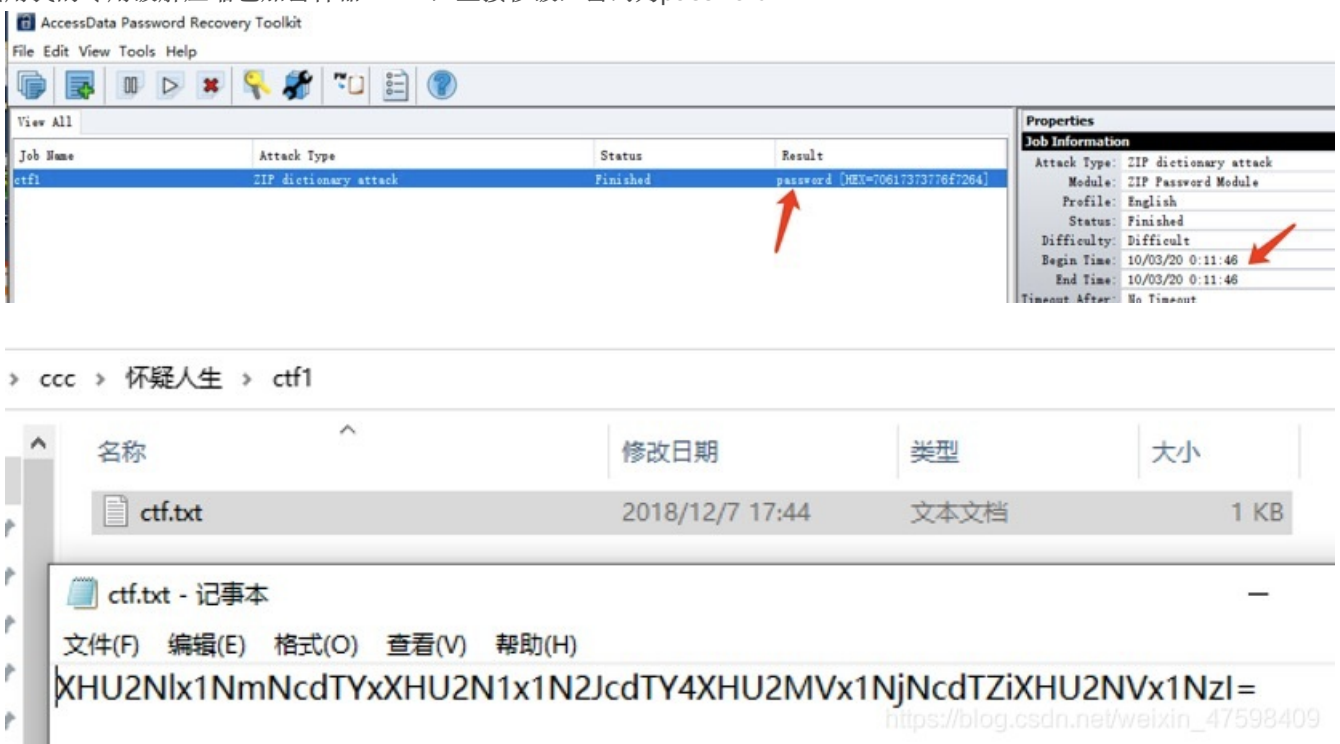
ccc &gt; 怀疑人生

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

里面有三个文件，当我想解压ctf1.zip文件时要我输入密码。。。。这。。



直接用我的专用破解压缩包加密神器PRTK，直接秒破，密码为password



这个是base64编码

填写所需检测的密码: (已输入字符数统计: 60)

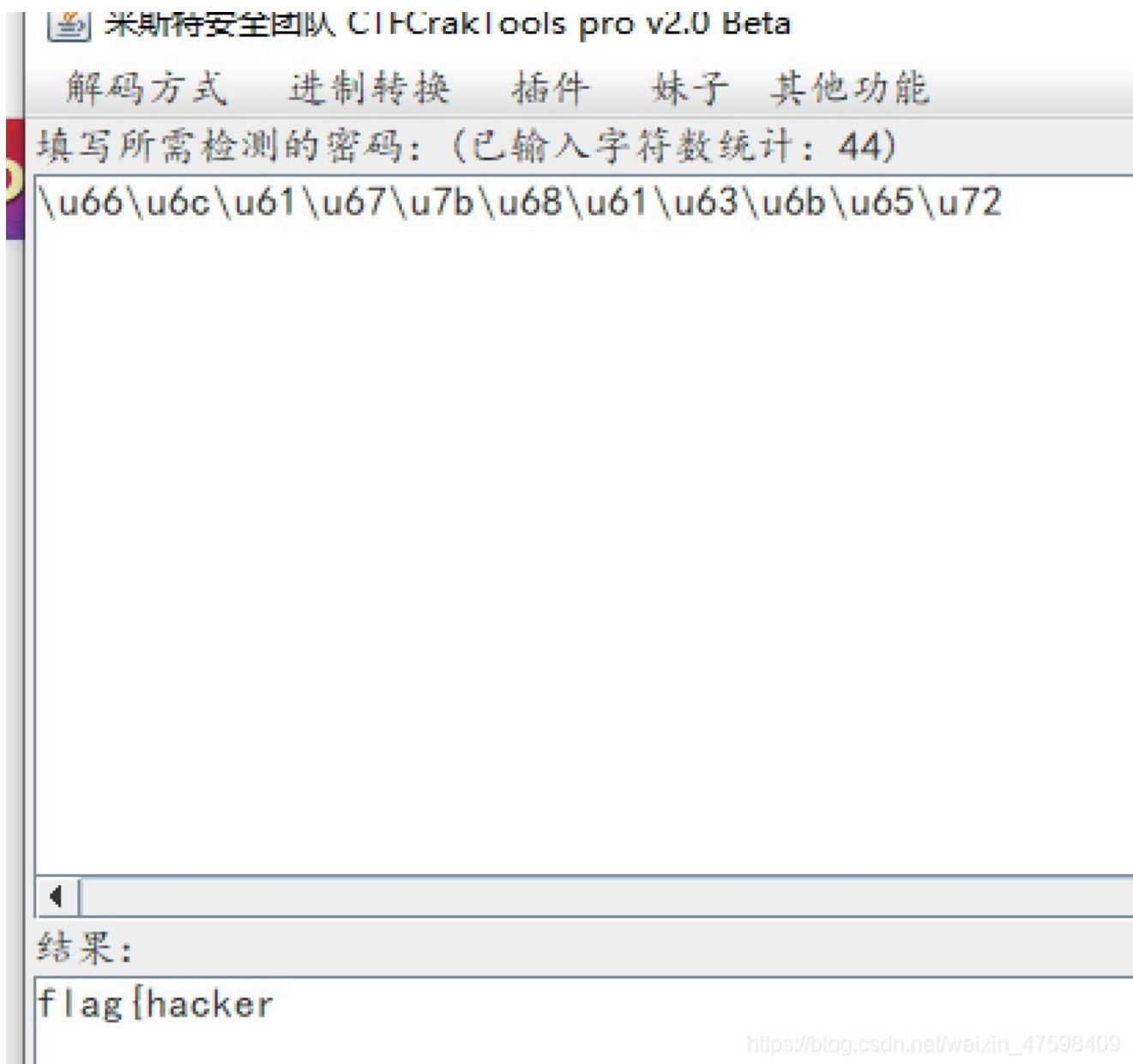
XHU2N|x1NmNcdTYxXHU2N1x1N2JcdTY4XHU2MVx1NjNcdTZiXHU2NVx1NzI=

结果:

\u66\u6c\u61\u67\u7b\u68\u61\u63\u6b\u65\u72

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

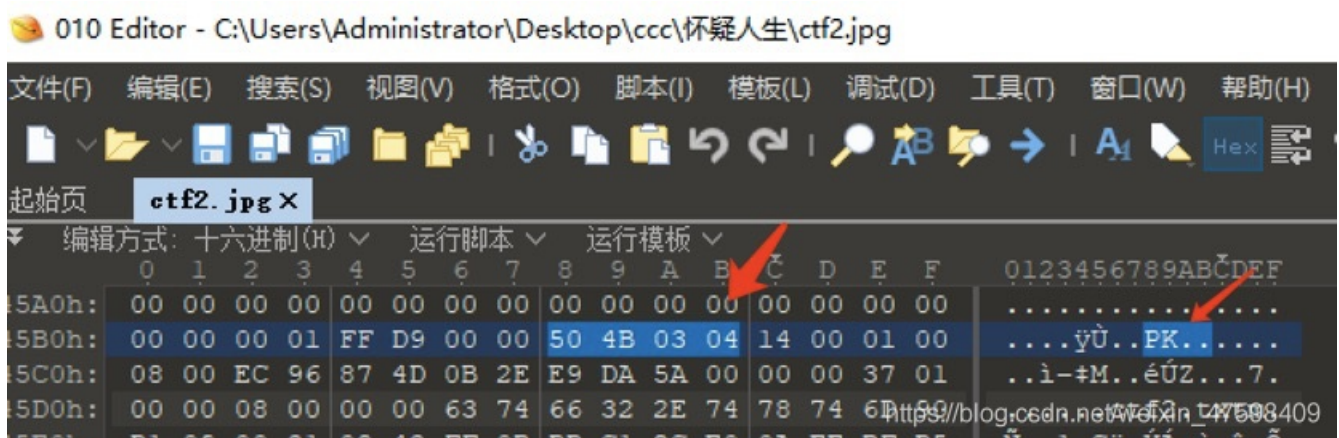
得到 Unicode编码, 直接Unicode转字符串



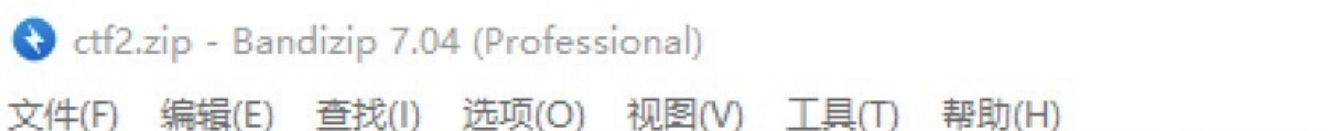
[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

得出flag的第一部分

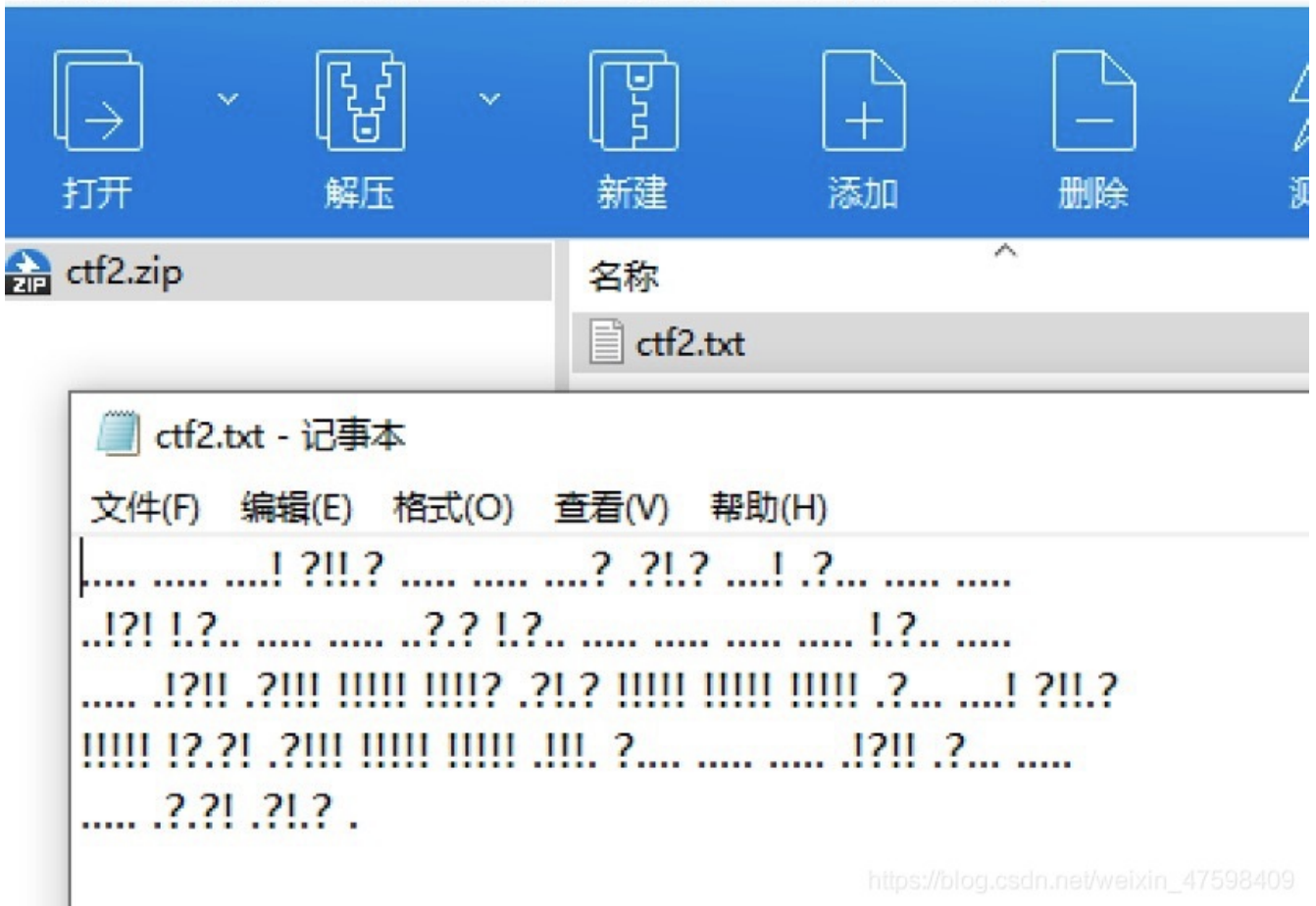
然后用010打开ctf2.jpg看一下



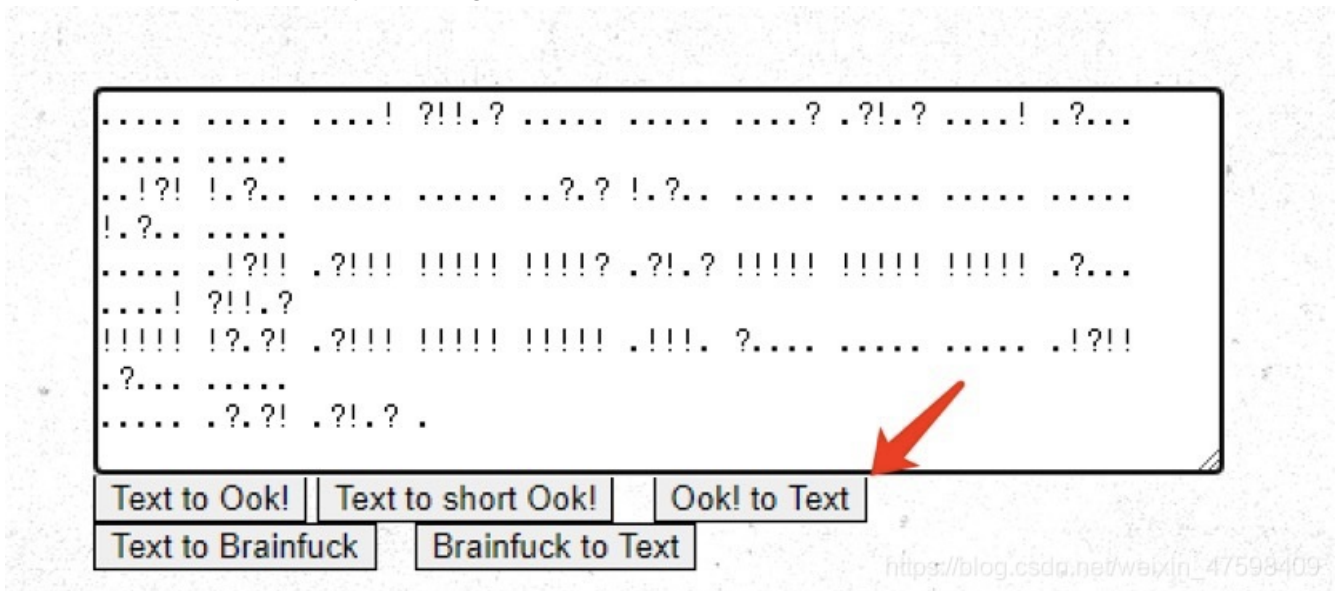
居然找到一个zip头，马上把这个文件的后缀名jpg改为zip







得出这串东西，然后查了一下资料发现是ook编码  
 直接拿去解码，传送门<https://www.splitbrain.org/services/ook>



Text to Ook!

Text to short Ook!

Ook! to Text

Text to Brainfuck

Brainfuck to Text

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

然后得到大佬的提示，这个是base58编码

传送门<https://www.jsuan.mobi/pbHzbBHbzHB6uSJx.html>

首页 / 密码学 / 密码解密



## BASE58算法解码计算器

字符串

3oD54e



计算

解码结果

misc



复制

base58

BASE58

decode

算法

解码

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

得出flag的第二部分

ctf3.jpg这个看上去是一个模糊的二维码，直接用 QR Research直接可以扫描出



得出flag的最后一部分

即flag为flag{hackermisc12580}

**Bugku-MISC-红绿灯**

Challenge

385 Solves



# 红绿灯

150

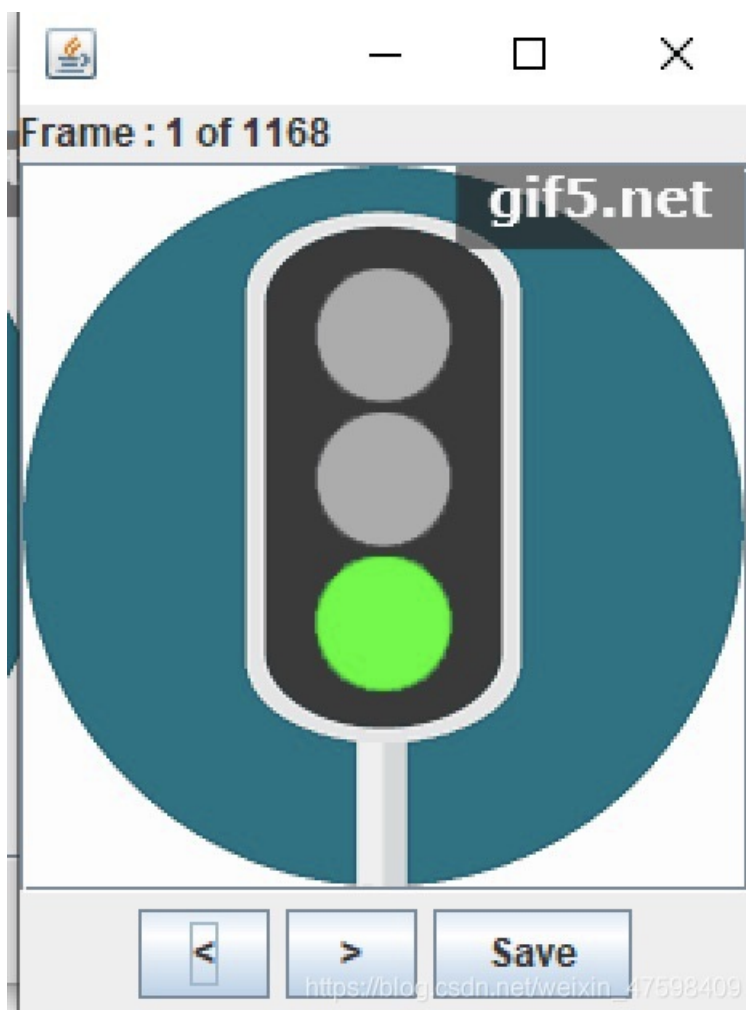
Traffic\_Light.gif

Flag

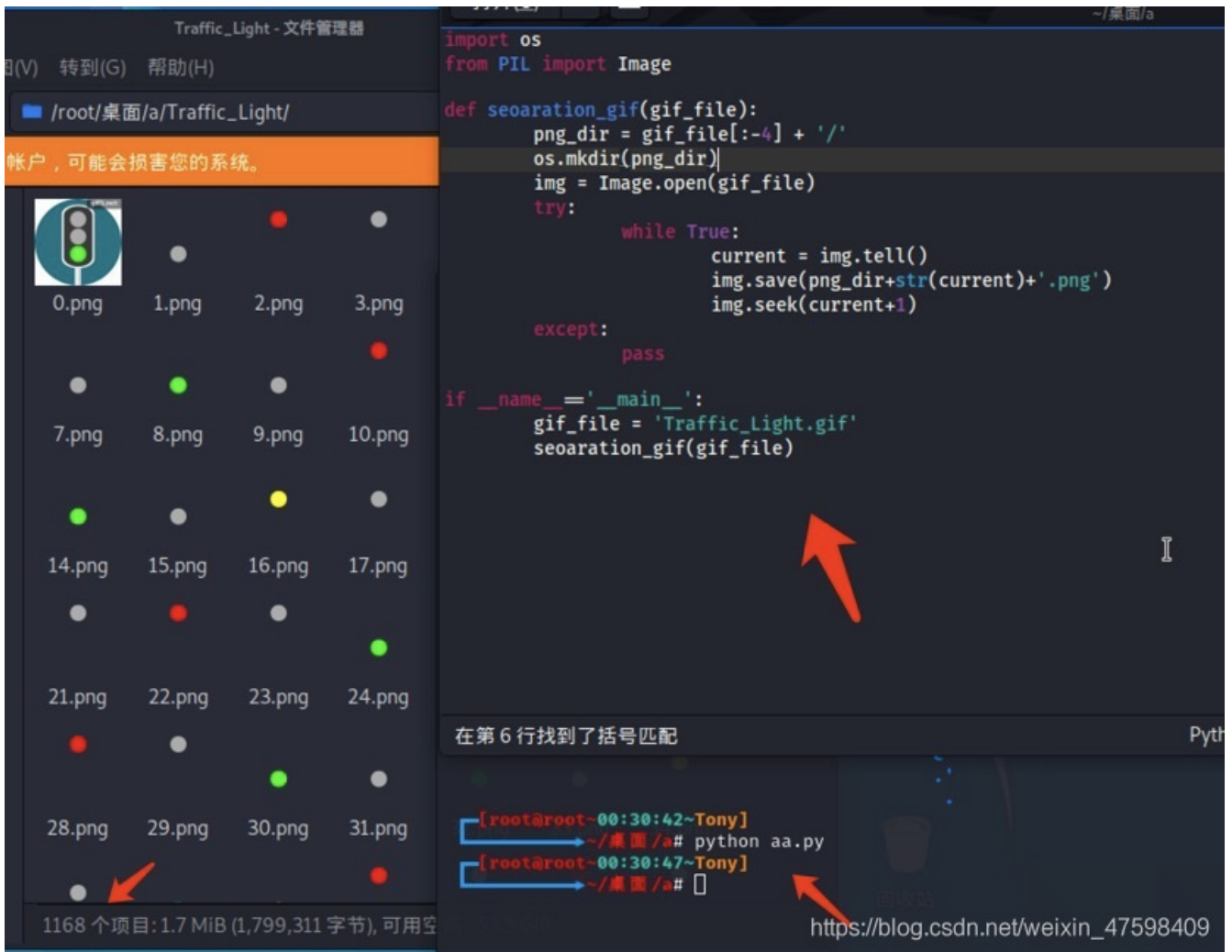
Submit

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

是一个gif的红绿灯图片，用stegsolve打开发现有1168张图片合成的一张gif图片



然后用py脚本把1168张图片分离出来，脚本如下



发现有一些图片是没有灯的，即灰色那些，红绿灯只有三种颜色，所以就把灰色(即没颜色的)删了，花了我15分钟才慢慢的删完，累死我了

删除完之后，在文件夹里面调节大小的时候，突然发现最左侧和最右侧的一列颜色都是一样的，推测最右侧的一列为空格(即黄灯)，因为ASCII码128位，最左侧的为0，得出绿色为0，红色为1，留下中间的7列。





一共455张图片，65组7位二进制数，红灯为1，绿灯为0，将图片信息记录下来

```
110011011011001100001110011111110111010000110110001100110110100111001101100111011111110000011010011110011011111
01101001110100111010001100111101110111010001100010110000110111010111111101000110000101111111010011100100110100
11001101100110011000111000111011111110011011010011001100110011111010011110011011111110111110100001100111101110
101111111100101100001110101101111101101001110010011001110111110110000111010111101001110011011000111001000110011
1111101
```

然后用py脚本把二进制转换为字符串

```
打开(O)  hah.py
~/桌面

def fun1():

    f =
'110011011011001100001110011111110111010000110110001100110110100111001101100111011111111
    b = ''
    i = 0
    j = 7
    while j <= len(f):
        a = '0' + f[i:j]
        b += chr(int(a,2))
        i = j
        j += 7
    print(b)
fun1()
```

Python 制表符宽度 : 8

```
[root@root~00:57:12~Tony]
~/桌面 # python hah.py
flag{Pl34s3_p4y_4tt3nt10n_t0_tr4ff1c_s4f3ty_wh3n_y0u_4r3_0uts1d3}
[root@root~00:57:15~Tony]
~/桌面 #
```

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

flag{Pl34s3\_p4y\_4tt3nt10n\_t0\_tr4ff1c\_s4f3ty\_wh3n\_y0u\_4r3\_0uts1d3}

## Bugku-MISC-不简单的压缩包

Challenge

294 Solves



# 不简单的压缩包

150

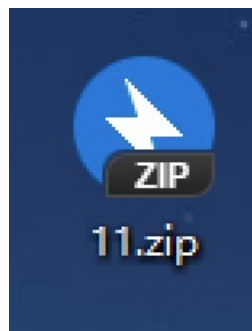
zip

Flag

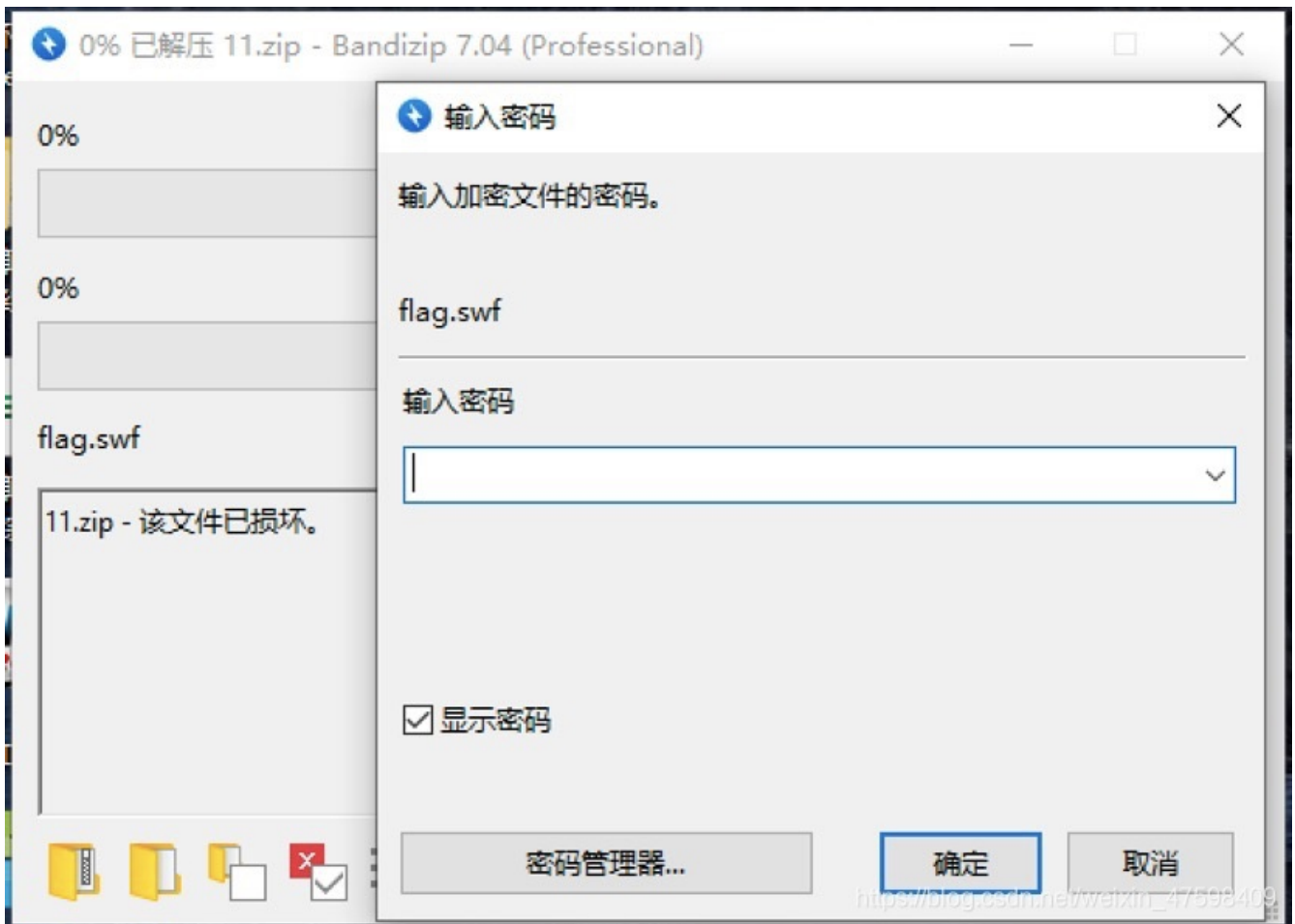
Submit

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

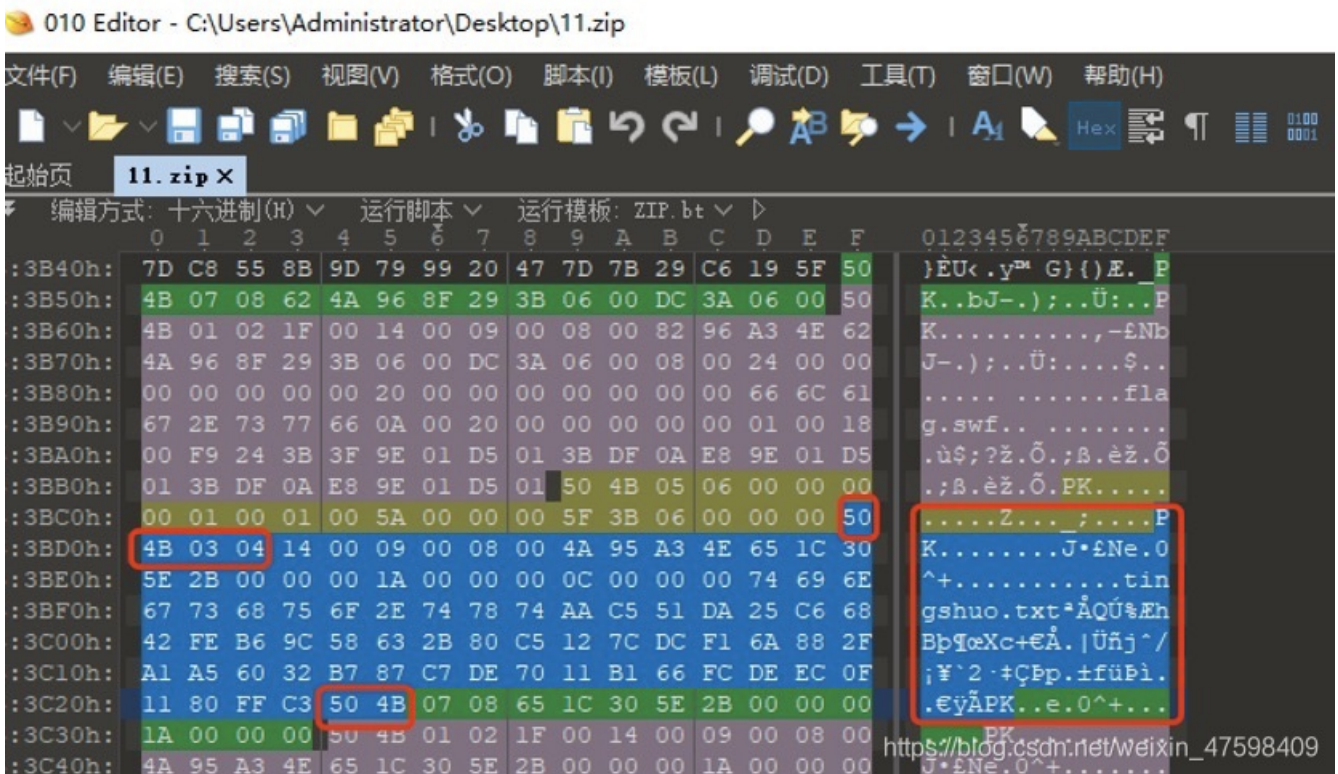
下载附件zip，修改一下文件名称



居然有密码。。。。



用010看了一下，发现这里有一个50 4B 03 04的十六进制zip文件头，而且还看到了50 4B的文件尾，应该是一个隐藏的zip文件吧



在Kali用binwalk工具查看到这个压缩包里面有一个flag.swf文件和隐藏压缩包里面有一个tingshuo.txt文本  
 直接用foremost分离，但是依然里面还是有密码







密码是50位? what?

然后根据全球知名解压缩工具bandizip给出的爆破密码的时间表

## 估计恢复时间

若档案为ZIP格式并使用Intel i7来恢复，且其密码使用**数字和小写字母...**  
(处理速度：150,000,000/秒)

- 1~7个字符的密码：不到10分钟
- 8个字符：~5小时
- 9个字符：~8天
- 10个字符：~280天
- 11个字符：~27年
- 12个字符：~1001年
- 13个字符：~36060年

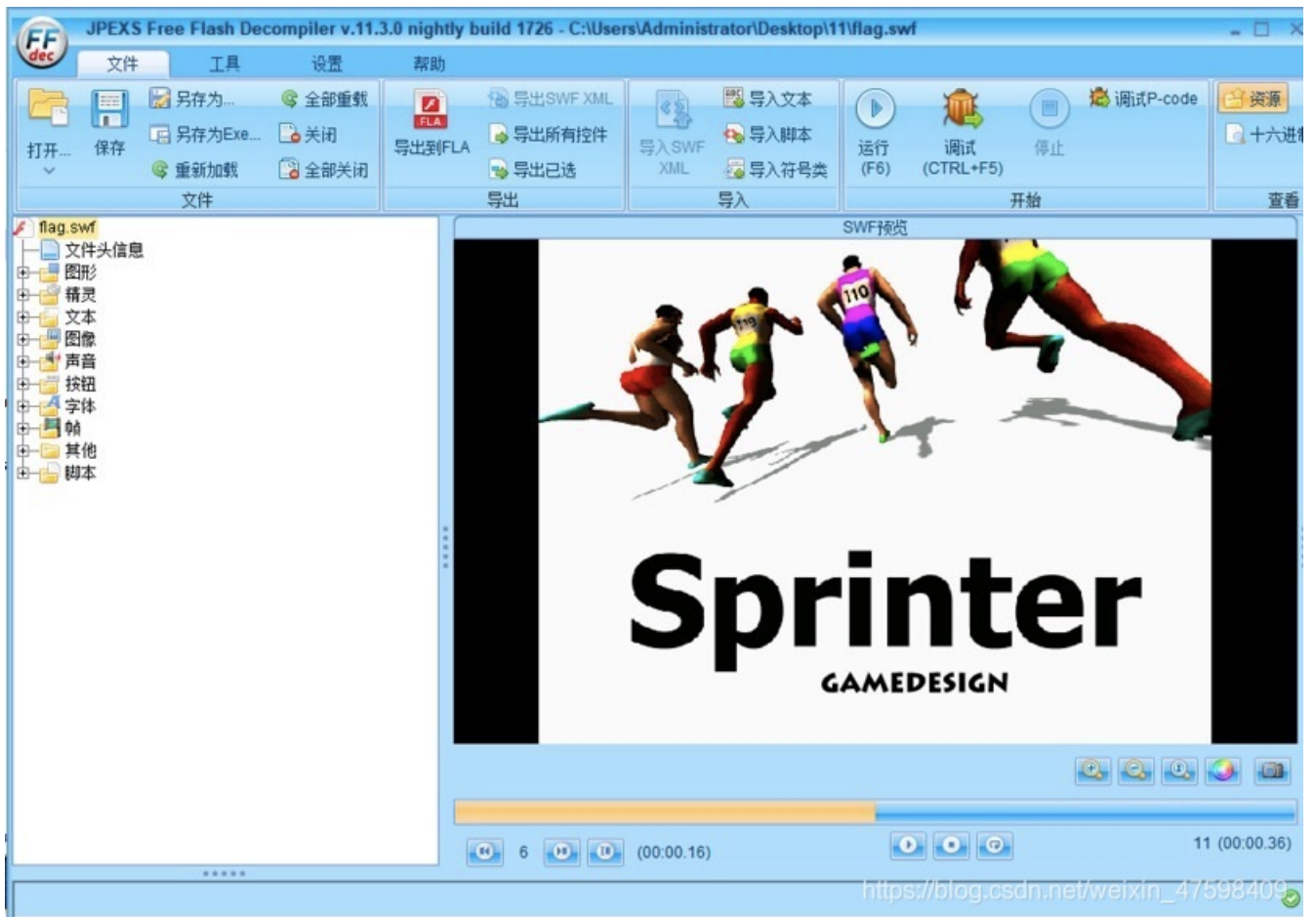
[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

密码50位，对应的应该是11.zip的，但是密码它有50位??? 暴力破解能解到地老天荒，密码会不会在属性里? 然后看了一下，什么线索都没有

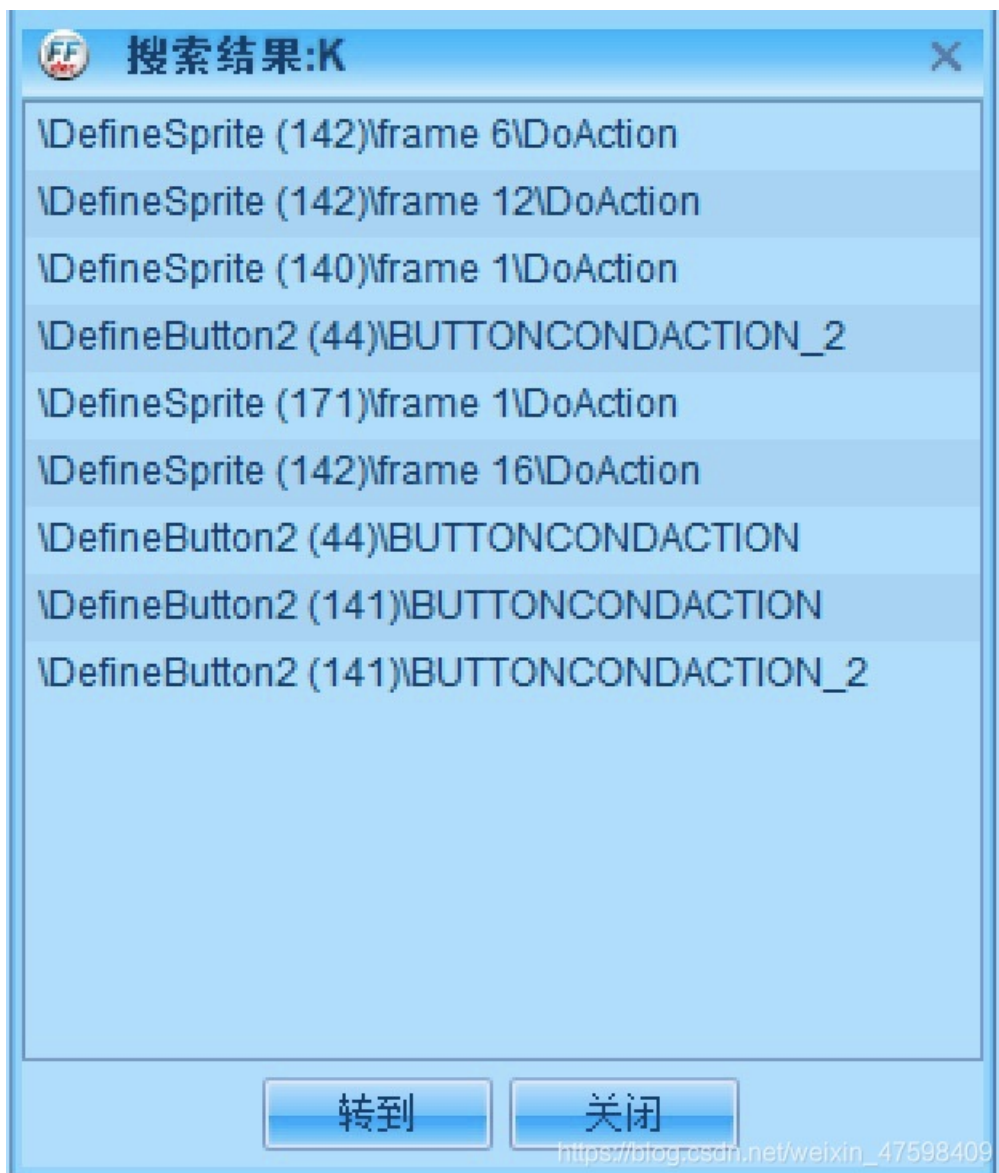
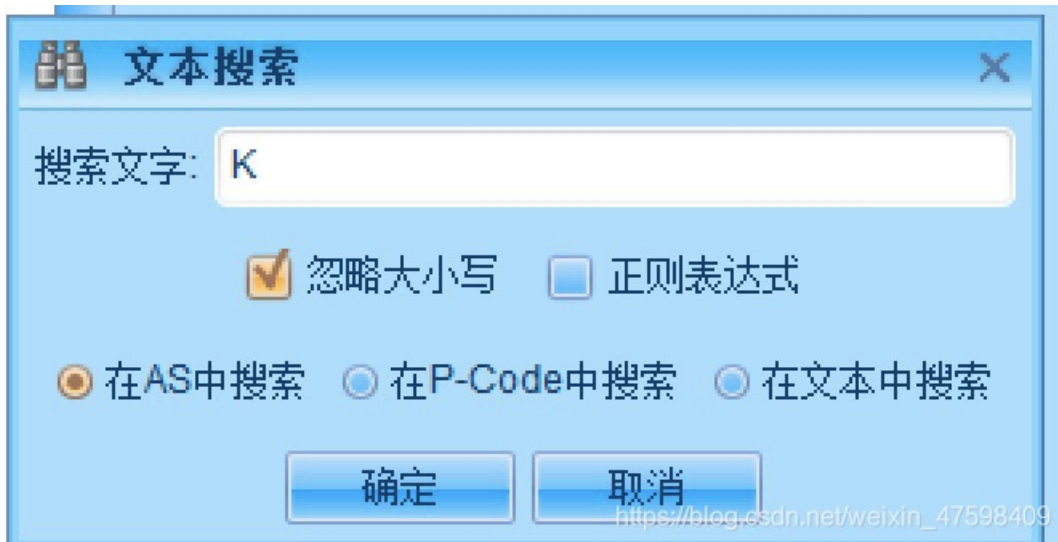








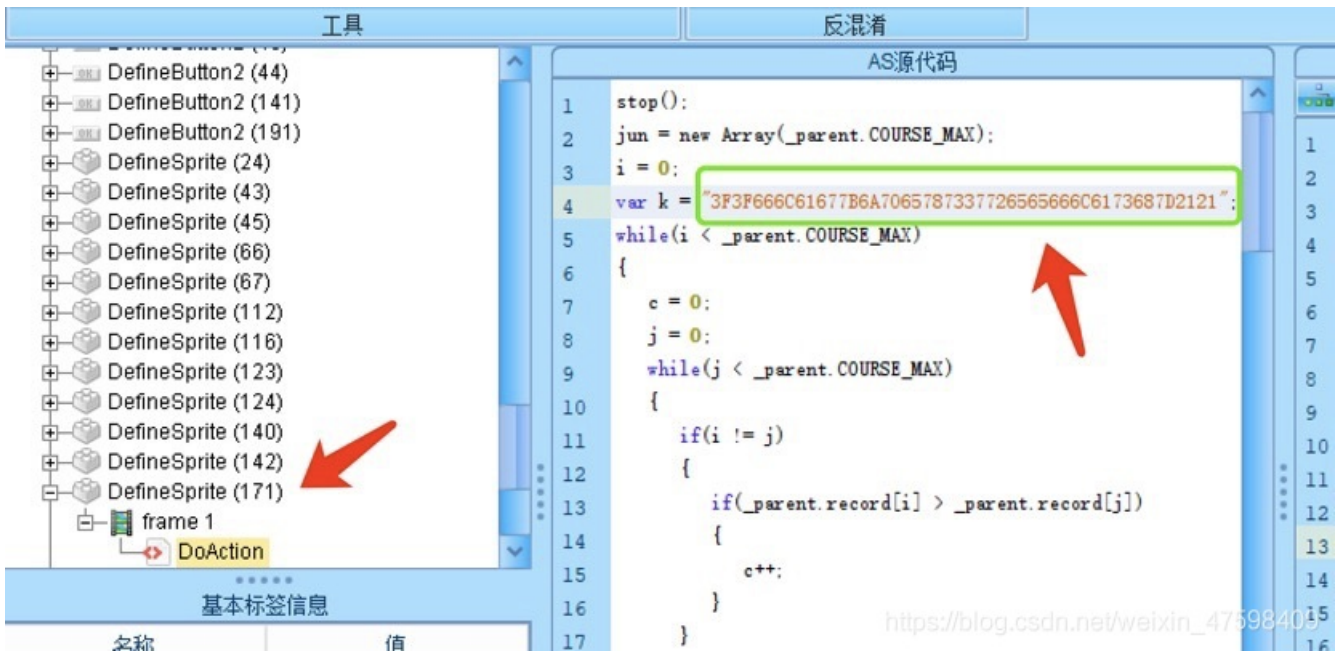
然后通过该工具的查找功能，查一下有没有flag或者Key等等关键词字



发现有这些文本里面有有K字符，然后逐个逐个的看

结果在第171个文本里面有一个K值，仔细看了一下是十六进制





然后十六进制转ASCII码得出flag



flag{jpexs7reeflash}

Bugku-MISC-一枝独秀

Challenge

257 Solves



# 一枝独秀

## 150

翻过四个栅栏即可得到flag

zip

Flag

Submit

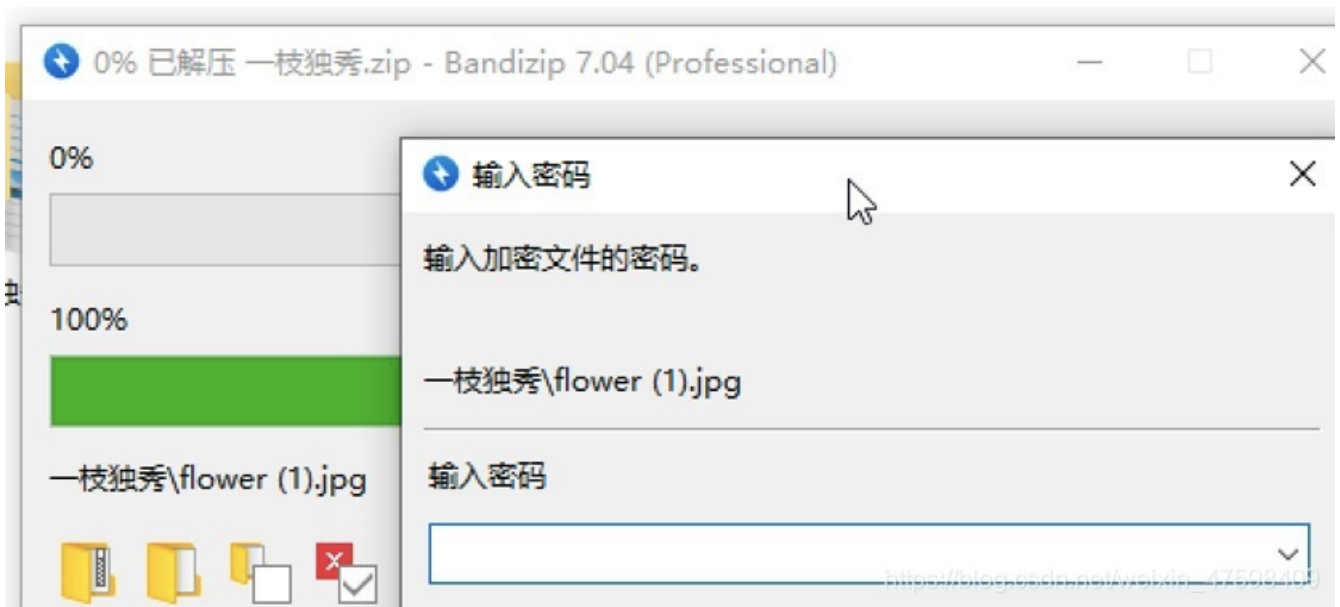
[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)



[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

打开压缩包发现有一张png图片，于是就打开png图片一看，这。。。。





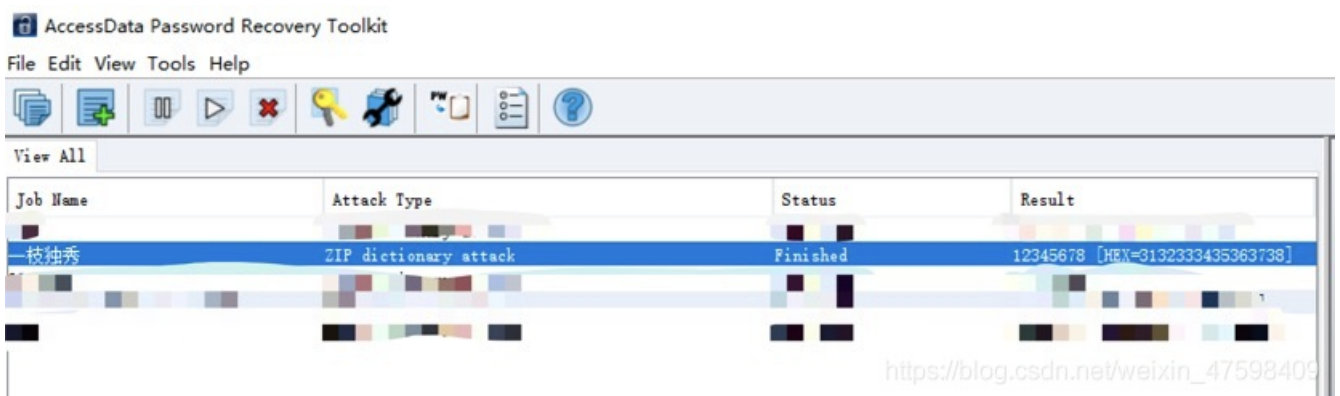
啊。。。。这。。压缩包居然有密码？

打开压缩包仔细看了一下



发现第81张图片的CRC32跟其他的不一样，有猫腻

然后直接用上我最喜欢破解压缩包密码的神器PRTK



密码秒破为12345678

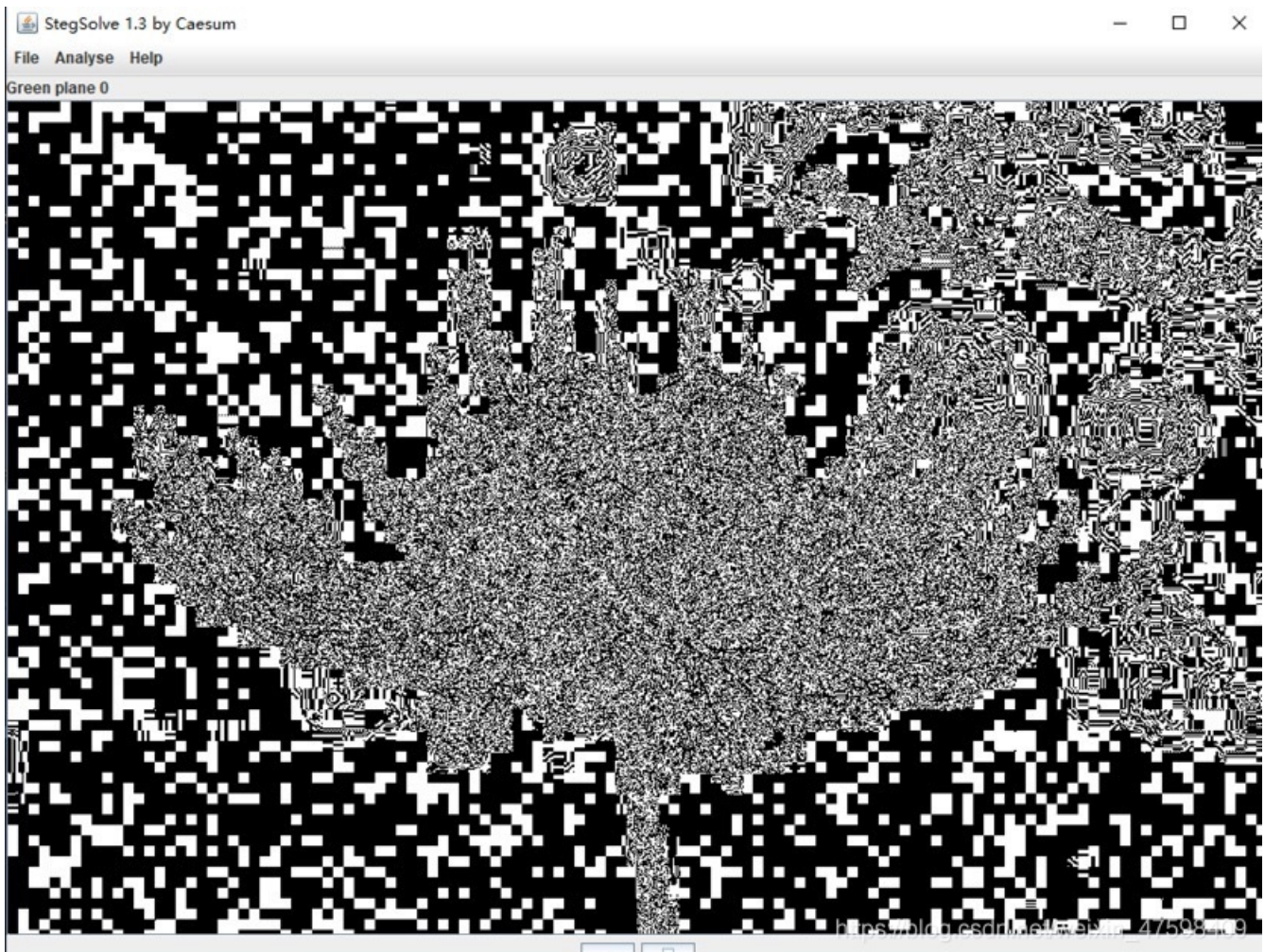
解压玩之后打开第81张图片，看上去很普通，再看一下图片的属性





Flowers 这应该是一些线索吧

然后用stegsolve看一下

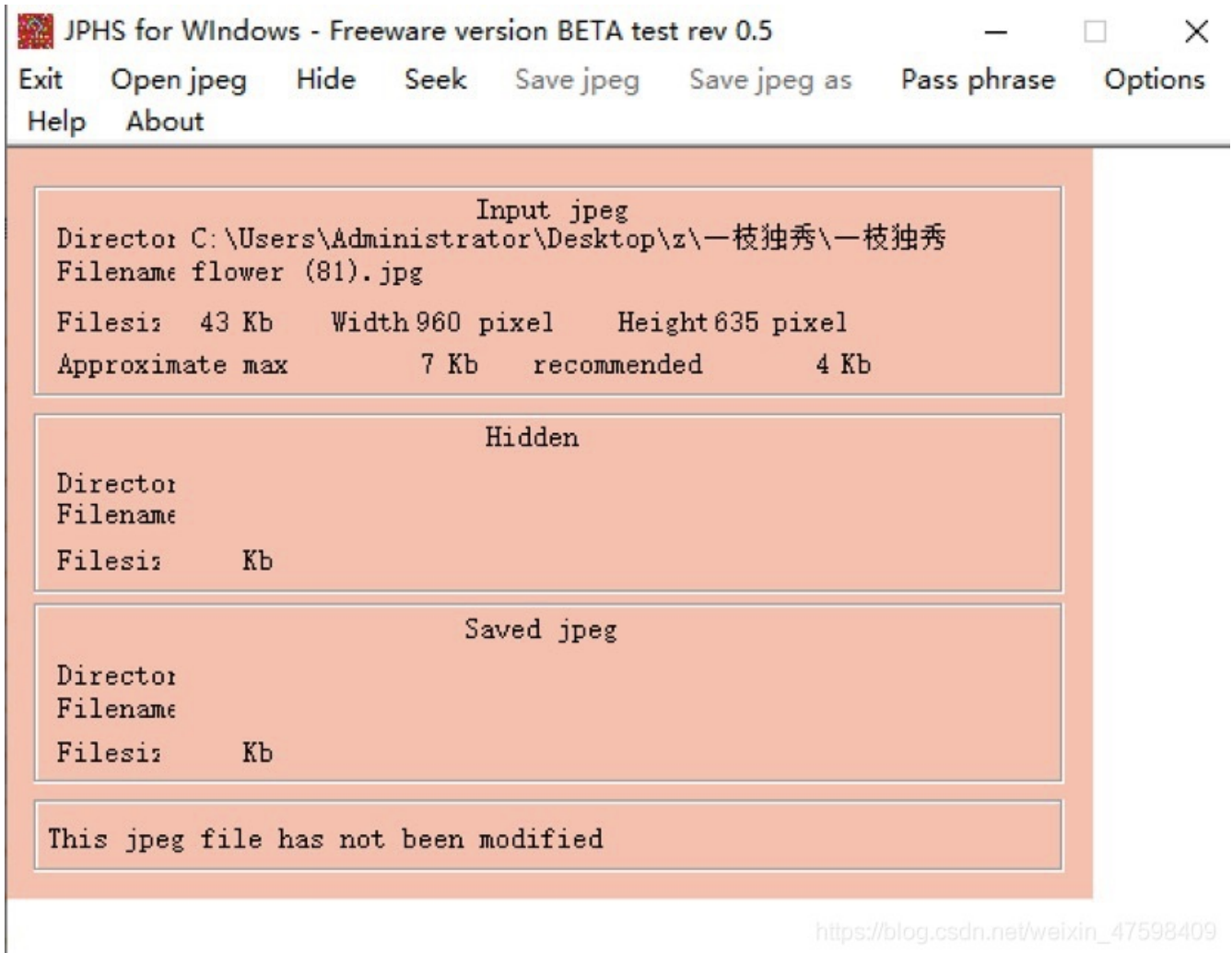


啊。。。这。。有点像二维码，但是中间一大朵花挡住了，再翻几下也没有突破新的线索

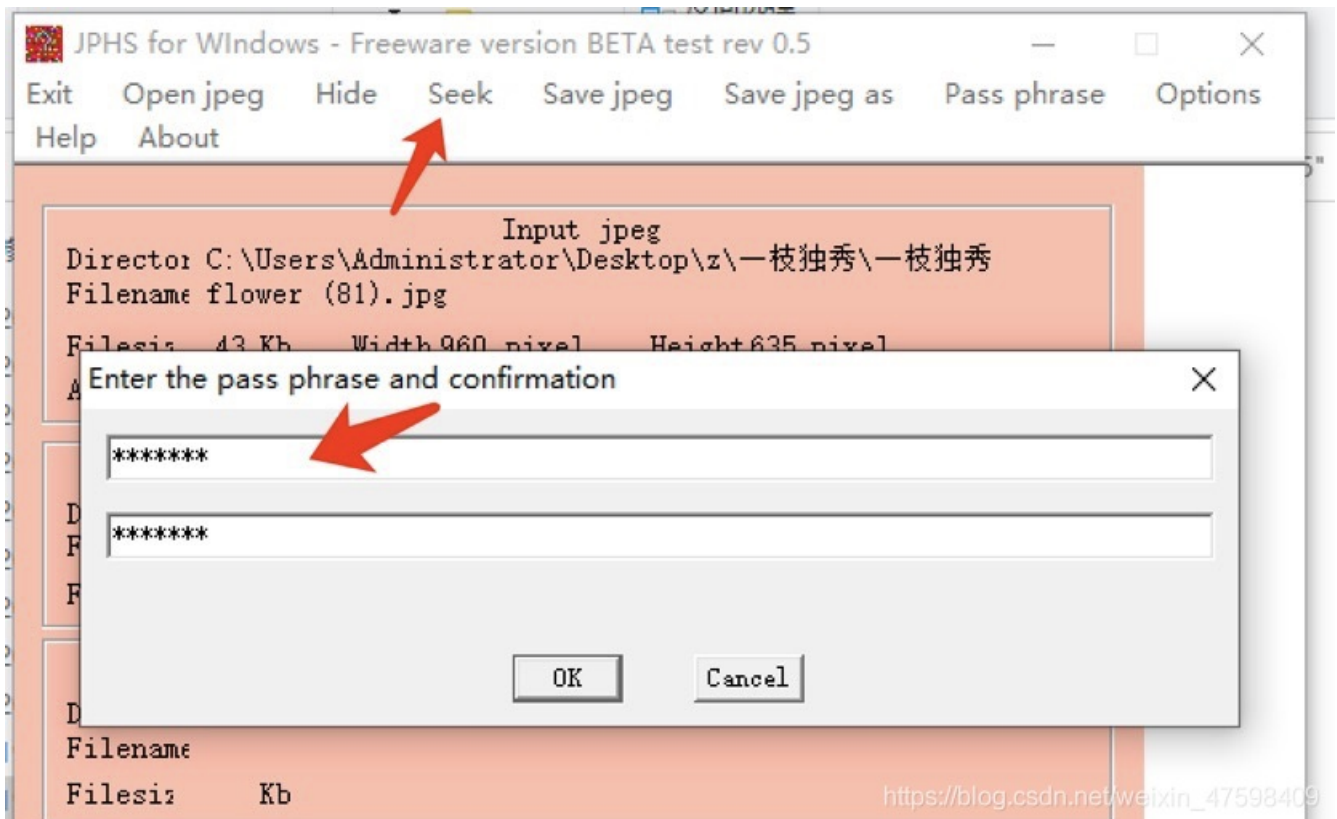
然后前段时间打强网杯遇到了一题杂项题里面也是有一张jpg图片隐写了，当时是使用了JPHS这个工具进行查看jpg的隐写文件

打开软件后导入图片





然后点击Seek



输入刚刚在图片属性里面的flowers

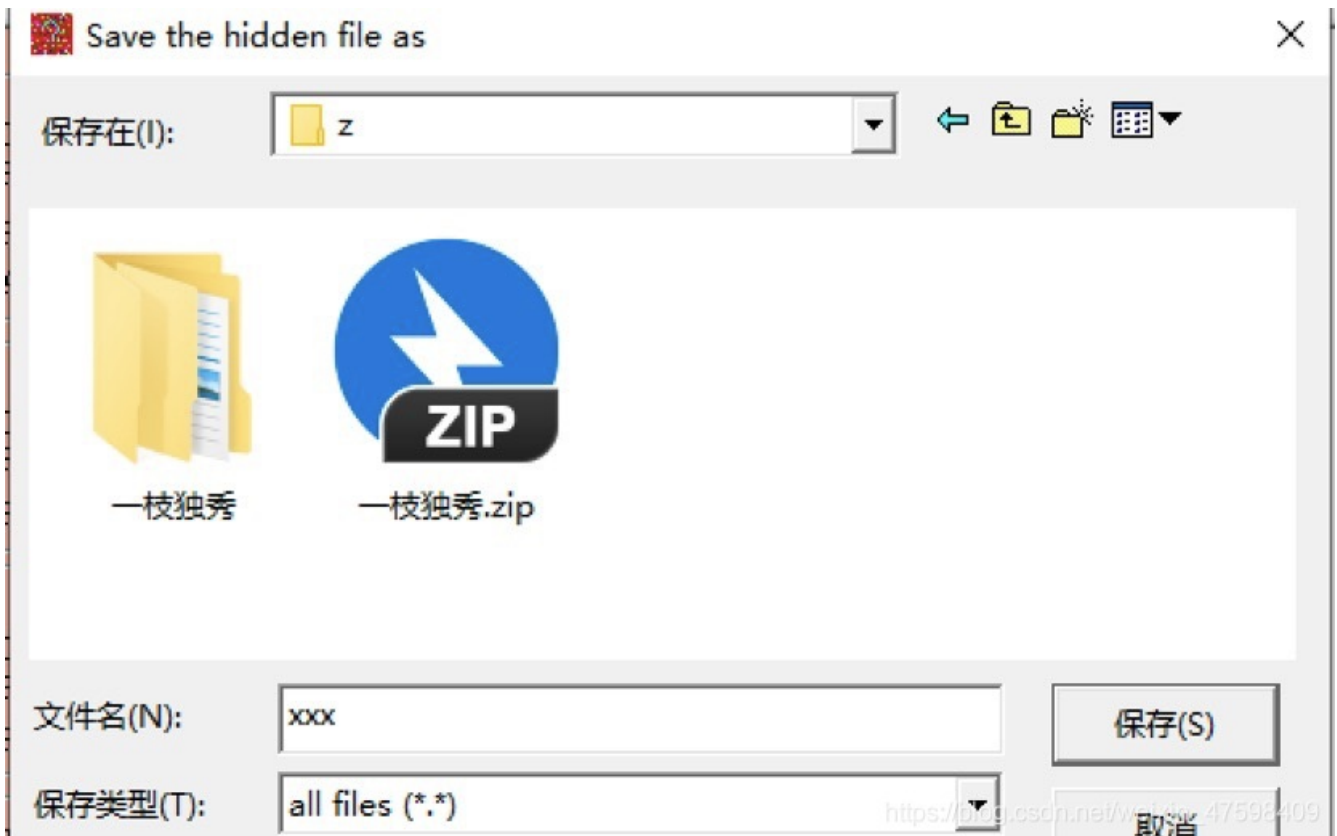
补充一点如果不知道密码的话，要使用stegbreak进行爆破，命令如下

```
C:\Users\Administrator\Desktop\stegdetect-0.4-windows>stegbreak.exe -r rules.ini -f password.txt p flower.jpg
Loaded 1 files...
flower.jpg : jphide[v5](flowers)
Processed 1 files, found 1 embeddings.
Time: 0 seconds: Cracks: 4,      Inf c/s
```

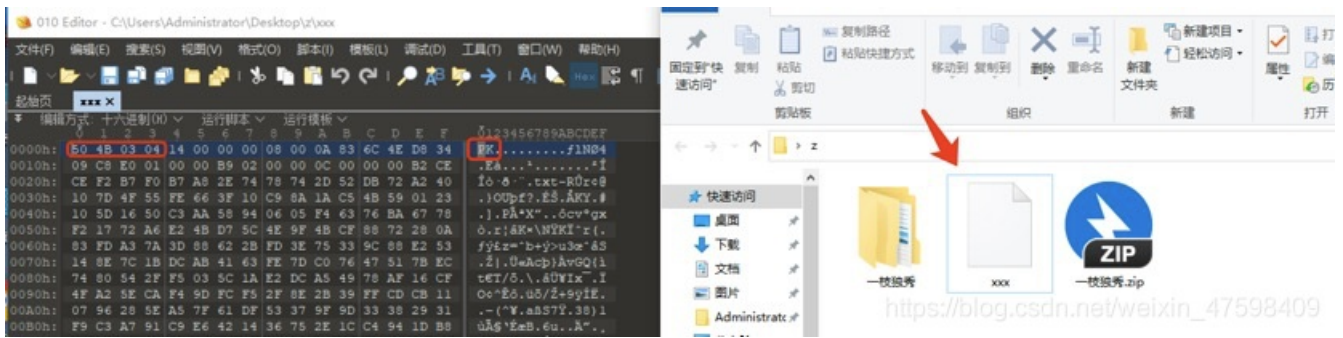
### flower (81).jpg 属性



然后另存为隐写文件



然后用010或者winhex打开隐写文件

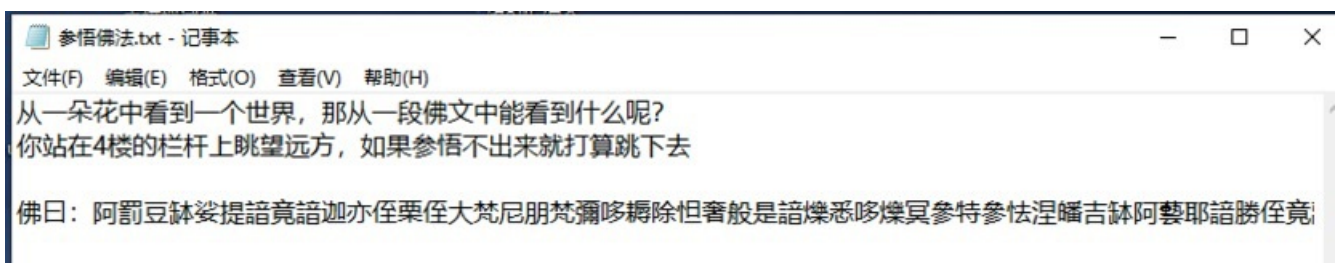


又发现了50 4B 03 04 的十六进制，这就是zip的文件头



立马修改后缀名

解压后发现一个txt文本



这。。。。佛语的意思不懂，再网上找到了这样的一个网站能进行佛语解码  
网址<http://www.keyfc.net/bbs/tools/tudoucode.aspx>

把佛曰那句话复制进去

# 与佛论禅

H-hDs10OZL3lhIZZbeRSbbbVRZnm32W2X33mGm3Txt999RdV9hx0

听佛说宇宙的真谛

参悟佛所言的真意

普度众生

心不变，万物皆不变

佛曰：阿罰豆鉢娑提諸竟諸迦亦侄栗侄大梵尼朋梵彌哆耨除怛奢般是諸燥悉哆燥冥參特參怯涅囉吉鉢阿藝耶諸勝侄竟離諸諸尼鉢曰。梵究呐耨盧他姪明漫究呐得哆藐集能冥盡滅知俱朋怯室神奢羅姪豆罰帝遠蘇明梵苦奢密侄曰鉢者特哆呼勝蘇不冥死等那阿冥悉奢薩豆涅鉢波罰。罰摩侄故罰夢鉢恐囉寫諸闍舍哆得波苦奢即罰恐冥道一哆究梵呼冥闍哆上罰南詞諸寫冥依囉者哆諦故死哆夷菩侄曰呐逝至囉佛諸耶

作者：[蓝色的风之精灵](#)；真米神表示对此工具的非非法使用概不负责。  
由 [KeyFansClub 我们的梦想](#) 提供，更多精彩不容错过！

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

点击参悟佛所言的真意，然后有一串东西

H-hDs10OZL3lhIZZbeRSbbbVRZnm32W2X33mGm3Txt999RdV9hx0

然后回想一下，题目给的提示

提示：翻过四个栅栏即可得到flag

## 栅栏密码加密解密

H-hDs10OZL3lhIZZbeRSbbbVRZnm32W2X33mGm3Ttxt999RdV9hx0|

每组字数

加密

解密

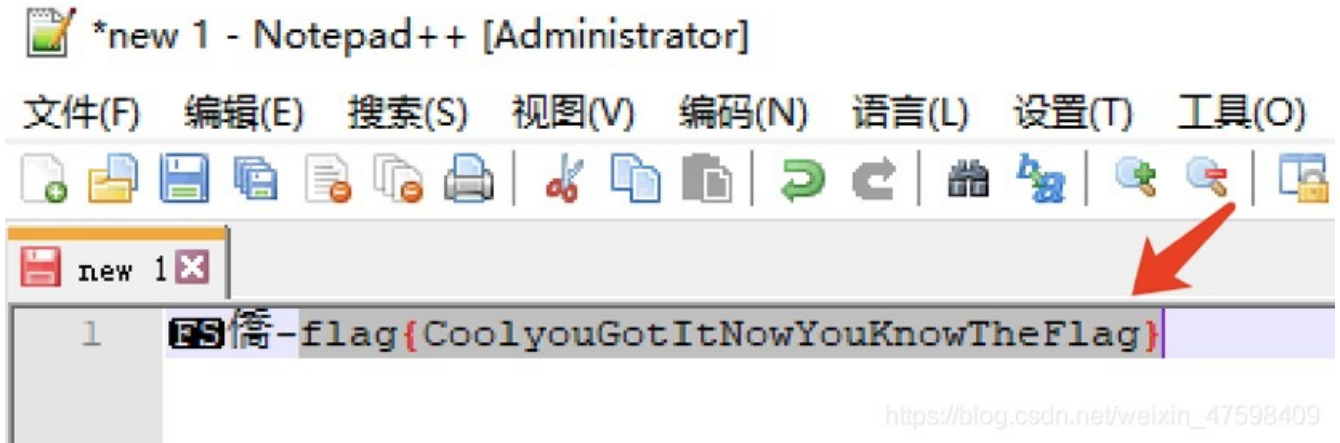
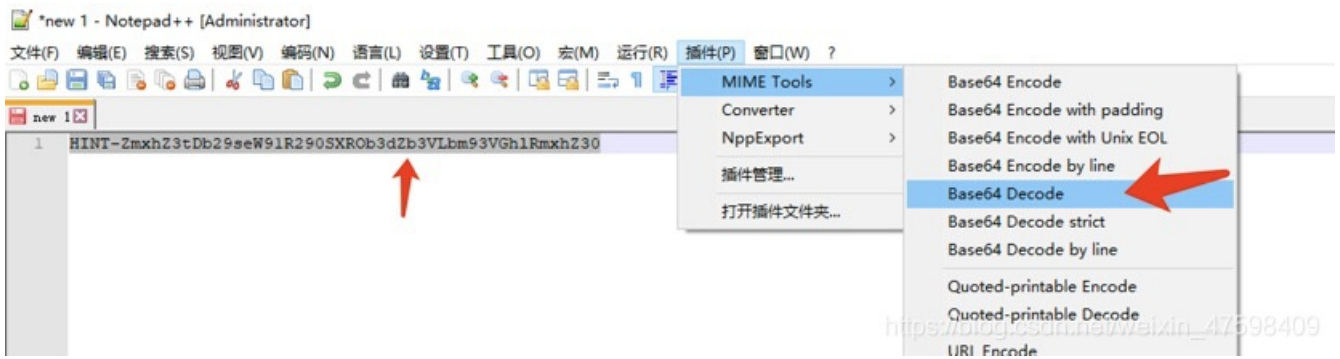
HINT-ZmxhZ3tDb29seW91R290SXROb3dZb3VLbm93VGhIRmxhZ30

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

但是解完后又来一串东西。。。感觉有点像是base64编码

然后用notepad++里面又一个插件可以解码





或者也可以来这里进行解码，传送门<https://base64.us/>

## Base64.us Base64 在线编码解码 (最好用的 Base64 在线工具)

Base64 | URLEncode | MD5 | TimeStamp

请输入要进行 Base64 编码或解码的字符

HINT-ZmxhZ3tDb29seW91R290SXROb3dZb3VLbm93VGhlRmxhZ30

编码 (Encode)

解码 (Decode)

↕ 交换

(编码快捷键: **Ctrl** + **Enter**)

Base64 编码或解码的结果:

Sflag{CoolyouGotItNowYouKnowTheFlag}

得出flag{CoolyouGotItNowYouKnowTheFlag}

Challenge

44 Solves



# 小猪佩奇 150

作者: valecalida

zip

Flag

Submit

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

这题有点捞，网上的人都没人写WP，害得我解了几个小时找方法，难受一批

下载完后有一张png图



属性就大概这样子吧，没啥好看的，只发现是32位深度

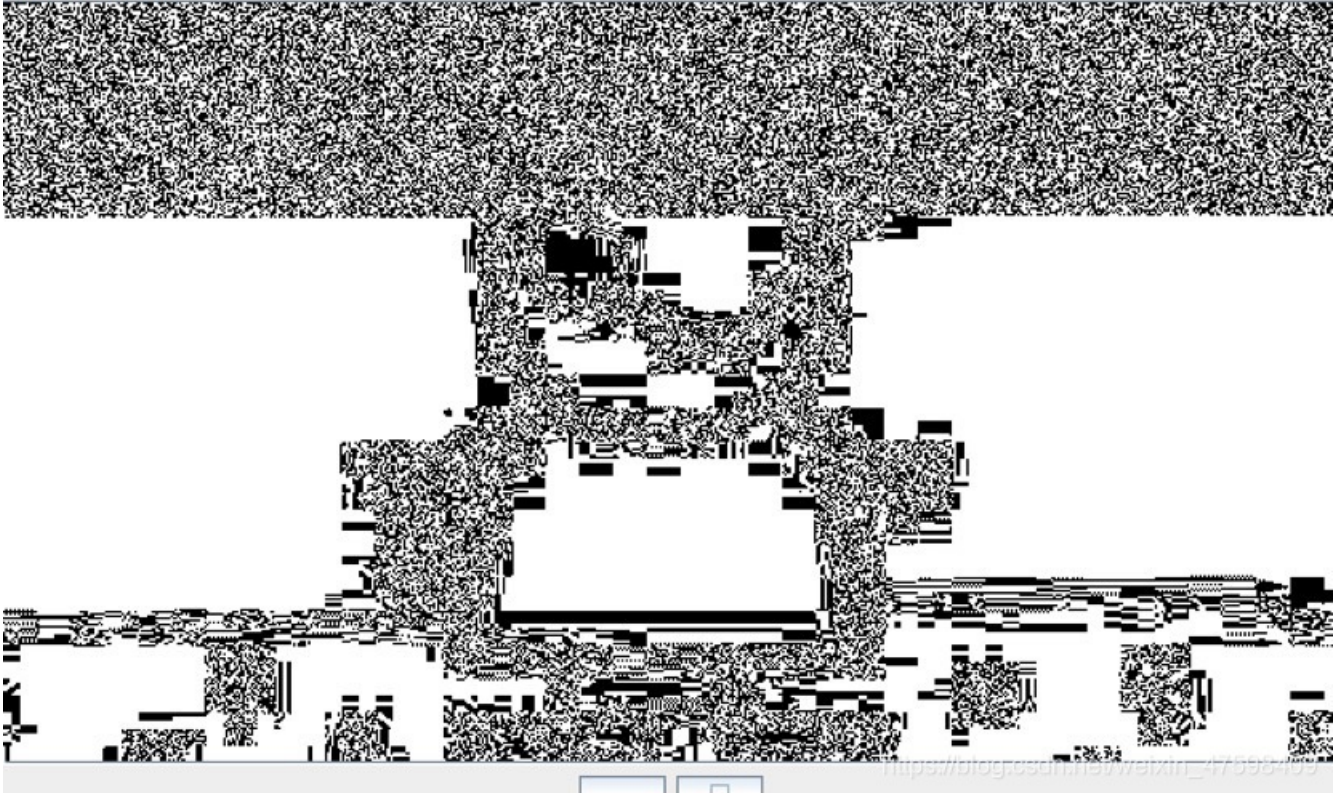


首先来讲png图，png图是种无损压缩的位图形格式，也只有在无损压缩或者压缩的图（BMP）上实现lsb隐写。如果图像是jpg图的话，就没法使lsb隐写了。原因是jpg图对像数进行了有损压缩，我们修改的信息就可能会在压缩的过程中被破坏。

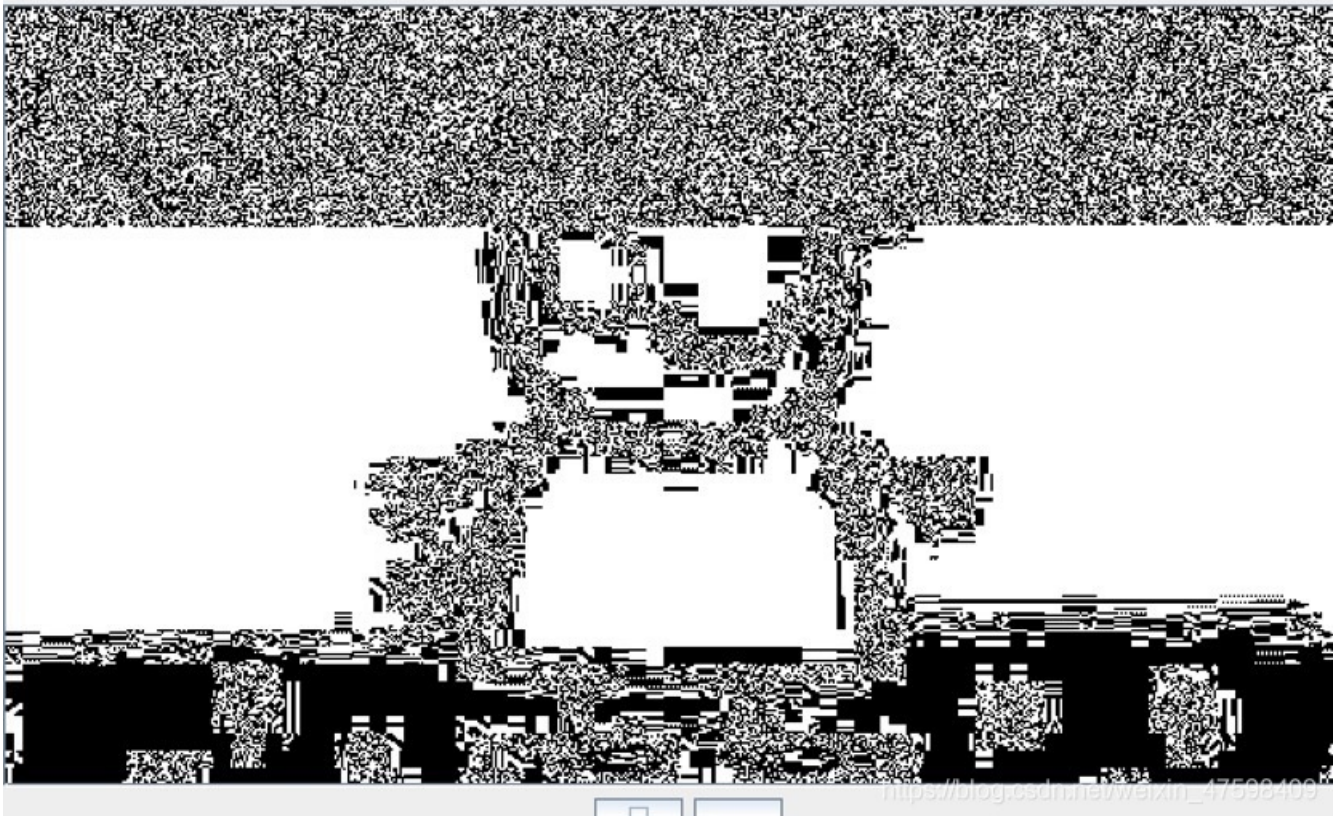
用stegsolve打开，发现Green plane 0 和Blue plane 0，两张图那个小猪里面好像有一个二维码的线条，那就证明了这个题目是lsb加密图片隐写



Green plane 0



Blue plane 0



然后，看到出题人写了一点提示

# Bugku杂项小猪佩奇思路

原创

valecalida

2019-11-18 19:45:01

2940

★ 收藏 1

版权

分类专栏:

CTF

文章标签:

CTF

Bugku

Misc

小猪佩奇

(个人希望做出题的大佬先不要把解题脚本公布出来，不然可能会造成做题乐趣的丢失，虽然我也管不着...)

字典：darkweb2017-top1000 (听说倒着用效果更好)

写个Python脚本，引入多线程，五分钟内就做出来了

如果还是做不出来的话，私聊我你的思路，正确的话我直接给你密码也ok

7位纯字母

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

去下载一个根据出题人说的字典

传送门:

<https://github.com/danielmiessler/SecLists/blob/master/Passwords/darkweb2017-top1000.txt>

但是python脚本的话我看到另外一个大佬写的WP



## bugku-小猪佩奇 wp

ctf Pythonic misc 发布于 3月15日

一道带有LSB加密的图片隐写题，难度在于要跑密码，搜wp时看到出题人的提示：

[https://blog.csdn.net/valecal...](https://blog.csdn.net/valecalida)

字典：darkweb2017-top10000

密码：7位纯字母

写个python脚本跑出来就好了，我跑单线程时也挺快的（有提示可以少跑好多密码）

github上的脚本：<https://github.com/livz/cloac...>

需要的库自己pip一下就行

运行时用python2

代码：

```
#coding:utf-8
import threading
from PIL import Image
from lsb import extract
```



```
import re
with open('darkweb2017-top10000.txt','r+') as f:
    f=f.readlines()
filename='flag.png'
```

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

需要的库自己pip一下就行

运行时用python2

代码:

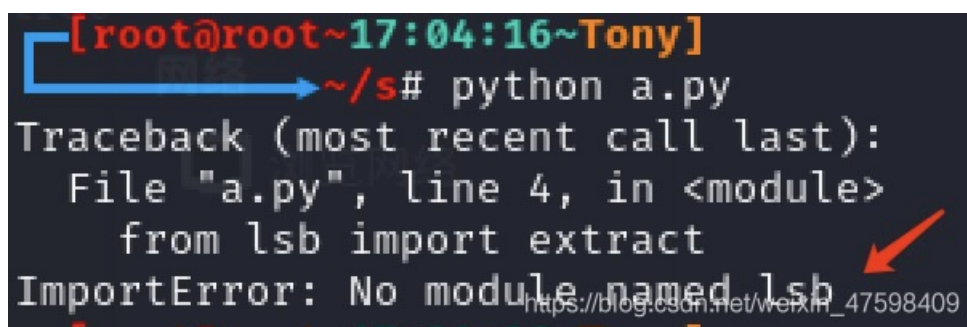
```
#coding:utf-8
import threading
from PIL import Image
from lsb import extract
import re
with open('darkweb2017-top10000.txt','r+') as f:
    f=f.readlines()
filename='flag.png'

def to_decode():
    for i in f:
        i=i.replace('\n','')
        if len(i)==7 and re.search('[0-9!?!]',i)==None:
            out_file=i+'.txt'
            extract(filename,out_file,i)
to_decode()
```

将解出的N多个文件在linux下用file命令识别一下可以得到一个PNG，改下后缀名打开就是二维码，扫描得flag

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

这位大佬提供的python用出题人说的字典进行爆破lsb隐写脚本，但是可能我太菜了，悟性不够高，运行了好几次这个脚本都是出现这种情况



```
[root@root ~]# python a.py
Traceback (most recent call last):
  File "a.py", line 4, in <module>
    from lsb import extract
ImportError: No module named lsb
```

The screenshot shows a terminal window with a blue arrow pointing to the command `python a.py` and a red arrow pointing to the error message `ImportError: No module named lsb`.

一直说我没有lsb库，我就用pip来安装，结果pip和pip3都是试过，依然不行，就这个问题花了我几个小时查资料还是没能解决，我就没使用这位大佬的脚本了。

```
[root@root~00:20:46~Tony]
└─# pip install lsb
/usr/share/python-wheels/pkg_resources-0.0.0-py3-none-any.whl/pkg_resources/py2_warn.py:21: UserWarning: S
etuptools will stop working on Python 2
*****
You are running Setuptools on Python 2, which is no longer
supported and
>>> SETUPTOOLS WILL STOP WORKING <<<
in a subsequent release (no sooner than 2020-04-20).
Please ensure you are installing
Setuptools using pip 9.x or later or pin to `setuptools<45`
in your environment.
If you have done those things and are still encountering
this message, please follow up at
https://bit.ly/setuptools-py2-warning.
*****
WARNING: pip is being invoked by an old script wrapper. This will fail in a future version of pip.
Please see https://github.com/pypa/pip/issues/5599 for advice on fixing the underlying issue.
To avoid this problem you can invoke Python with '-m pip' instead of running pip directly.
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade your Python as Py
thon 2.7 is no longer maintained. A future version of pip will drop support for Python 2.7. More details a
bout Python 2 support in pip, can be found at https://pip.pypa.io/en/latest/development/release-process/#p
ython-2-support
ERROR: Could not find a version that satisfies the requirement lsb (from versions: none)
ERROR: No matching distribution found for lsb
https://blog.csdn.net/weixin_47598409
```

```
[root@root~00:20:57~Tony]
└─# pip3 install lsb
ERROR: Could not find a version that satisfies the requirement lsb (from versions: none)
ERROR: No matching distribution found for lsb
```

然后我就用github上提供的lsb脚本

<https://github.com/livz/cloacked-pixel>

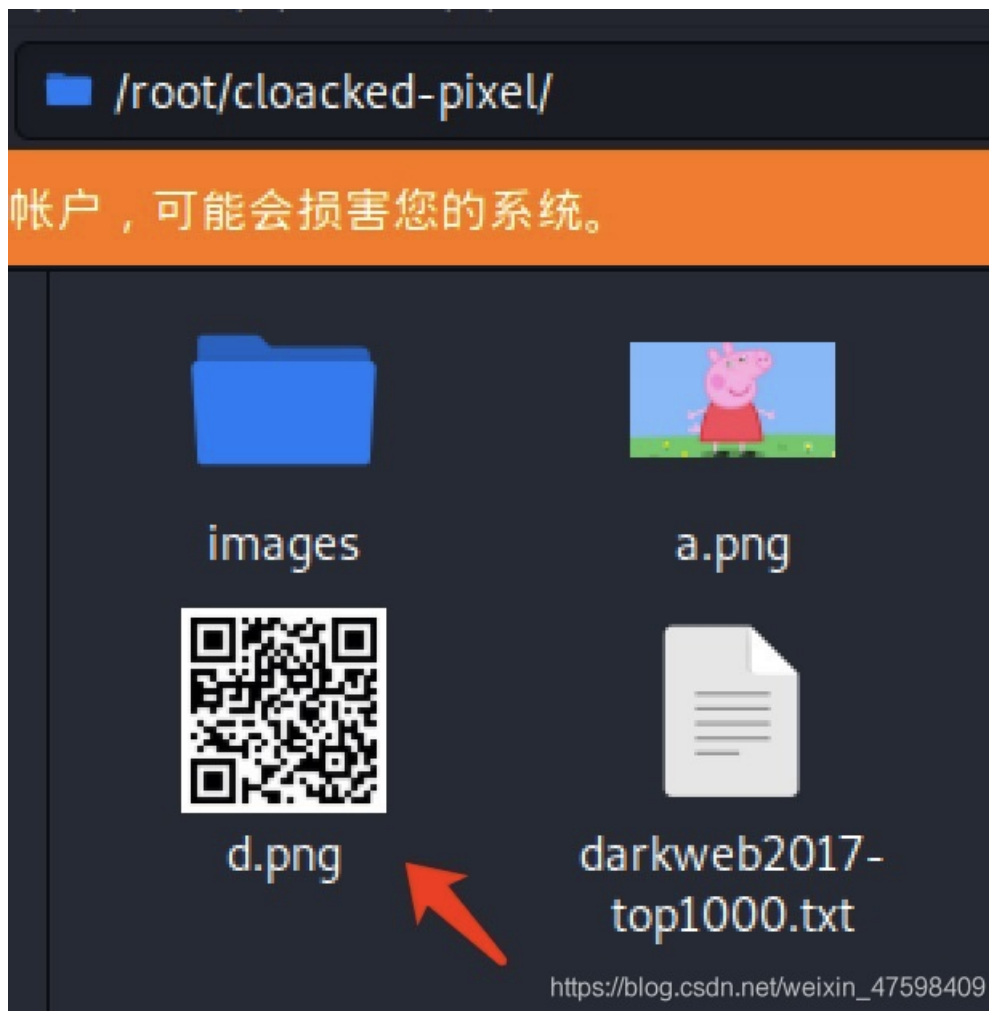
根据脚本的使用教程，和出题人的提示说密码是纯7为字母，并且字典建议倒着用效果会更好，然后经过一番的排除操作

```
[root@root~22:43:20~Tony]
└─~/cloacked-pixel# python lsb.py extract a.png d.png raiders
[+] Image size: 630x359 pixels.
[+] Written extracted data to d.png.
```

结果发现了正确密码就是raiders，中文意思是掠夺者，那应该代表就是获得flag的行为吧



然后就爆破出一张二维码



通过扫描二维码得出flag



flag{37d9704c-9752-434c-8891-ee15e1800490}

[Bugku-MISC-好多压缩包](#)

Challenge

562 Solves



# 好多压缩包

## 200

123.zip

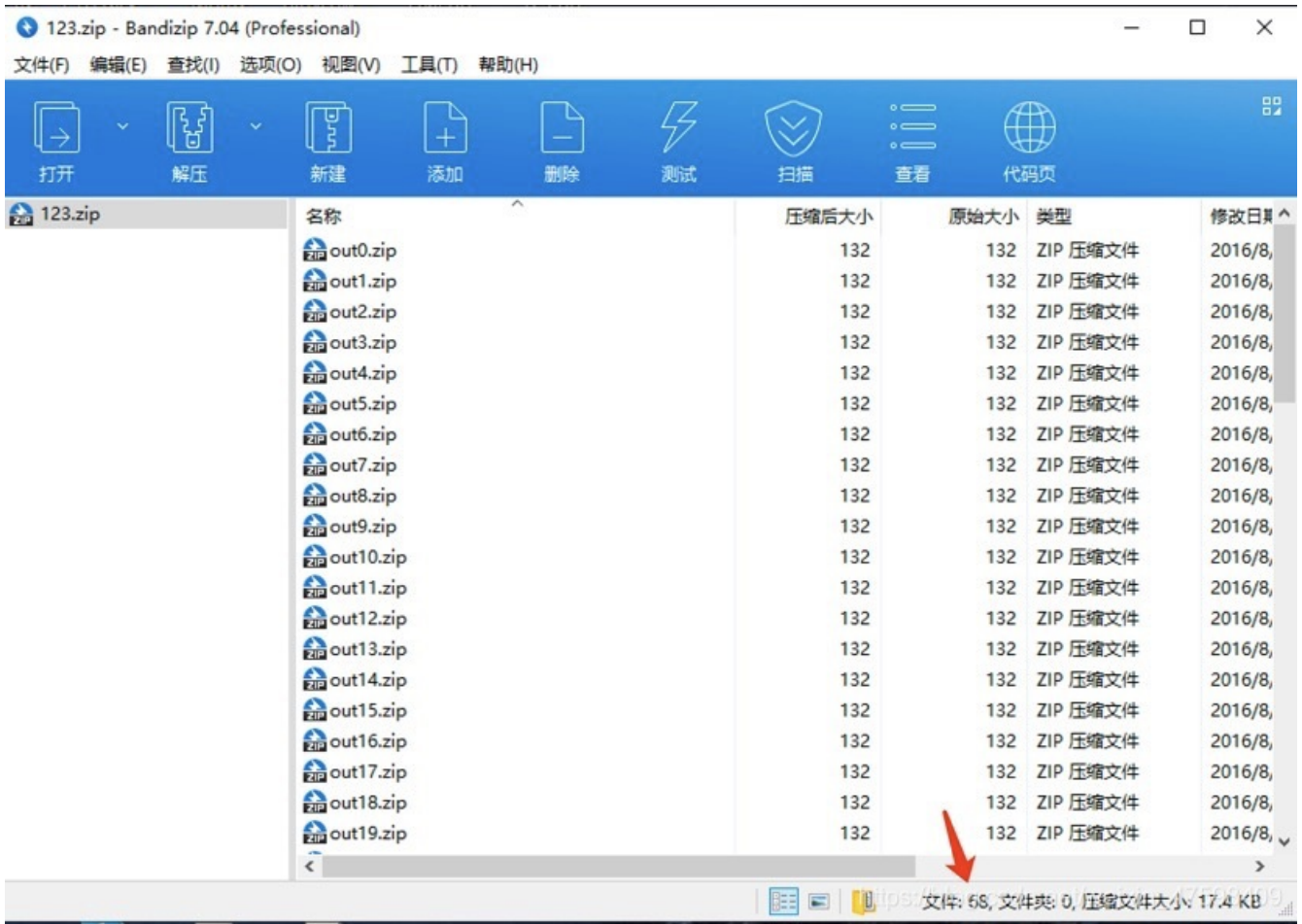
Flag

Submit

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

打开压缩包里面有68个加密的压缩包





然后随便点开一个压缩包看了一下发现被加密的压缩包里面有一个大小为4字节的文本



科普一下CRC32碰撞

### 注意

一般情况，碰撞的字节数不会超过5（通常是3或者4字节），否则要碰撞很久，碰撞时间太久的话这个题就没什么意思了。一般看见压缩包里有很多文件，每个文件大小都小于5字节，才会用crc32碰撞。

然后可以用CRC32碰撞脚本进行爆破

由于前段时间打强网杯的时候遇到了一题MISC题目，里面也需要CRC32碰撞，但是那个脚本只能单独的一个一个压缩包进行碰撞，虽然速度是秒破，但是一个一个的搞太麻烦了

```
[root@root~11:09:47~Tony]
~/桌面/zip-crc-cracker-master# python3 crack.py out0.zip
reading zip files ...
file found: out0.zip / data.txt: crc = 0x75f90d3a, size = 4
compiling ...
searching ...
crc found: 0x75f90d3a: "z5Bz"
done
out0.zip / data.txt : 'z5Bz'
```

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

然后在网上找到了大佬们的WP，他们写的CRC32碰撞脚本是直接吧68个压缩包连在一起进行碰撞，然后把碰撞出来的内容保存在out.txt文本里面

这里有一点要注意，如果用的是python2来运行的话就必须加上0xffffffff，如果是用python3就不用加要特别注意

```
if (binascii.crc32(str(i)) & 0xffffffff) == crc:
```

在 Python 2.x 的版本中，binascii.crc32 所计算出来的 CRC 值域为 $[-2^{31}, 2^{31}-1]$ 之间的有符号整数，为了要与一般CRC结果作对比，需要将其转为无符号整数，所以加上& 0xffffffff来进行转换。如果是 Python 3.x 的版本，其计算结果为 $[0, 2^{32}-1]$ 间的无符号整数，因此不需额外加上& 0xffffffff。

代码如下：

```
import zipfile
import string
import binascii

def CrackCrc(crc):
    for i in dic:
        for j in dic:
            for p in dic:
                for q in dic:
                    s = i + j + p + q
                    if crc == (binascii.crc32(s) & 0xffffffff):
                        #print s
                        f.write(s)
                        return

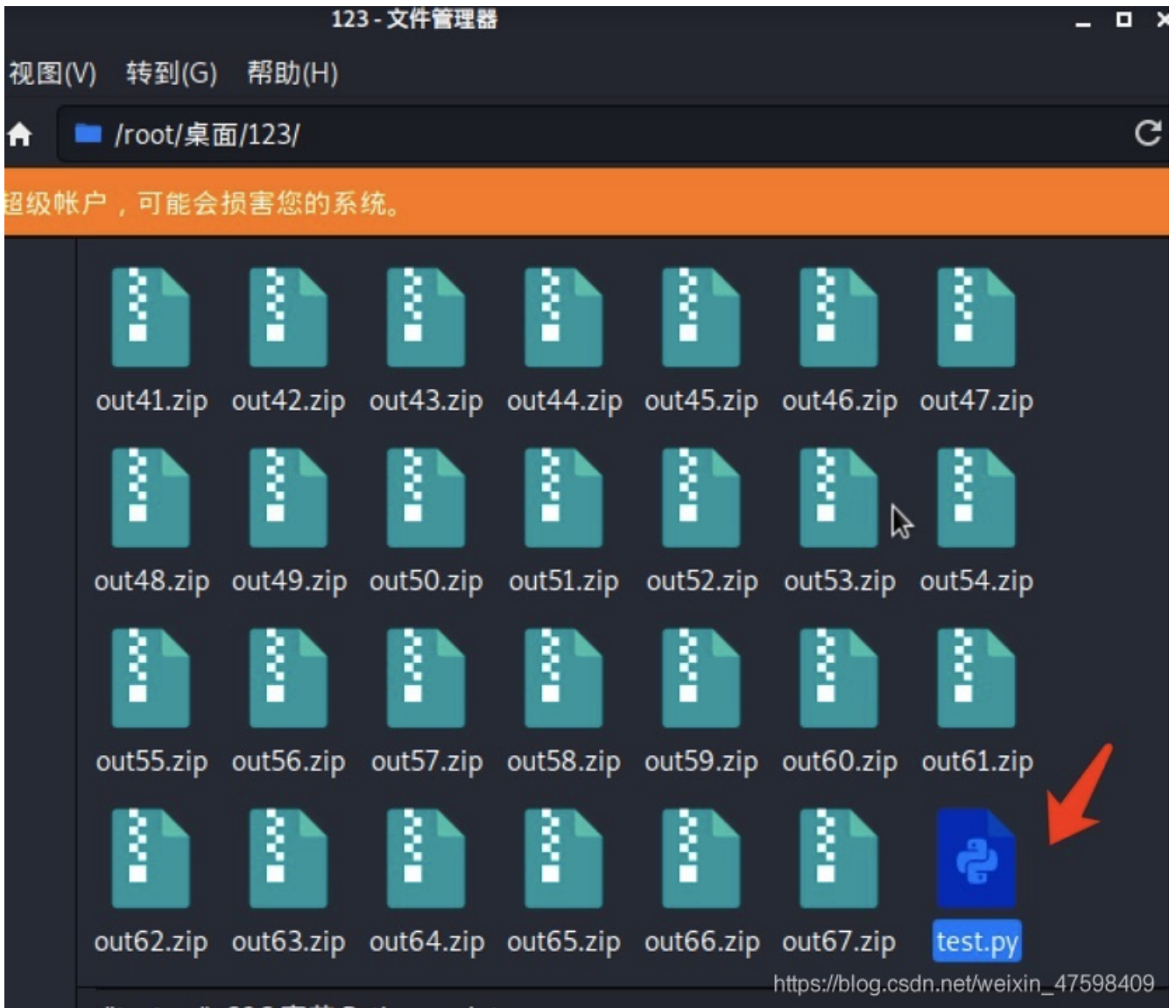
def CrackZip():
    for I in range(68):
        file = 'out' + str(I) + '.zip'
        f = zipfile.ZipFile(file, 'r')
        GetCrc = f.getinfo('data.txt')
        crc = GetCrc.CRC

        #print hex(crc)
        CrackCrc(crc)

dic = string.ascii_letters + string.digits + '+/='

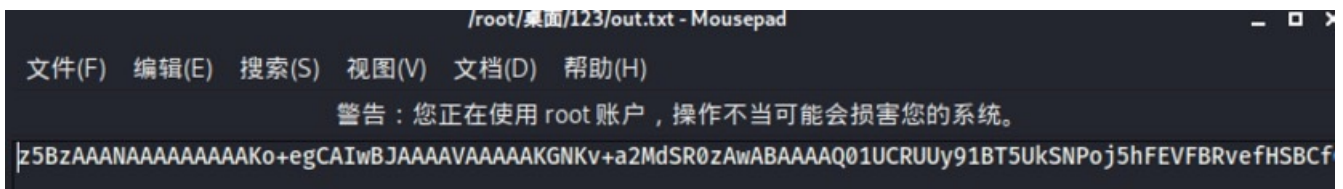
f = open('out.txt', 'w')
CrackZip()
f.close()
```

并且把脚本放在123文件夹里面



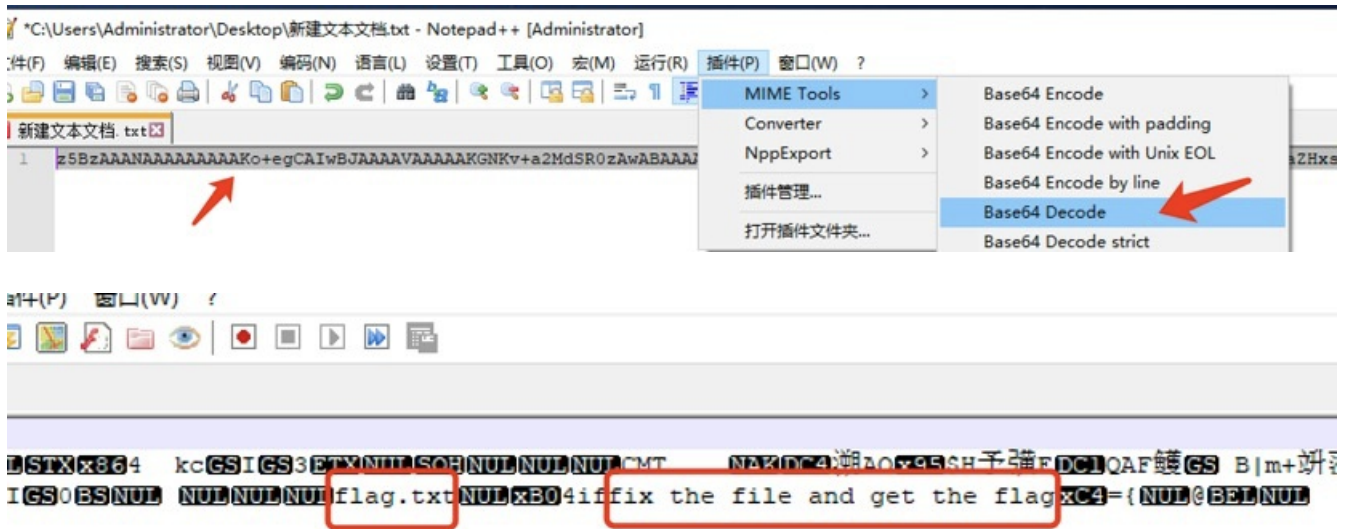
因为68个数量有点多，所以碰撞的话有点久

打开out.txt发现有一串base64编码



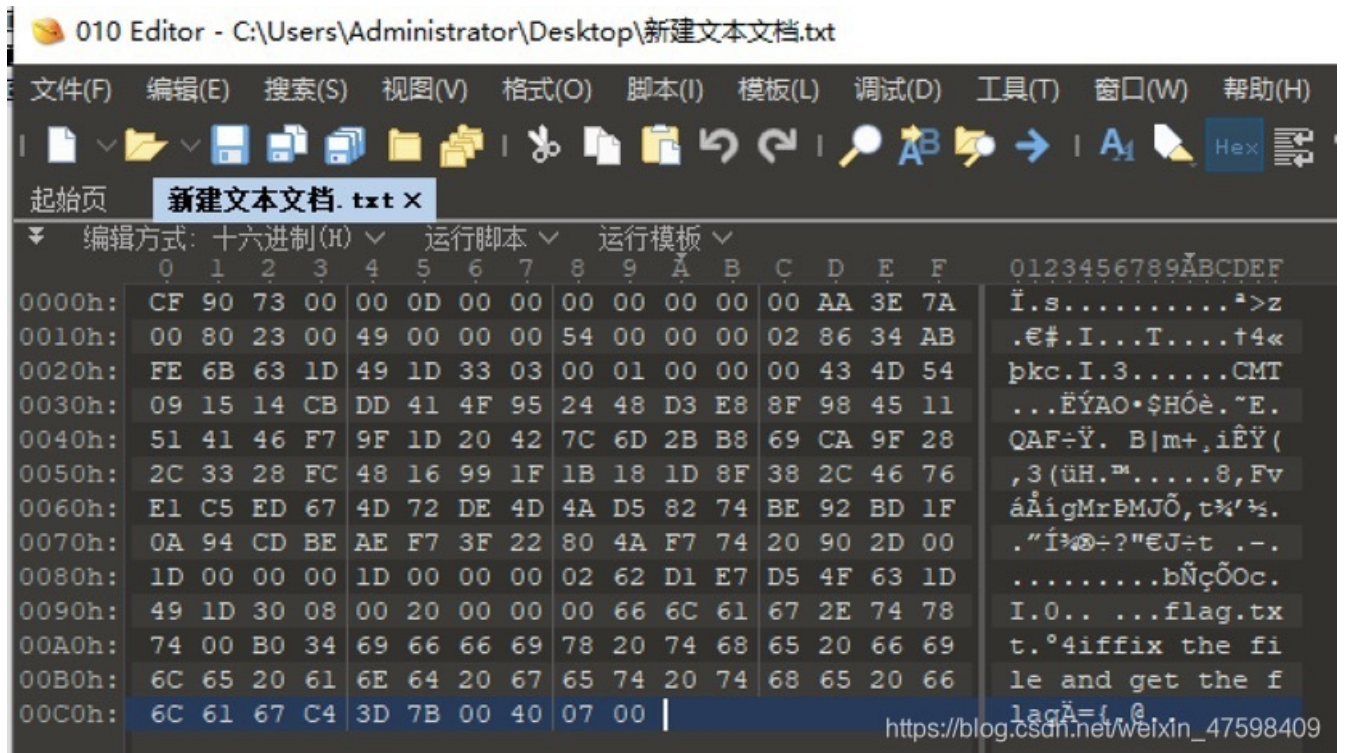
于是用notepad++打开，并且用里面的插件进行解码



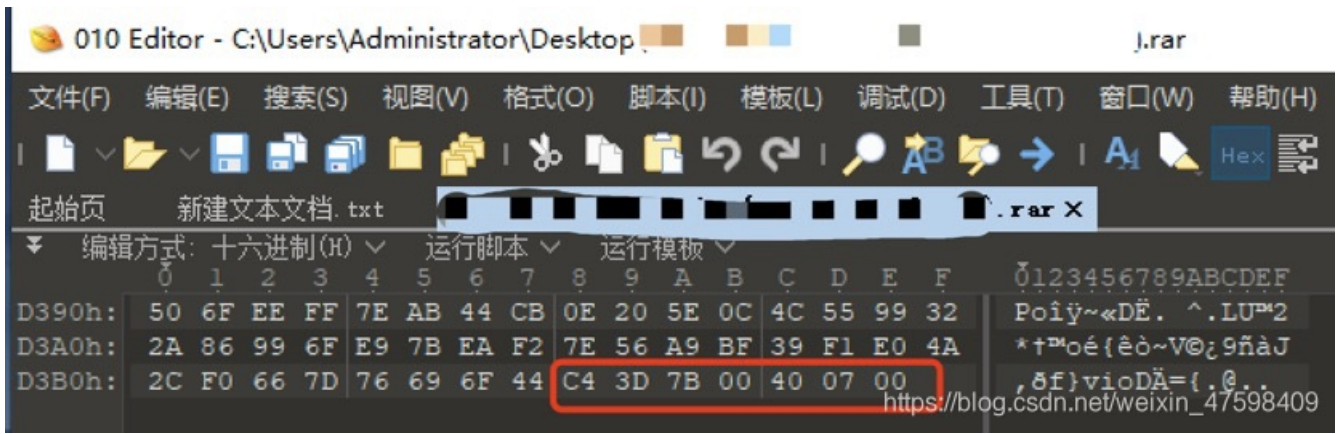


Flag.txt fix the file and get the flag ----里面有一个flag的文本，提示说要修复文件并且 获得flag

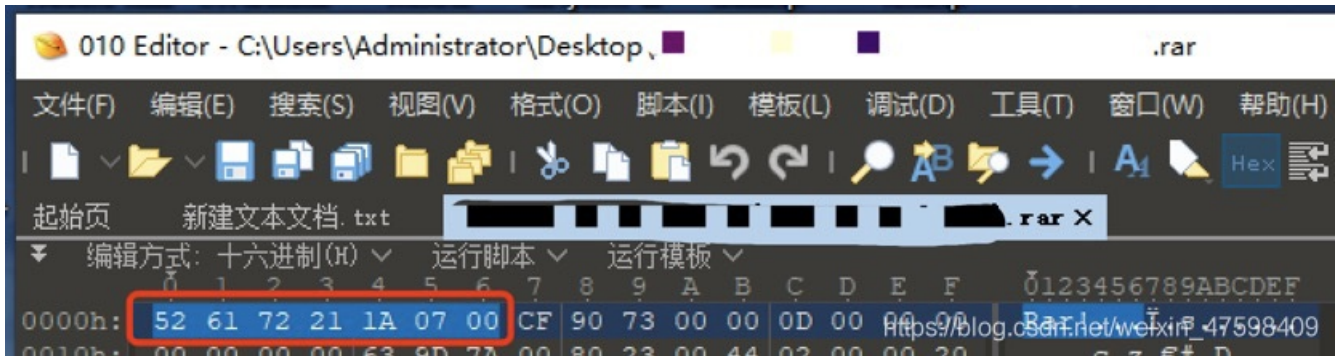
然后我就用010打开这个base64解码后的文本



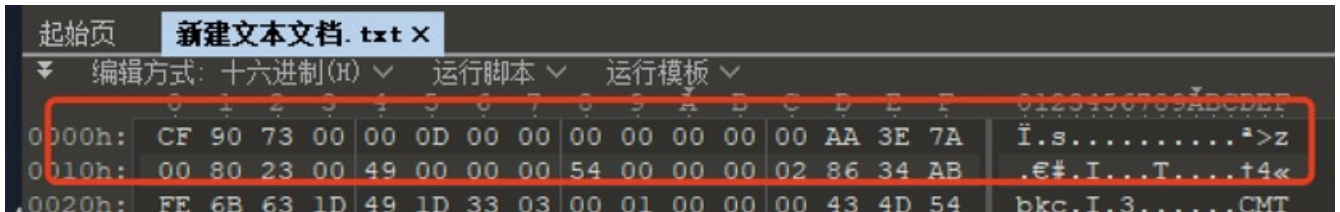
发现文件末尾是C43D7B00400700，不知道是什么东西，百度查了一下大佬们的WP说是rar文件尾，于是自己觉得不是很相信，就随便用一个rar文件导入进去看一下



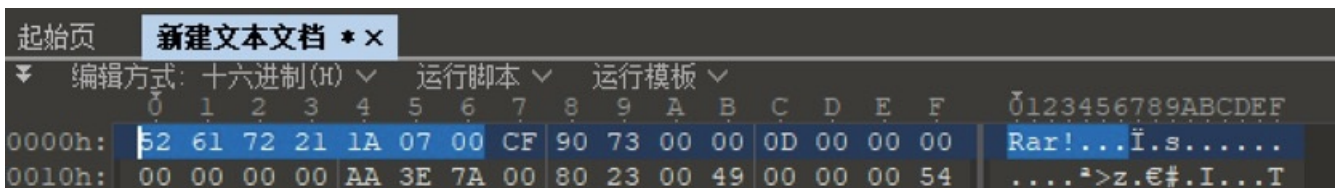
结果还真的是rar文件尾，然后我猜应该是要我们修复文件头吧



这个526172211A0700是rar的文件头

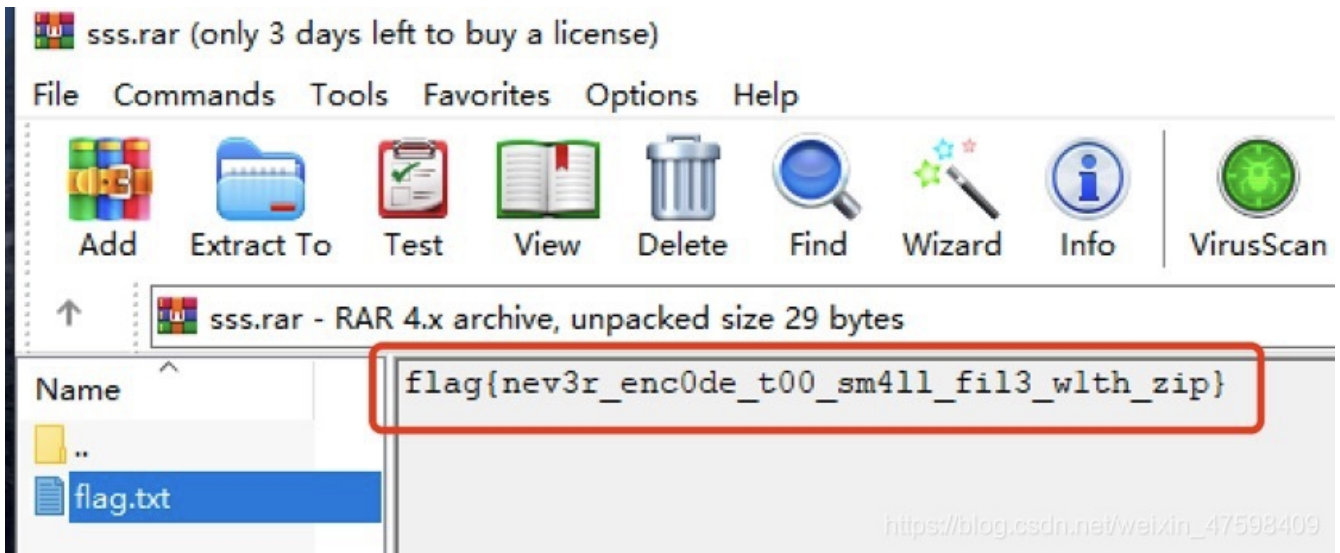


只要我把rar头添加上去就可以了



另存为rar文件





打开压缩包即可获得flag

flag{nev3r\_enc0de\_t00\_sm4ll\_fil3\_wlth\_zip}

### Bugku-MISC-一个普通的压缩包



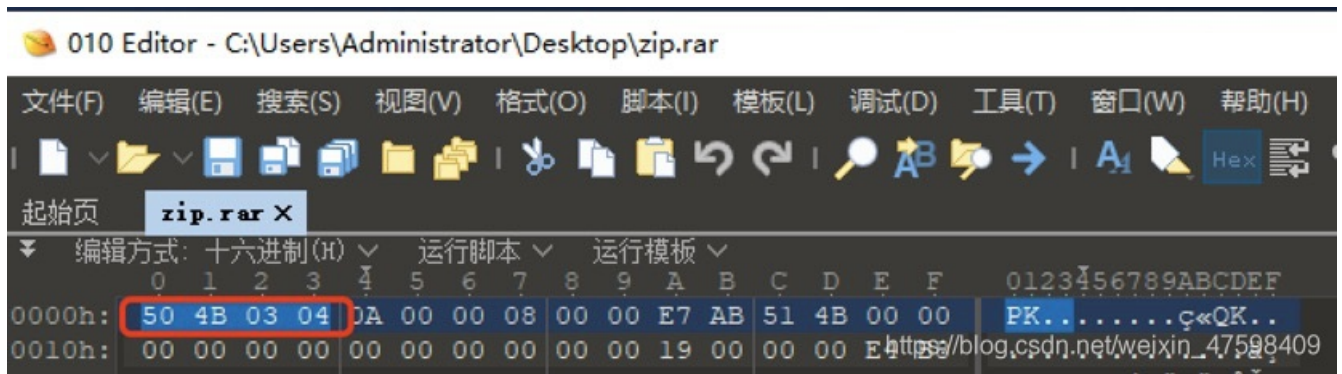
下载附件zip.rar

然后解压里面的flag.rar压缩包出现了这种情况



Secret.png ???我猜应该里面有一张图

但是这样是解压出来是没有图片的，然后把刚刚题目的附件放进010或者winhex看了一下发现是50 4B 03 04 明显是zip文件头



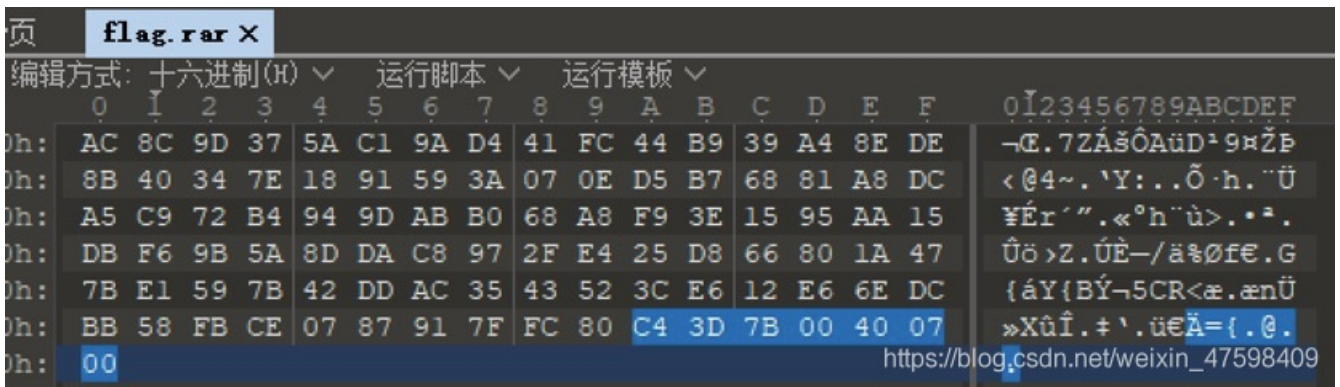
直接修改这个文件的后缀名，我这里改成是aaa.zip

再次用winrar解压flag.rar，还是出现这种情况



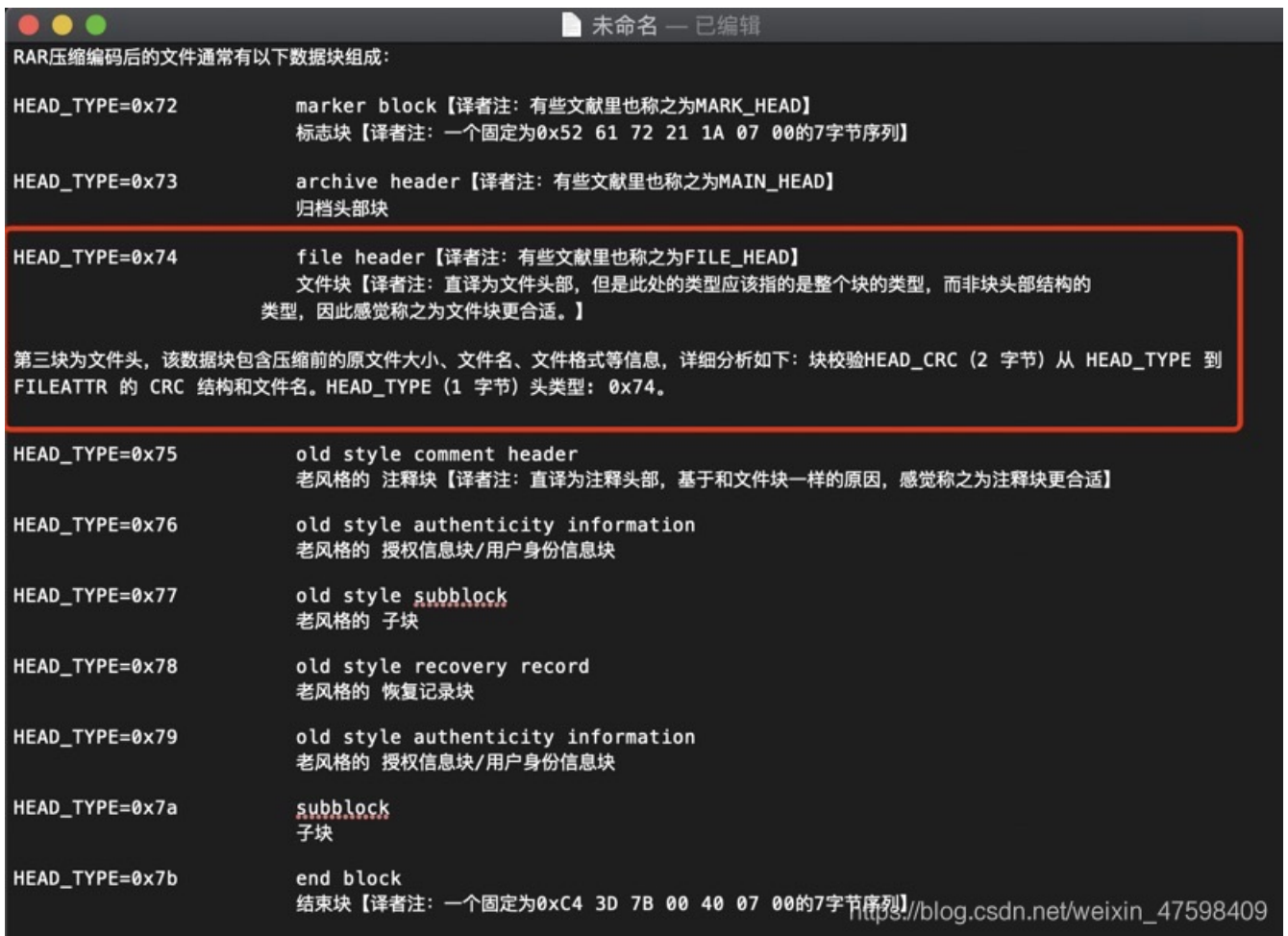
文本就只有一句话 flag is not here

然后把这个flag.rar文件放进010查看



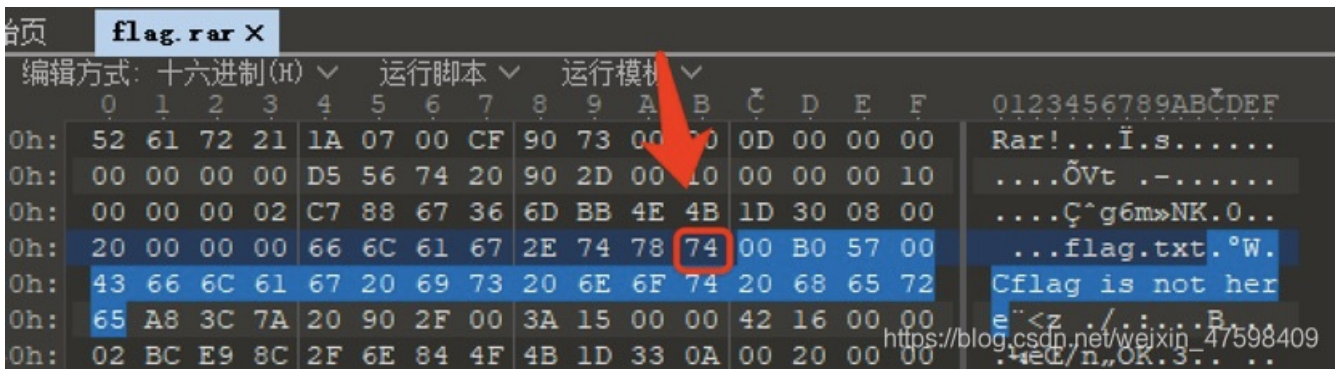
文件头和尾都正常，然后查了一些资料终于发现是这题涉及到rar文件的格式编码  
 详情参考<https://blog.csdn.net/vevenlcf/article/details/51538837>

如果用WinRAR的修复功能，则只能修复rar文件，不能把这个图片复原，rar格式编码中：



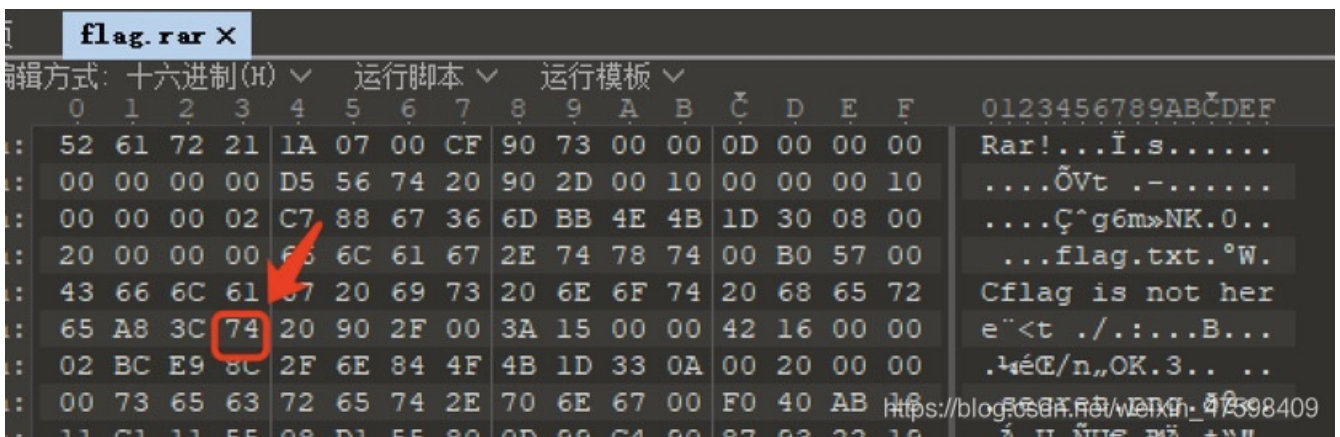
然后根据这个编码的解释再来分析一下这个rar文件



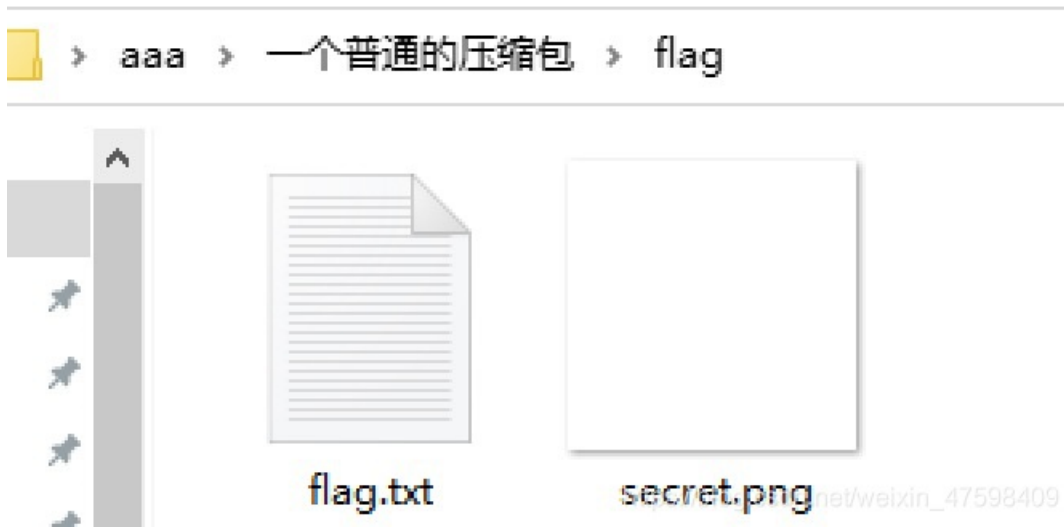


红色框圈住74就是rar里面的flag.txt文本头部编码，这个文本的十六进制一直从00到65才结束，这样可以判断出74是在rar文件的格式编码的一种文件数据块头部，里面包含文件大小、文件名、信息等等

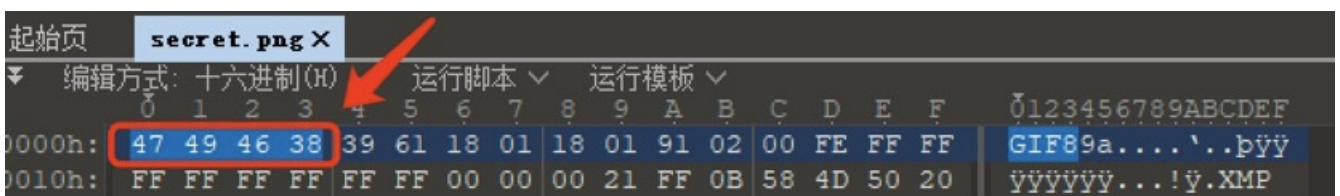
由此可见rar里面还有一张名为Secret.png的图，在flag.txt结束的地方就是下一个块开始的地方，所以png文件的头部编码为74，我们修改过来即可：



保存后就可以正常的用winrar解压缩

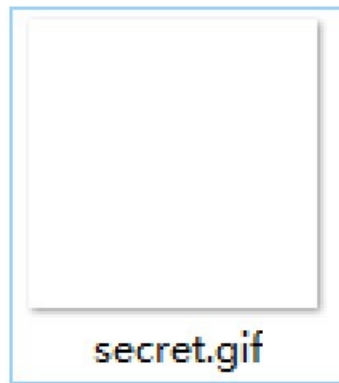


图片终于出来了  
但是很遗憾这个图片是一张空白图片，很不甘心，直接把图片放进010或者winhex查看



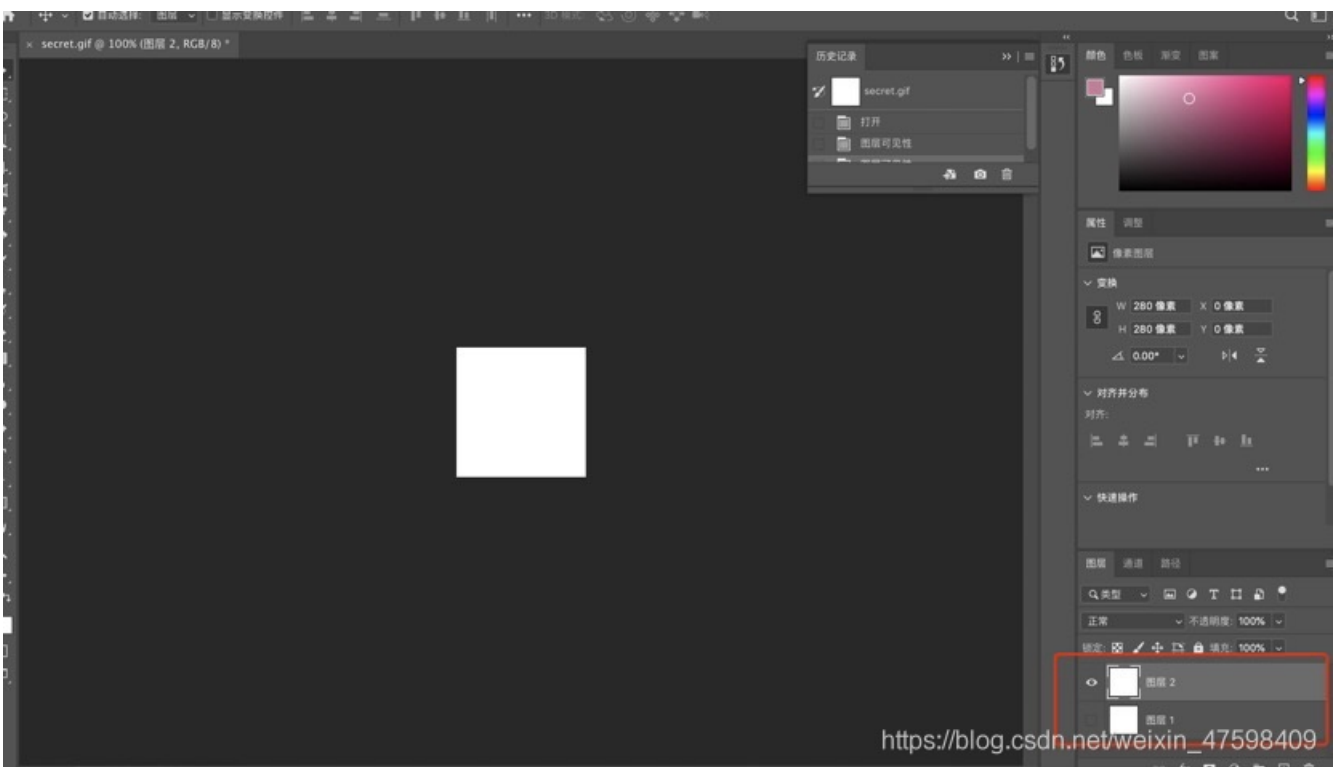
这个47 49 46 38 很明显是GIF的文件头，直接改后缀名

压缩包 > flag

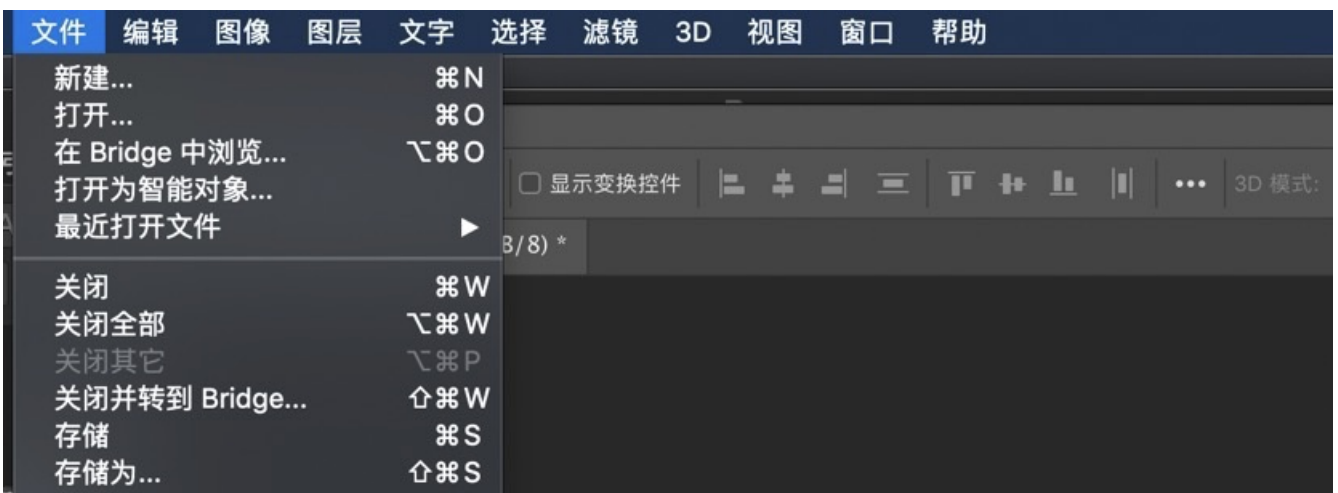


[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

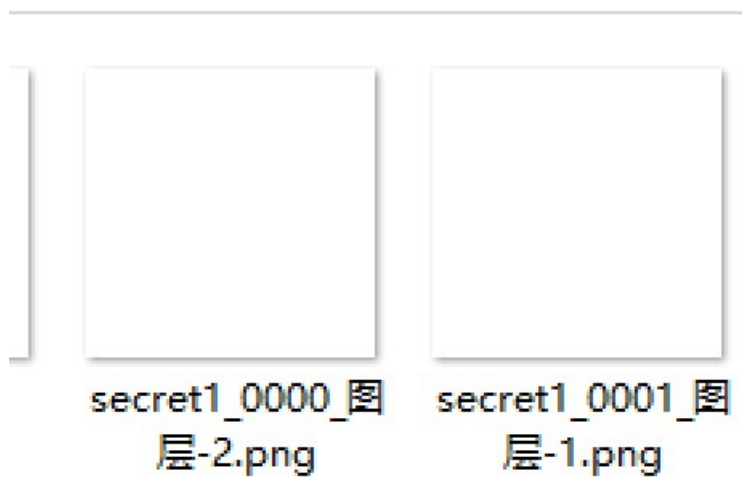
但是改了后缀还是空白图，然后我就用ps打开这个文件，发现有两个白色图层



然后进行每个图层单独分离为png文件，共两个

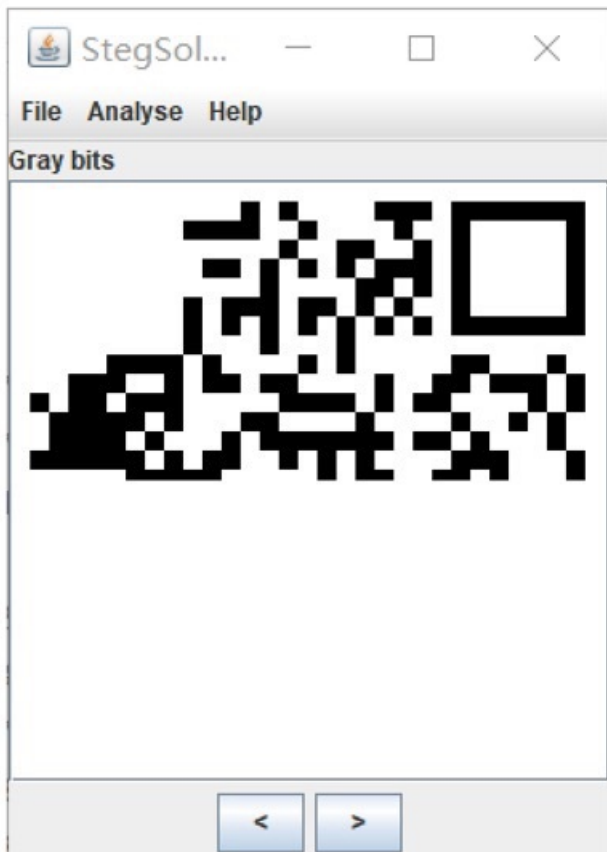






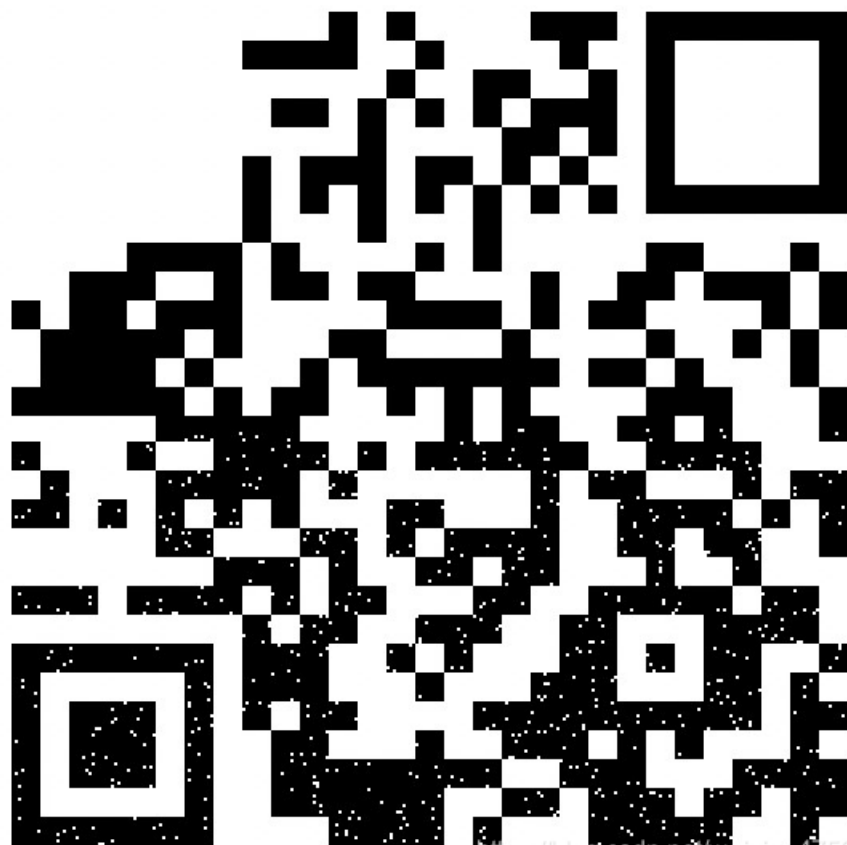
[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

然后我用stegsolve分别打开这两个图



[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

然后把两个图进行合并



[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

这个时候扫描得不出什么信息，二维码的定位符都不完整，左上角的不说了，没法补上，右上角可以补上黑方块：



[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

然后使用二维码查看工具QR Research直接扫描就可以得到flag



已解码数据 1:

位置: (57.2,38.2)-(1127.1,42.5)-(50.8,1112.0)-(1124.8,1115.9)

颜色正常, 正像

版本: 3

纠错等级: H, 掩码: 4

内容:

flag{yanji4n\_bu\_we1shi}

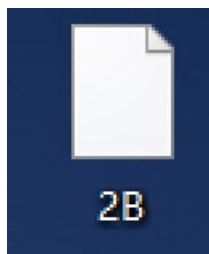


[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

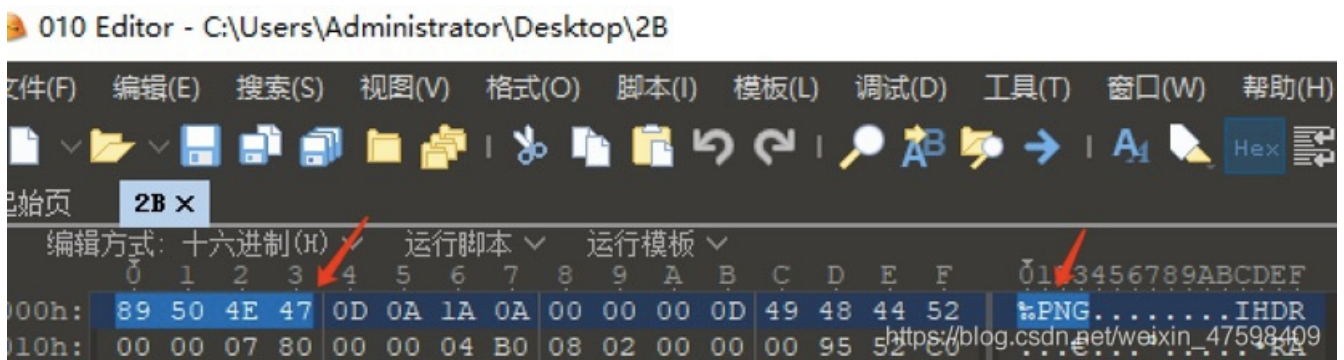
flag{yanji4n\_bu\_we1shi}

**Bugku-MISC-2B**

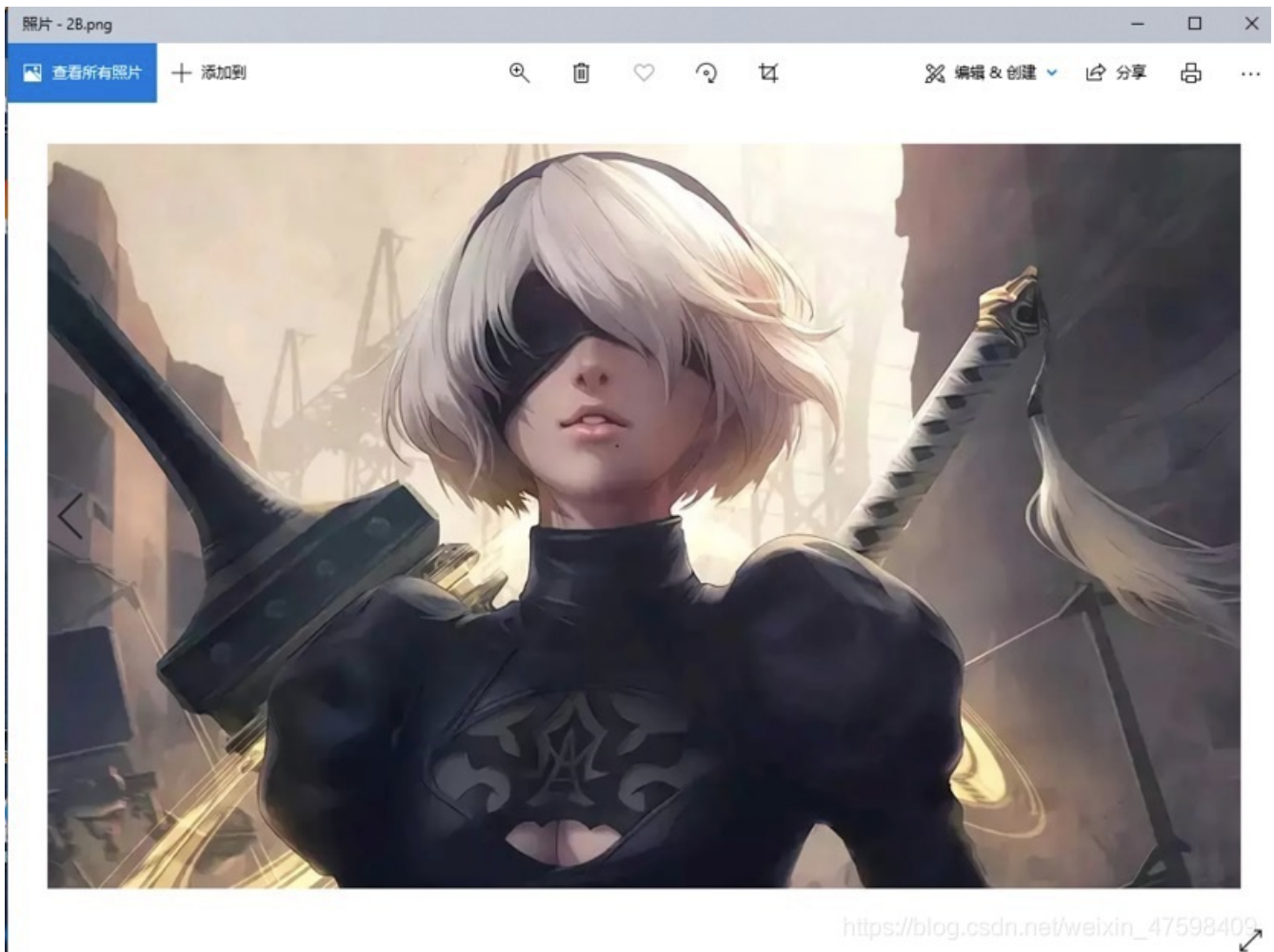




一个这样的文件不知道是什么东西，用010打开看了一下是PNG文件



改后缀，打开图片，这。。。。。。看过图片属性没东西



用binwalk看到了一个zip文件隐藏了

```
[root@root~10:48:21~Tony]
~/桌面 # binwalk 2B.png
```

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	PNG image, 1920 x 1200, 8-bit/color RGB, non-interlaced
2066228	0x1F8734	Zip archive data, at least v2.0 to extract, compressed size: 3065475, uncompress size: 3099670, name: B2.png
5131827	0x4E4E33	End of Zip archive, footer length: 22

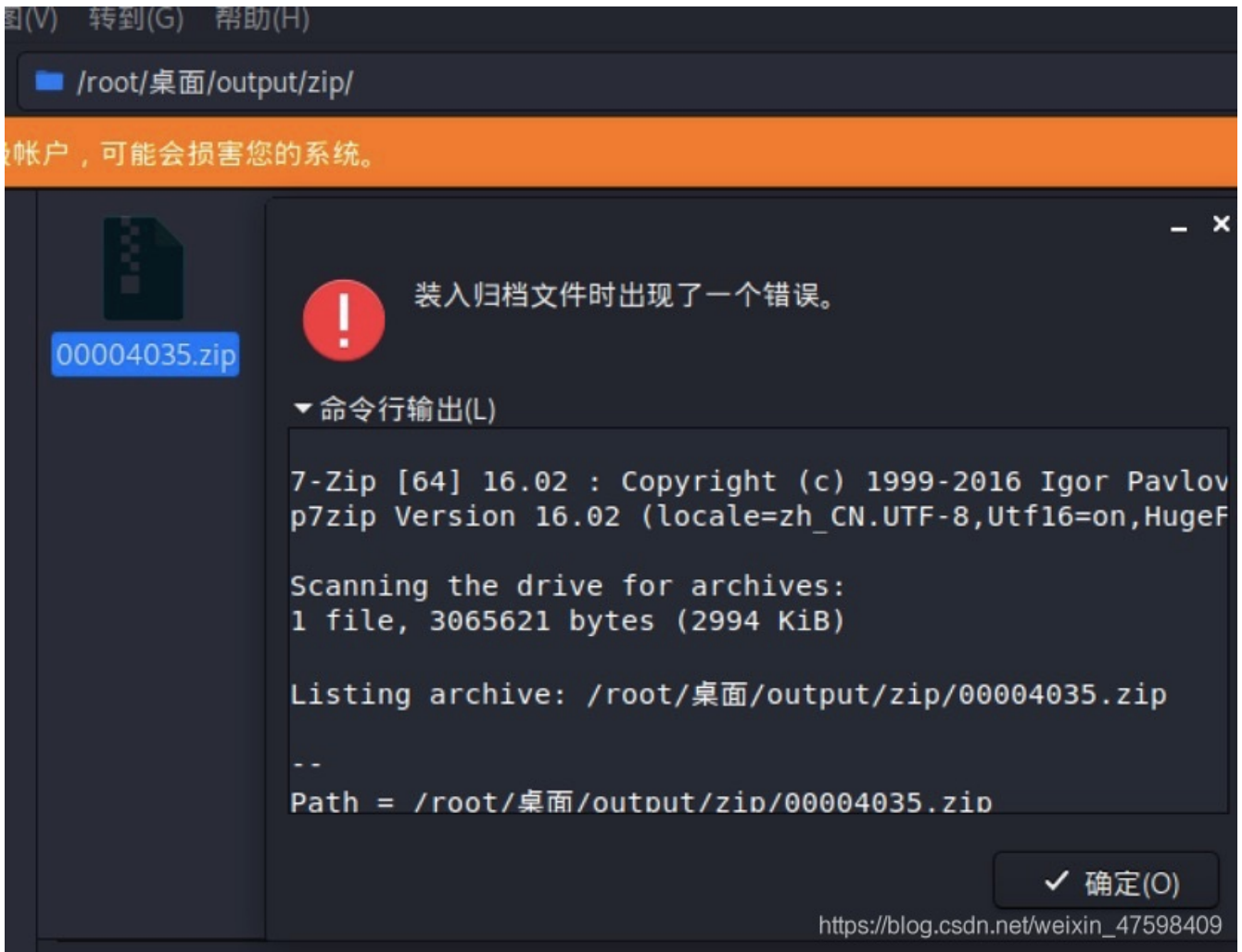
用foremost进行分离

```
[root@root~10:48:40~Tony]
~/桌面 # foremost 2B.png
```

Processing: 2B.png

```
| foundat=B2.png
D@A(H@1
1wfy(n~)wtSg7sN 2W
+fA6r(g)5\ T2|k3V//v'g](EMUtJ
qrZG
t(tFu<Fu S,u1kxPW0x5;6wX$Vo
k%V[u]s/Ud
z,c,F9(0# /c/3K9r9KY 4A
*|
https://blog.csdn.net/weixin_47598409
```

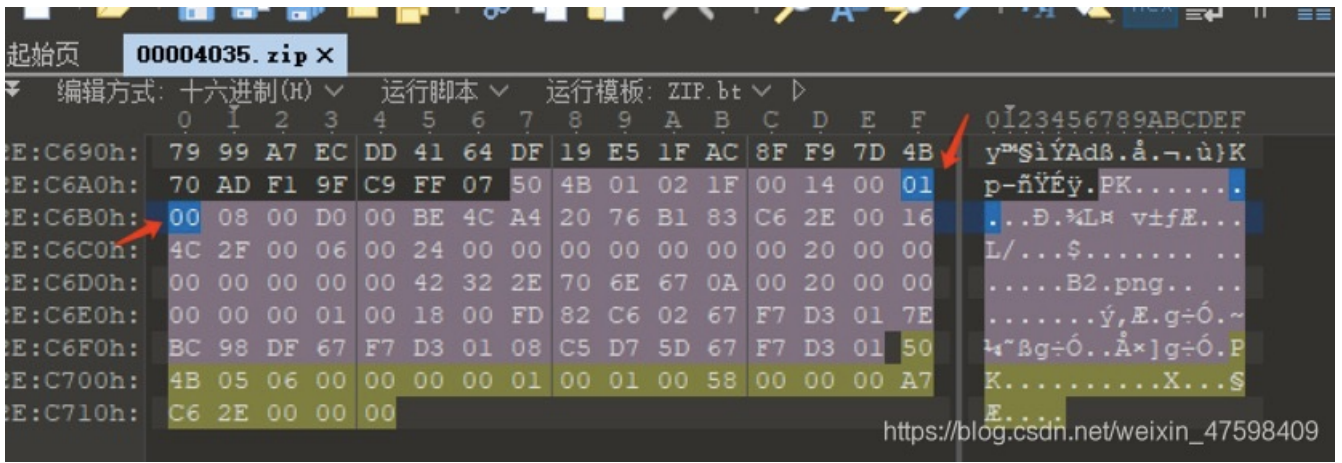
但是在Kali解压出现了错误



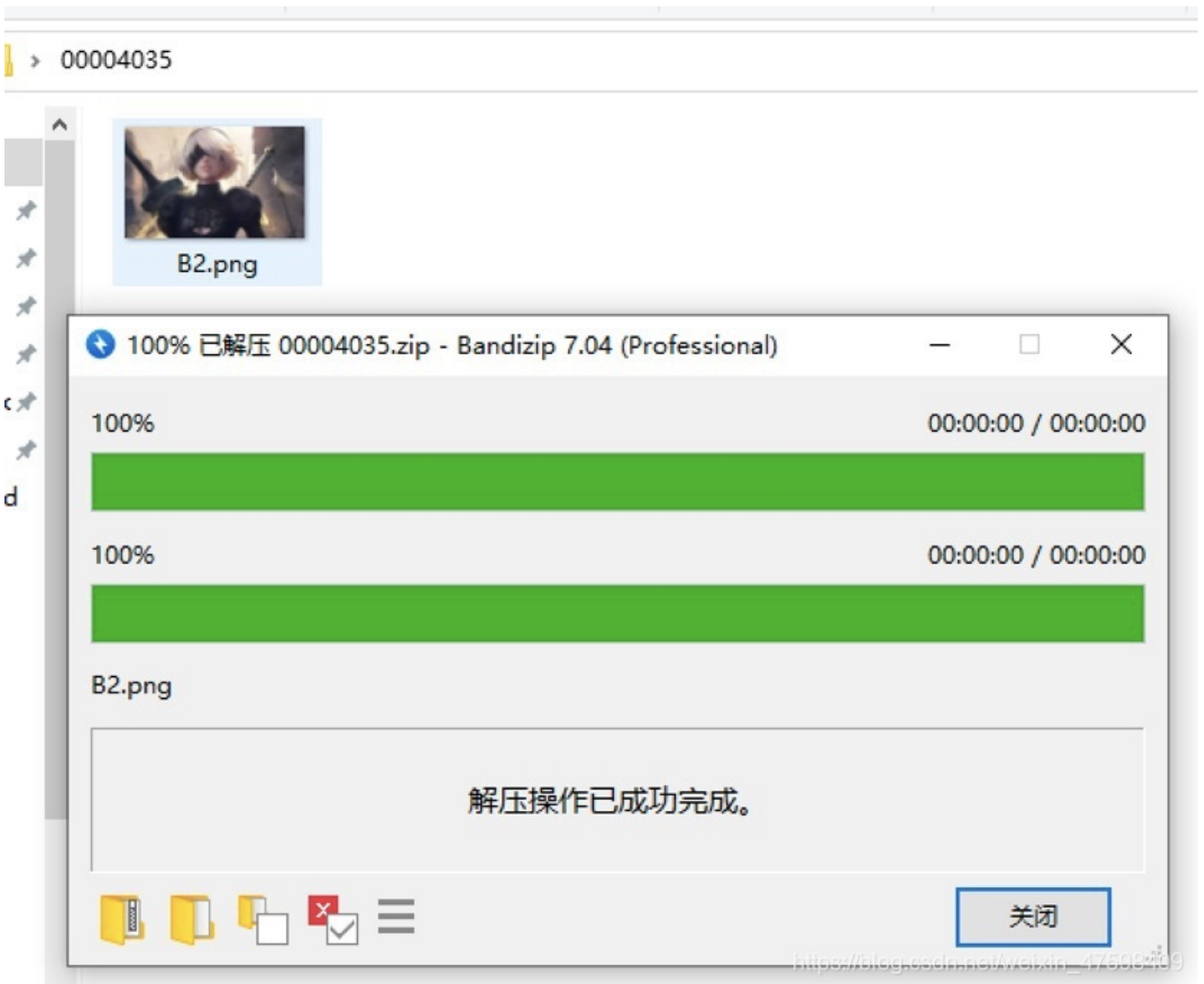
在window系统里面进行解压，发现它要我输入密码



现在只有一种想法，就是这个是一个伪加密ZIP，因为在win打开要输入密码，在Kali打开直接报错，如果正常有密码的话，在两种系统都能正常要你输入密码的，为了验证是不是伪加密，我使用010进行修改一下试试



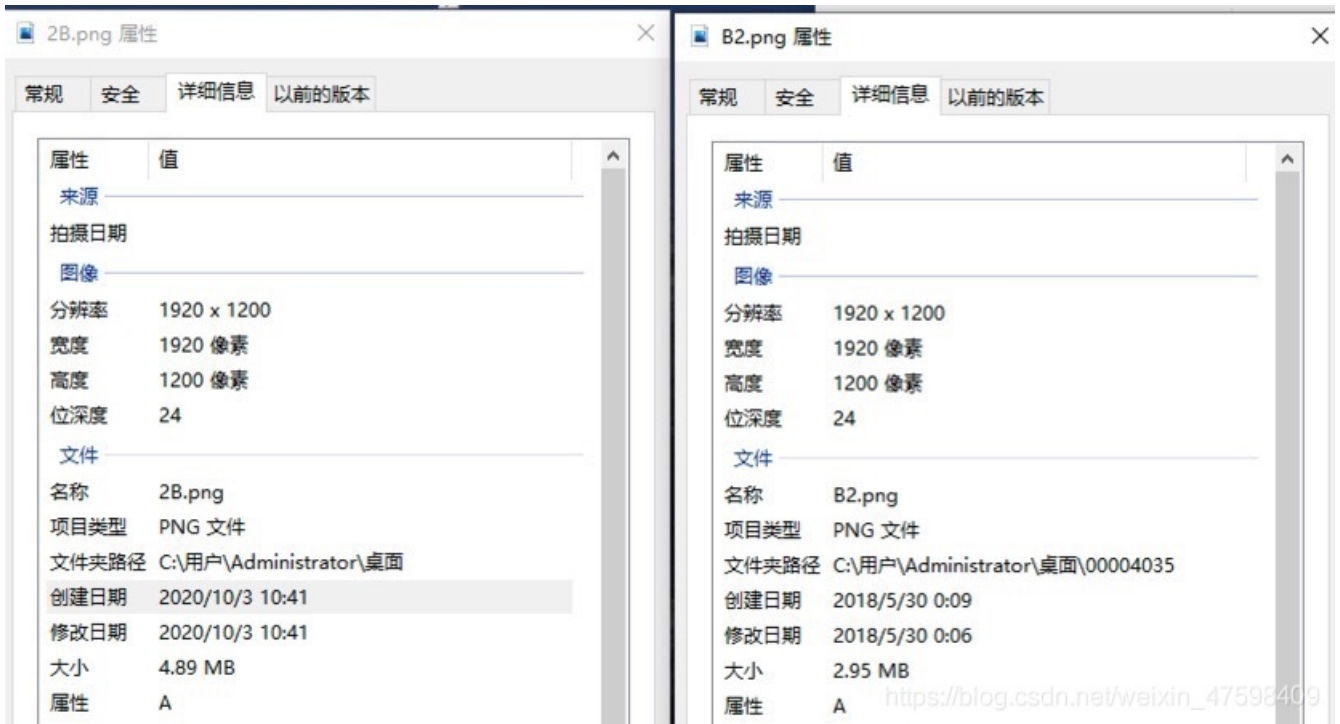
结果发现蓝色选中的那两个十六进制的数为01 00，这个就是伪加密，如果想要把伪加密移除的话要改为00 00



直接解压成功

打开里面发现有一张为B2.png的图片，看上去和题目给的附件的图片好像是一样的，然后我去看一下两张图的属性信息对比一下





结果发现两张图就是除了名字和大小不一样之外，其他的属性都是一样的，我猜应该是图片盲水印攻击，因为前段时间打强网杯的时候也是遇到这种情况，所以有了经验

盲水印攻击脚本下载传送门<https://github.com/linyacool/blind-watermark>

然后把两张图都放在这个文件夹里面，用python2环境跑一下，注明一下B2是无水印图片，2B是有水印图片

```
[root@root ~]# python decode.py --original B2.png --image 2B.png --result flag.png
```



这里要吐槽这个flag，真的恶心，那个F后面的看上去真的很像一个0，但却是一个大写O,那个前括号后面的是i的大写不是L的小写

NUST{I\_10v3\_2B\_F0r3v3r}

**Bugku-MISC-QAQ**

Challenge

232 Solves



# QAQ

## 200

by LordCasser 原创

此题投稿原创题，writeup尽量不要发网上

暂时不放Tips

cipher.txt

QAQ

Flag

Submit

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

下载两个附件

```
cipher.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
FSAnRAIzNIMjPQMjNyBJNTs6NIIFPFIqDDVTJy0zGE8rKxZBJDirJkYoPUQML1M3MDYJZTEIFyl7
UzE6DTtSNxckNDw2Mxk9Jzc=
```

文本是一串base64编码拿去解码一下

## Base64.us Base64 在线编码解码 (最好用的 Base64 在线工具)

Base64 | URLEncode | MD5 | TimeStamp

请输入要进行 Base64 编码或解码的字符

```
FSAnRAIzNIMjPQMjNyBJNTs6NIIFPFIqDDVTJy0zGE8rKxZBJDirJkYoPUQML1M3MDYJZTEIFyl7
UzE6DTtSNxckNDw2Mxk9Jzc=
```

编码 (Encode)

解码 (Decode)

↓ 交换

(编码快捷键: **Ctrl** + **Enter**)

Base64 编码或解码的结果:

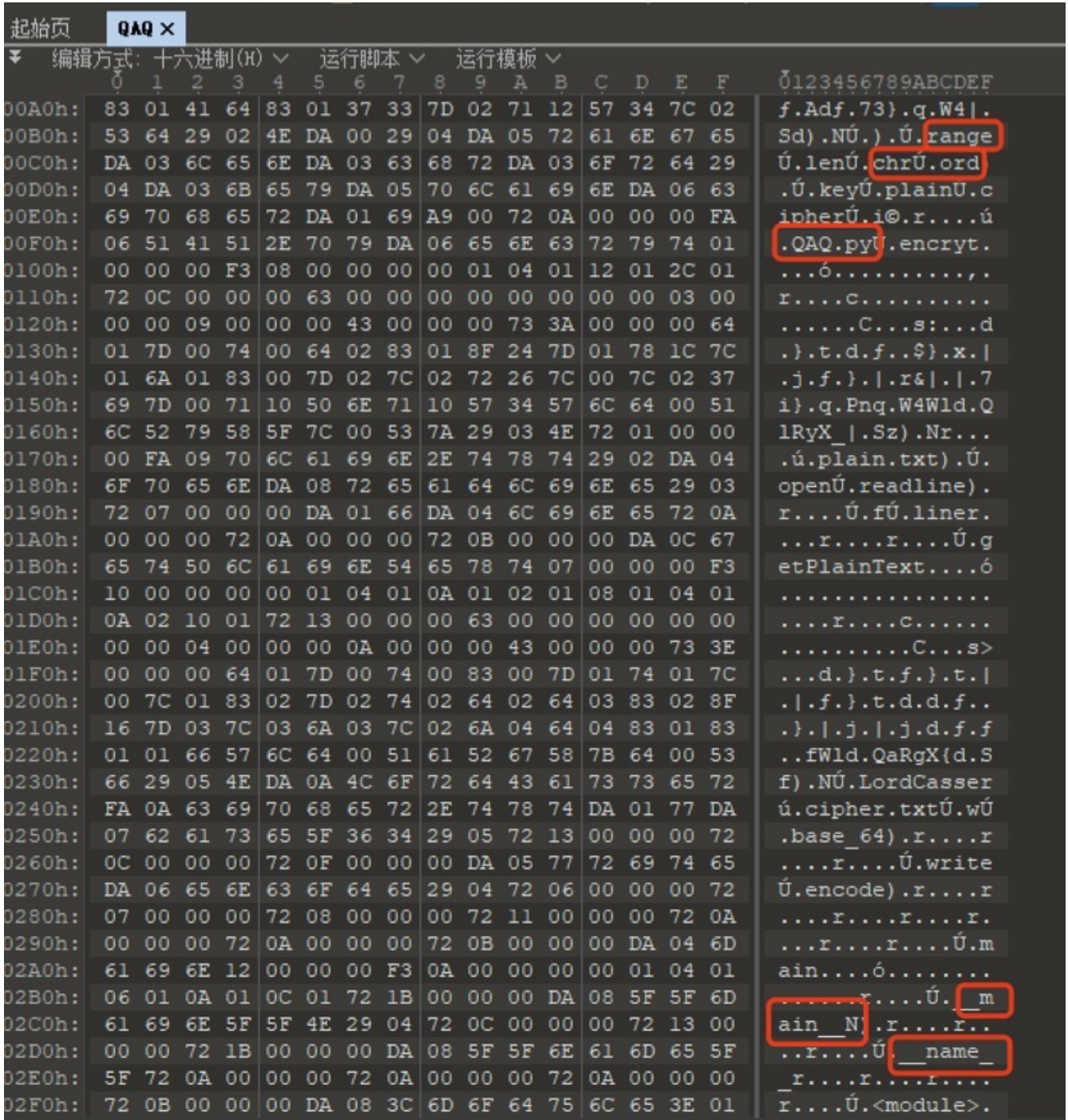
```
□ 'D□36S#=#□#7 I5;:6R□<R*5S'-3□O++□A$2+&F(=D/S706 e1%□";S1:
;R7□$4<63□='
```

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

有点是乱码



然后把QAQ附件放进去010分析



查找结果

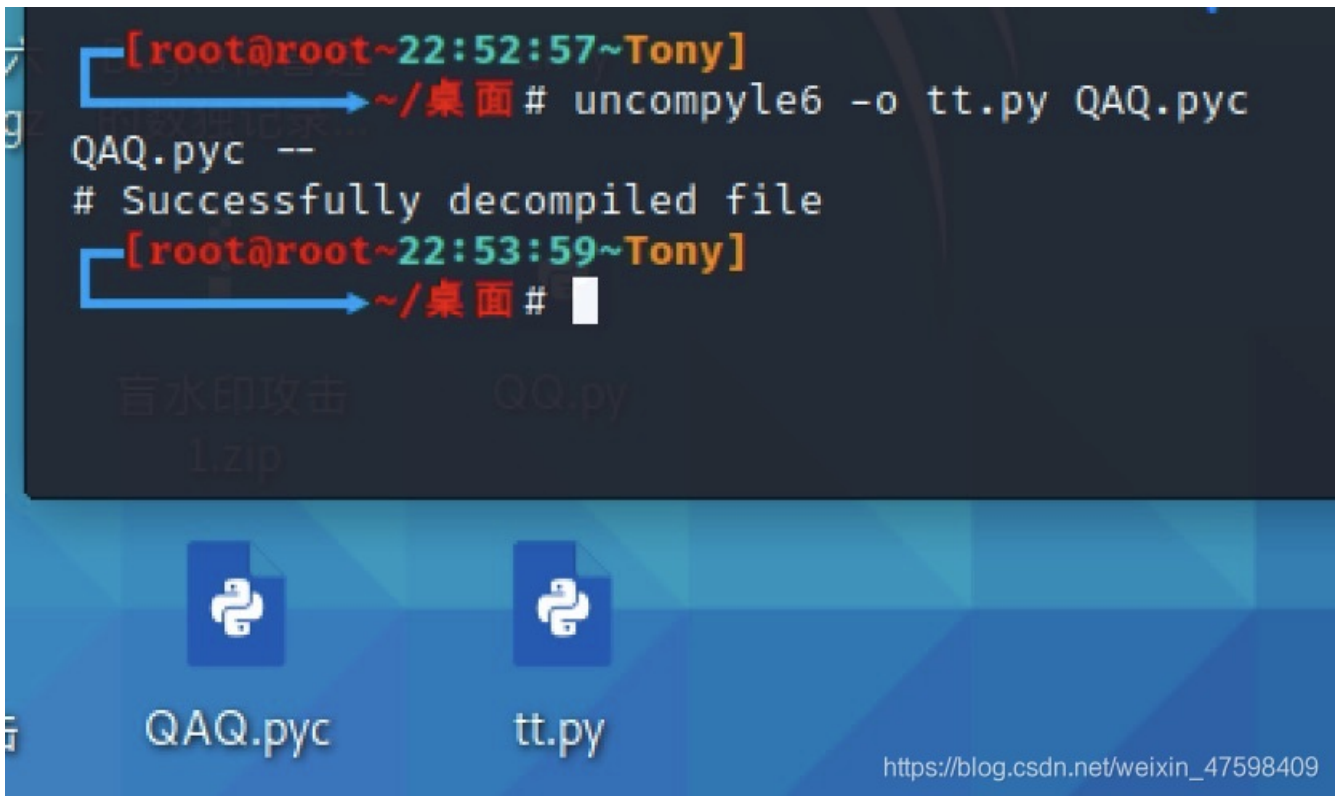
[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

看到chr、ord、range、QAQ.py、main、name基本可以判断是个pyc文件

添加后缀名为pyc文件后用uncompyle6进行pyc反编译

pip install uncompyle6 #安装





得到QAQ.pyc反编译后的tt.py脚本

```
tt.py
~/桌面

# Python bytecode 310 (39??)
# Decompiled from: Python 2.7.18 (default, Apr 20 2020, 20:30:41)
# [GCC 9.3.0]
# Warning: this version of Python has problems handling the Python 3 "byte" type in constants properly.
# Embedded file name: QAQ.py
# Compiled at: 2018-08-15 14:38:31
# Size of source mod 2**32: 620 bytes

def encrypt(key, plain):
    cipher = ''
    for i in range(len(plain)):
        cipher += chr(ord(key[(i % len(key))]) ^ ord(plain[i]))

    return cipher

def getPlainText():
    plain = ''
    with open('plain.txt') as (f):
        while True:
            line = f.readline()
            if line:
                plain += line
            else:
                break

    return plain

def main():
    key = 'LordCasser'
    plain = getPlainText()
    cipher = encrypt(key, plain)
    with open('cipher.txt', 'w') as (f):
        f.write(cipher.encode('base_64'))

if __name__ == '__main__':
    main()
```

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

然后根据以上的加密脚本进行编写一个解密脚本pp.py

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
# Author: virgin-forest
# Time: 2019-03-22 12:00:56
# Describe:

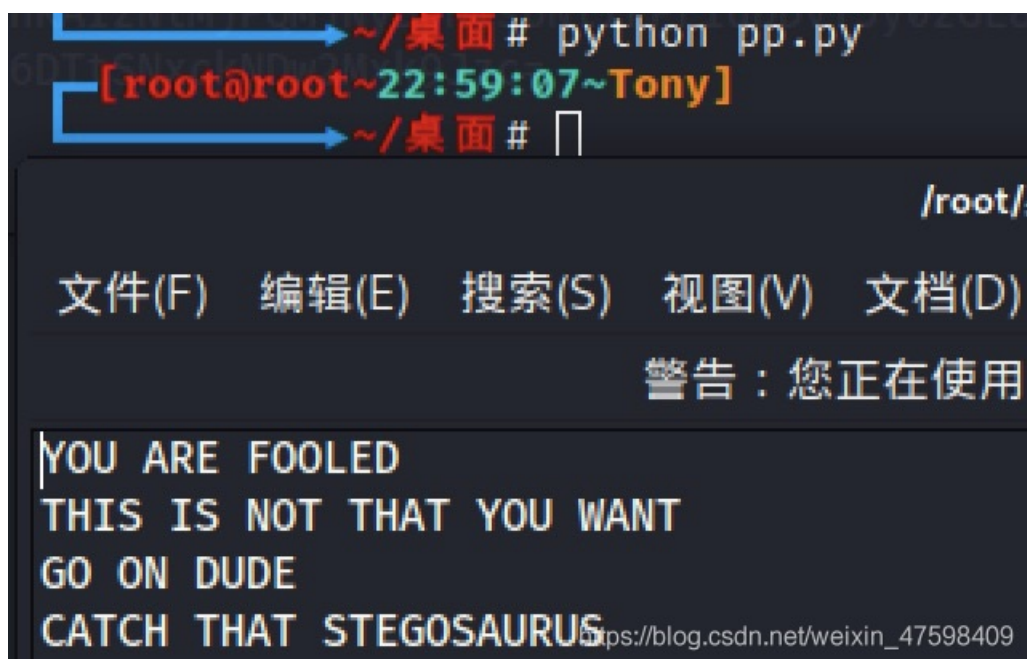
def decrypt(key,plain):
    cipher = ''
    for i in range(len(plain)):
        cipher += chr(ord(key[i % len(key)]) ^ ord(plain[i]))
    return cipher

def getPlainText():
    plain = ''
    with open('cipher.txt') as (f):
        while True:
            line = f.readline()
            if line:
                plain += line
            else:
                break
    return plain.decode('base_64')

def main():
    key = 'LordCassen'
    plain = getPlainText()
    clear = decrypt(key,plain)
    with open('clear.txt','w') as (f):
        f.write(clear)

if __name__ == '__main__':
    main()
```

得到clear.txt文本是一些明文



翻译一波



发现一个单词STEGOSAURUS——剑龙??

于是百度一波

## Python隐写工具

用于在Python字节码中嵌入有效负载的隐写工具。

Stegosaurus是一种**隐写术工具**，允许在Python字节码（pyc或pyo）文件中嵌入任意有效载荷。嵌入过程不会改变运营文件的运行时行为或文件大小，并且通常会导致编码密度较低。有效负载分散在整个字节码中，因此工具 `strings` 不会显示实际的有效负载。`dis` 在使用Stegosaurus嵌入有效负载之前和之后，Python的模块将返回相同的字节码结果。此时，没有关于这种类型的有效载荷传递的先前工作或检测方法。

剑龙需要Python 3.6或更高版本。

### 用法

```
$ python3 -m stegosaurus -h
用法: stegosaurus.py [-h] [-p PAYLOAD] [-r] [-s] [-v] [-x] 载体
```

#### 位置参数:

运营商运营商py, pyc或pyo文件

#### 可选参数:

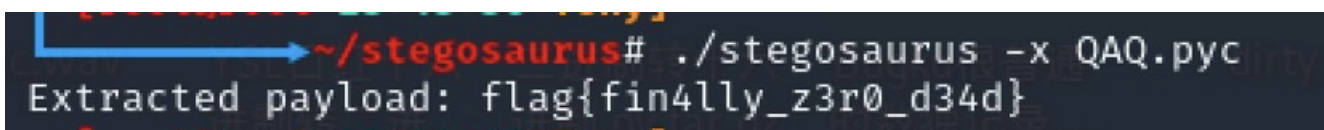
- h, - help显示此帮助消息并退出
- p PAYLOAD, - payload PAYLOAD  
将有效负载嵌入运营商文件中
- r, - report报告运营商支持的最大可用载荷大小
- s, - side-by-side不要覆盖运营商文件，并排安装代替。
- v, - verbose每次使用增加一次详细程度
- x, - extra从运营商文件中提取有效负载

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

然后去下载stegosaurus脚本

传送门<https://github.com/AngelKitty/stegosaurus>

执行这个脚本，分离出flag



flag{fin4lly\_z3r0\_d34d}

科普一下这个stegosaurus的知识和用法



提示告诉我们用 `STEGOSAURUS` 工具进行隐写的，我们直接将隐藏的payload分离出来即可。

```
python3 stegosaurus.py -x QAQ.pyc
```

```
*( 11/29/18@ 4:14下午 )( python@Sakura ):~/桌面
python3 stegosaurus.py -x QAQ.pyc
Extracted payload: flag{fin4lly_z3r0_d34d}
```

我们得到了最终的 `flag` 为: `flag{fin4lly_z3r0_d34d}`

既然都说到这个份子上了，我们就来分析一下我们是如何通过 `Stegosaurus` 来嵌入 `Payload` 。

我们仍然以上面这个代码为例子，我们设置脚本名称为 `encode.py` 。

第一步，我们使用 `Stegosaurus` 来查看在不改变源文件 `(Carrier)` 大小的情况下，我们的 `Payload` 能携带多少字节的数据：

```
python3 -m stegosaurus encode.py -r
```

```
*( 11/29/18@ 4:22下午 )( python@Sakura ):~/桌面
python3 -m stegosaurus encode.py -r
Carrier can support a payload of 24 bytes
```

现在，我们可以安全地嵌入最多24个字节的 `Payload` 了。如果不想覆盖源文件的话，我们可以使用 `-s` 参数来单独生成一个嵌入了 `Payload` 的 `py` 文件：

```
python3 -m stegosaurus encode.py -s --payload "flag{fin4lly_z3r0_d34d}"
```

```
*( 11/29/18@ 4:24下午 )( python@Sakura ):~/桌面
python3 -m stegosaurus encode.py -s --payload "flag{fin4lly_z3r0_d34d}"
Payload embedded in carrier
```

现在我们可以用 `ls` 命令查看磁盘目录，嵌入了 `Payload` 的文件( `carrier` 文件)和原始的字节码文件两者大小是完全相同的：  
[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

现在我们可以用 `ls` 命令查看磁盘目录，嵌入了 `Payload` 的文件( `carrier` 文件)和原始的字节码文件两者大小是完全相同的：

```
*( 11/29/18@ 4:34下午 )( python@Sakura ):~/桌面
ls -l __pycache__/encode.cpython-36.pyc
-rw-r--r-- 1 python python 785 11月 29 16:30 __pycache__/encode.cpython-36.pyc
*( 11/29/18@ 4:35下午 )( python@Sakura ):~/桌面
ls -l __pycache__/encode.cpython-36-stegosaurus.pyc
-rw-r--r-- 1 python python 785 11月 29 16:30 __pycache__/encode.cpython-36-stegosaurus.pyc
```

注：如果没有使用 `-s` 参数，那么原始的字节码文件将会被覆盖。

我们可以通过向 `Stegosaurus` 传递 `-x` 参数来提取出 `Payload` :

```
python3 -m stegosaurus __pycache__/encode.cpython-36-stegosaur
```

```
*( 11/29/18@ 4:35下午 )( python@Sakura ):~/桌面
python3 -m stegosaurus __pycache__/encode.cpython-36-stegosaurus.pyc -x
Extracted payload: flag{fn4lly_z3r0_d34d}
```

我们构造的 `Payload` 不一定要是一个 `ASCII` 字符串, `shellcode` 也是可以的:

```
*( 11/29/18@ 4:39下午 )( python@Sakura ):~/桌面
python3 -m stegosaurus encode.py -s --payload "\xeb\x2a\x5e\x89\x76"
Payload embedded in carrier
*( 11/29/18@ 4:42下午 )( python@Sakura ):~/桌面
python3 -m stegosaurus __pycache__/encode.cpython-36-stegosaurus.pyc -x
Extracted payload: \xeb\x2a\x5e\x89\x76
```

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

我们重新编写一个 `example.py` 模块, 代码如下:

```
import sys
import os
import math
def add(a,b):
    return int(a)+int(b)
def sum1(result):
    return int(result)*3
def sum2(result):
    return int(result)/3
def sum3(result):
    return int(result)-3
def main():
    a = 1
    b = 2
    result = add(a,b)
    print(sum1(result))
    print(sum2(result))
    print(sum3(result))
if __name__ == "__main__":
    main()
```

我们让它携带 `Payload` 为 `flag_is_here` .

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)



我们让它携带 Payload 为 flag\_is\_here 。

```
#!/bin/bash
python3 -m stegosaurus example.py -r
Carrier can support a payload of 14 bytes
python3 -m stegosaurus example.py -s --payload "flag_is_here"
Payload embedded in carrier
python3 -m stegosaurus __pycache__/example.cpython-36-stegosaurus.pyc -x
Extracted payload: flag_is_here
```

我们可以查看嵌入 Payload 之前和之后的 Python 代码运行情况：

```
#!/bin/bash
python3 example.py
9
1.0
0
python3 __pycache__/example.cpython-36.pyc
9
1.0
0
python3 __pycache__/example.cpython-36-stegosaurus.pyc
9
1.0
0
```

通过 strings 查看 Stegosaurus 嵌入了 Payload 之后的文件输出情况(payload 并没有显示出来)：

```
#!/bin/bash
strings __pycache__/example.cpython-36-stegosaurus.pyc
rSe)
int)
example.py
hSe)
resultr
sum1
sS_)
sum2
_Si)
sum3
printr
main
__main__
mathr
__name__r
<module>
```

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

通过 strings 查看 Stegosaurus 嵌入了 Payload 之后的文件输出情况(payload 并没有显示出来)：

```
#!/bin/bash
strings __pycache__/example.cpython-36-stegosaurus.pyc
rSe)
int)
example.py
hSe)
resultr
sum1
sS_)
sum2
_Si)
sum3
printr
main
__main__
mathr
__name__r
<module>
#!/bin/bash
python3 __pycache__/example.cpython-36-stegosaurus.pyc
```

```
python3 -m stegosaurus __pycache__/_example.cpython-36-stegosaurus.pyc -x
Extracted payload: flag_is_here
```

接下来使用 Python 的 dis 模块来查看 Stegosaurus 嵌入 Payload 之前和之后的文件字节码变化情况:

嵌入payload之前:

```
python3 -m dis example.py
1          0 LOAD_CONST          0 (0)
          2 LOAD_CONST          1 (None)
          4 IMPORT_NAME         0 (sys)
          6 STORE_NAME        0 (sys)

2          8 LOAD_CONST          0 (0)
```

```
24         72 LOAD_NAME           7 (main)
          74 CALL_FUNCTION        0
          76 POP_TOP
>>       78 LOAD_CONST          1 (None)
          80 RETURN_VALUE
```

嵌入 payload 之后:

```
python3 -m dis example.py
1          0 LOAD_CONST          0 (0)
          2 LOAD_CONST          1 (None)
          4 IMPORT_NAME         0 (sys)
          6 STORE_NAME        0 (sys)

2          8 LOAD_CONST          0 (0)
         10 LOAD_CONST          1 (None)
         12 IMPORT_NAME         1 (os)
         14 STORE_NAME        1 (os)

         16 LOAD_CONST          0 (0)
         18 LOAD_CONST          1 (None)
         20 IMPORT_NAME         1 (os)
         22 STORE_NAME        1 (os)

         24 LOAD_CONST          0 (0)
         26 LOAD_CONST          1 (None)
         28 IMPORT_NAME         1 (os)
         30 STORE_NAME        1 (os)

         32 LOAD_CONST          0 (0)
         34 LOAD_CONST          1 (None)
         36 IMPORT_NAME         1 (os)
         38 STORE_NAME        1 (os)

         40 LOAD_CONST          0 (0)
         42 LOAD_CONST          1 (None)
         44 IMPORT_NAME         1 (os)
         46 STORE_NAME        1 (os)

         48 LOAD_CONST          0 (0)
         50 LOAD_CONST          1 (None)
         52 IMPORT_NAME         1 (os)
         54 STORE_NAME        1 (os)

         56 LOAD_CONST          0 (0)
         58 LOAD_CONST          1 (None)
         60 IMPORT_NAME         1 (os)
         62 STORE_NAME        1 (os)

         64 LOAD_CONST          0 (0)
         66 LOAD_CONST          1 (None)
         68 IMPORT_NAME         1 (os)
         70 STORE_NAME        1 (os)

         72 LOAD_CONST          0 (0)
         74 LOAD_CONST          1 (None)
         76 IMPORT_NAME         1 (os)
         78 STORE_NAME        1 (os)

         80 LOAD_CONST          0 (0)
         82 LOAD_CONST          1 (None)
         84 IMPORT_NAME         1 (os)
         86 STORE_NAME        1 (os)

         88 RETURN_VALUE
```

注: Payload 的发送和接受方法完全取决于用户个人喜好, Stegosaurus 只提供了一种向 Python 字节码文件嵌入或提取 Payload 的方法。但是为了保证嵌入之后的代码文件大小不会发生变化,因此 Stegosaurus 所支持嵌入的 Payload 字节长度十分有限。因此,如果你需要嵌入一个很大的 Payload,那么你可能要将其分散存储于多个字节码文件中了。

为了在不改变源文件大小的情况下向其嵌入 Payload,我们需要识别出字节码中的无效空间(Dead Zone)。这里所谓的无效空间指的是那些即使被修改也不会改变原 Python 脚本正常行为的那些字节数据。

需要注意的是,我们可以轻而易举地找出 Python3.6 代码中的无效空间。Python 的引用解释器 CPython 有两种类型的操作码,即无参数的和有参数的。在版本是低于 3.5 的 Python 版本中,根据操



Python 有内建空白的操作码，即无参数的和有参数的。在版本低于 3.0 的 Python 版本中，根据操作码是否带参，字节码中的操作指令将需要占用 1 个字节或 3 个字节。在 Python3.6 中就不一样了，Python3.6 中所有的指令都占用 2 个字节，并会将无参数指令的第二个字节设置为 0，这个字节在其运行过程中将会被解释器忽略。这就意味着，对于字节码中每一个不带参数的操作指令，Stegosaurus 都可以安全地嵌入长度为 1 个字节的 Payload 代码。

我们可以通过 Stegosaurus 的 -vv 选项来查看 Payload 是如何嵌入到这些无效空间之中的：

```
#( 11/29/18@10:35下午 )( python@Sakura ):~/桌面
python3 -m stegosaurus example.py -s -p "ABCDE" -vv
2018-11-29 22:36:26,795 - stegosaurus - DEBUG - Validated args
2018-11-29 22:36:26,797 - stegosaurus - INFO - Compiled example.py as __pycac
2018-11-29 22:36:26,797 - stegosaurus - DEBUG - Read header and bytecode from
2018-11-29 22:36:26,798 - stegosaurus - DEBUG - POP_TOP (0)
2018-11-29 22:36:26,798 - stegosaurus - DEBUG - POP_TOP (0)
```

```
python3 -m stegosaurus example.py -s -p "ABCDE" -vv
2018-11-29 22:36:26,795 - stegosaurus - DEBUG - Validated args
2018-11-29 22:36:26,797 - stegosaurus - INFO - Compiled example.py as __pycac
2018-11-29 22:36:26,797 - stegosaurus - DEBUG - Read header and bytecode from
2018-11-29 22:36:26,798 - stegosaurus - DEBUG - POP_TOP (0)
2018-11-29 22:36:26,798 - stegosaurus - DEBUG - POP_TOP (0)
2018-11-29 22:36:26,798 - stegosaurus - DEBUG - POP_TOP (0)
2018-11-29 22:36:26,798 - stegosaurus - DEBUG - RETURN_VALUE (0)
2018-11-29 22:36:26,798 - stegosaurus - DEBUG - BINARY_SUBTRACT (0)
2018-11-29 22:36:26,798 - stegosaurus - DEBUG - RETURN_VALUE (0)
2018-11-29 22:36:26,798 - stegosaurus - DEBUG - BINARY_TRUE_DIVIDE (0)
2018-11-29 22:36:26,798 - stegosaurus - DEBUG - RETURN_VALUE (0)
2018-11-29 22:36:26,798 - stegosaurus - DEBUG - BINARY_MULTIPLY (0)
2018-11-29 22:36:26,798 - stegosaurus - DEBUG - RETURN_VALUE (0)
2018-11-29 22:36:26,798 - stegosaurus - DEBUG - BINARY_ADD (0)
2018-11-29 22:36:26,798 - stegosaurus - DEBUG - RETURN_VALUE (0)
2018-11-29 22:36:26,798 - stegosaurus - DEBUG - POP_TOP (0)
2018-11-29 22:36:26,798 - stegosaurus - DEBUG - RETURN_VALUE (0)
2018-11-29 22:36:26,798 - stegosaurus - INFO - Found 14 bytes available for p
Payload embedded in carrier
2018-11-29 22:36:26,799 - stegosaurus - DEBUG - POP_TOP (65) ----A
2018-11-29 22:36:26,799 - stegosaurus - DEBUG - POP_TOP (66) ----B
2018-11-29 22:36:26,799 - stegosaurus - DEBUG - POP_TOP (67) ----C
2018-11-29 22:36:26,799 - stegosaurus - DEBUG - RETURN_VALUE (68) ----D
2018-11-29 22:36:26,799 - stegosaurus - DEBUG - BINARY_SUBTRACT (69) ----E
2018-11-29 22:36:26,799 - stegosaurus - DEBUG - RETURN_VALUE (0)
2018-11-29 22:36:26,799 - stegosaurus - DEBUG - BINARY_TRUE_DIVIDE (0)
2018-11-29 22:36:26,799 - stegosaurus - DEBUG - RETURN_VALUE (0)
2018-11-29 22:36:26,799 - stegosaurus - DEBUG - BINARY_MULTIPLY (0)
2018-11-29 22:36:26,799 - stegosaurus - DEBUG - RETURN_VALUE (0)
2018-11-29 22:36:26,799 - stegosaurus - DEBUG - BINARY_ADD (0)
```

```
2018-11-29 22:36:26,798 - stegosaurus - DEBUG - RETURN VALUE (0)
```

```
2018-11-29 22:36:26,798 - stegosaurus - INFO - Found 14 bytes available for p
Payload embedded in carrier
2018-11-29 22:36:26,799 - stegosaurus - DEBUG - POP_TOP (65) ----A
2018-11-29 22:36:26,799 - stegosaurus - DEBUG - POP_TOP (66) ----B
2018-11-29 22:36:26,799 - stegosaurus - DEBUG - POP_TOP (67) ----C
2018-11-29 22:36:26,799 - stegosaurus - DEBUG - RETURN_VALUE (68) ----D
2018-11-29 22:36:26,799 - stegosaurus - DEBUG - BINARY_SUBTRACT (69) ----E
2018-11-29 22:36:26,799 - stegosaurus - DEBUG - RETURN_VALUE (0)
2018-11-29 22:36:26,799 - stegosaurus - DEBUG - BINARY_TRUE_DIVIDE (0)
2018-11-29 22:36:26,799 - stegosaurus - DEBUG - RETURN_VALUE (0)
2018-11-29 22:36:26,799 - stegosaurus - DEBUG - BINARY_MULTIPLY (0)
2018-11-29 22:36:26,799 - stegosaurus - DEBUG - RETURN_VALUE (0)
2018-11-29 22:36:26,799 - stegosaurus - DEBUG - BINARY_ADD (0)
2018-11-29 22:36:26,799 - stegosaurus - DEBUG - RETURN_VALUE (0)
2018-11-29 22:36:26,799 - stegosaurus - DEBUG - POP_TOP (0)
2018-11-29 22:36:26,799 - stegosaurus - DEBUG - RETURN_VALUE (0)
2018-11-29 22:36:26,799 - stegosaurus - DEBUG - Creating new carrier file nam
2018-11-29 22:36:26,799 - stegosaurus - INFO - Wrote carrier file as __pycach
```

## 参考文献

- <https://bitbucket.org/jherron/stegosaurus/src>
- <https://github.com/AngelKitty/stegosaurus>
- <https://www.freebuf.com/sectool/129357.html>

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

## Bugku-MISC- apple

(这题不会做，拿大佬WP)



# apple 200

Apple在2017年WWDC上发布了全新的APFS，带有一系列新功能。Bugku很好奇，马上试了一下。而且，他给你留下了一些惊喜:)

N1CTF\_APFS\_v...

Flag

Submit

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

附件是一个dmg格式的文件，dmg是MacOS上的压缩镜像文件,相当于在Windows上常见的iso文件

名称	修改日期	类型	大小
 N1CTF_APFS_ver2.dmg	2020/10/6 10:55	DMG 文件	8,344 KB

## apple

Apple在2017年WWDC上发布了全新的APFS，带有一系列新功能。Bugku很好奇，马上试了一下。而且，他给你留下了一些惊喜:)

N1CTF\_APFS\_ver2.dmg

尝试打开发现有密码，使用010editor看到最后有个字符串：

```
8B E9 7A 5D FA 5A D7 75 0A 09 7A 1sR1<ez juZxu..z
D5 3A 0D 4F FD B6 A7 58 BA 8D 86 >, @à™Ö: .Oý¶SX°.†
BD F3 3E 2B 20 E1 21 36 93 0F 41 à>°Y—½ó>+ á!6".A
5F 41 50 46 53 | N1CTF_APFS
```

使用N1CTF\_APFS成功打开并挂载。打开挂载盘看到531个0字节的txt，思路中断...查看writeup得知可以通过以下命令查看快照：

```
1 #查看快照
2 tmtutil listlocalsnapshots /Volumes/N1CTF_APFS
3 #挂载快照
4 mkdir temp
5 mount_apfs -s ctf /Volumes/N1CTF_APFS ./temp/
```

使用以下命令取得文件的修改时间（精确到100ns级）：

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

使用以下命令取得文件的修改时间（精确到100^9级）：

```
1 gstat -c "%n %y" * > ./1.txt
2 gstat -c "%n %y" * > ./2.txt
```

使用脚本分别取两个镜像所有文件修改时间的最后一位秒数，并做异或，然后作为8进制数转2进制，再转Ascii码写入文件。

```
1 import re
2 path1 = './1.txt'
3 path2 = './2.txt'
4 l1 = []
5 l2 = []
6 req = ''
7 with open(path1, 'r') as f1:
8     lines = f1.readlines()
9     for line in lines:
10         req = re.findall(r'[0-9]{4}.txt 2018-02-23 07:33:33.[0-9]{8}([0-9])', line)[0]
11         l1.append(req)
12 with open(path2, 'r') as f2:
13     lines = f2.readlines()
14     for line in lines:
15         req = re.findall(r'[0-9]{4}.txt 2018-02-23 07:33:33.[0-9]{8}([0-9])', line)[0]
16         l2.append(req)
17 b = ''
18 for i in range(len(l1)):
19     f = int(l1[i]) ^ int(l2[i])
20     b += bin(int('0o'+str(f), 8)).replace("0b", "").zfill(3)
21 with open('./result.data', 'wb') as f:
22     for i in range(0, len(b), 8):
23         f.write(chr(int(b[i:i+8], 2)).encode('utf-8'))
```

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

```
22     for i in range(0, len(b), 8):
23         f.write(chr(int(b[i:i+8], 2)).encode('utf-8'))
```

写入之后用010editor打开发现是个zip，使用之间得解压密码N1CTF\_APFS可得flag。

N1CTF{APFS\_a\_N3xt\_30\_Year\_Fileystem}

N1CTF{APFS\_a\_N3xt\_30\_Year\_Fileystem}

Bugku-MISC-妹子的陌陌

Challenge

1308 Solves



# 妹子的陌陌

## 250



想要妹子陌陌号吗?  
做题来拿吧

下载这个图片做题

momo.jpg

Flag

Submit

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

片 - momo.jpg

查看所有照片

+ 添加到



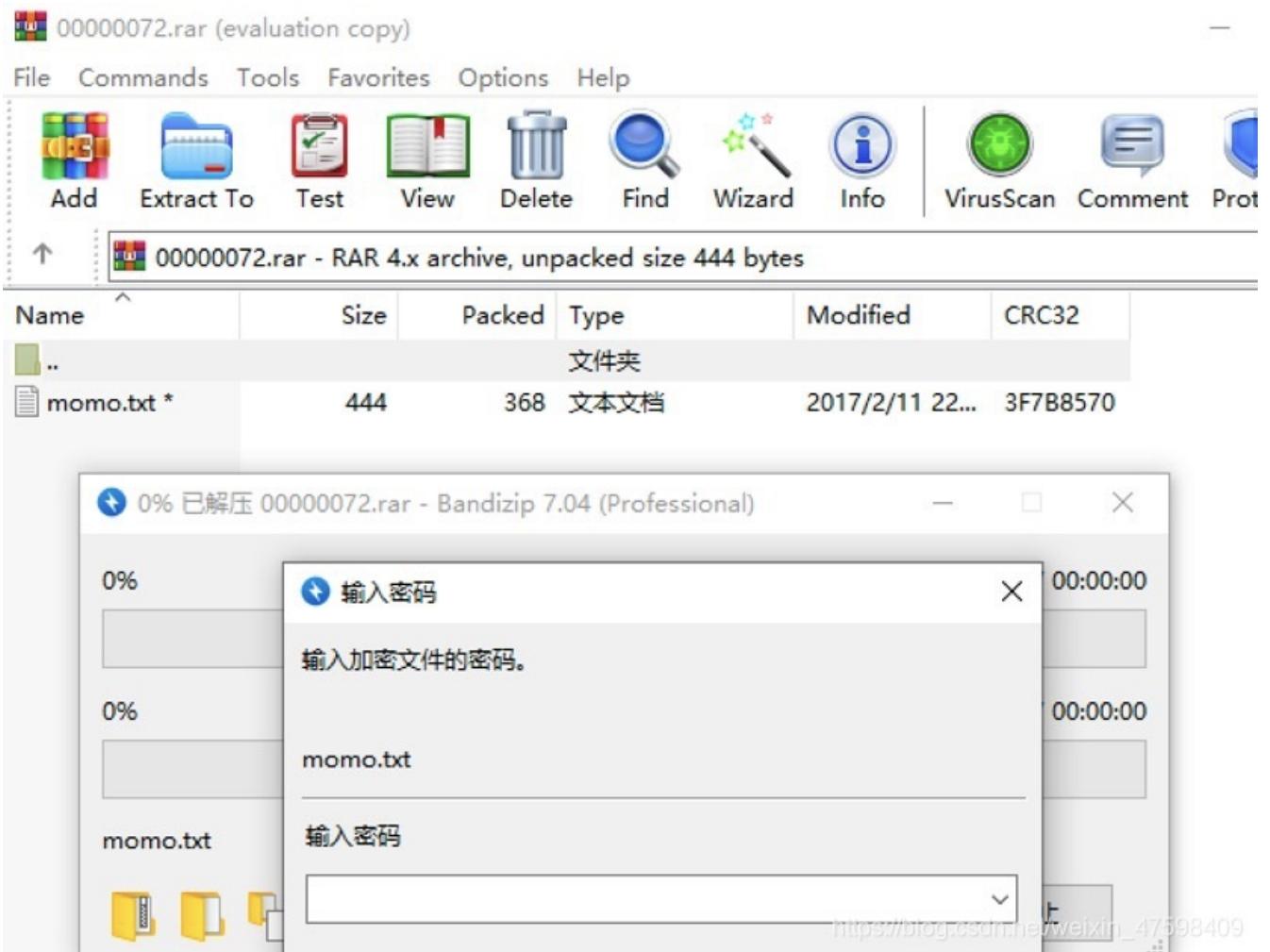
打开图片，这。。。。。。不敢多想其他，还是赶紧把它做出来吧

```
[root@root~14:23:07~Tony]
└─# binwalk momo.jpg
DECIMAL      HEXADECIMAL  DESCRIPTION
-----
0            0x0         JPEG image data, JFIF standard 1.01
37340       0x91DC      RAR archive data, version 4.x, first volume type: MAIN_HEAD

[root@root~14:23:11~Tony]
└─# foremost momo.jpg
Processing: momo.jpg
|*|

[root@root~14:23:20~Tony]
└─#
```

用binwalk查看图片有一个rar文件并用foremost把它分离



居然有密码,爆破密码没啥用，然后仔细看了一下图片有一句话，试试这句话是不是密码，结果居然成功了。解压密码竟然是图片中的“喜欢我吗。”

> 00000072

名称	修改日期	类型	大小
momo.txt	2017/2/11 22:50	文本文档	1 KB

momo.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

嘟嘟嘟嘟

士兵：报告首长！已截获纳粹的加密电报！

首长：拿来看看

电报内容：

.../-/-/..-/-.../-..-/-..-/-./-/-..-/-.../-..-/-..-/-.../-/-/.../-/.../-..-/-..-/-.../-..-/-..-/-...-

首长：我操你在逗我吗？你确定是他们纳粹发的吗？

士兵：难道我弄错了？哦。。。等等是这一条

内容：http://c.bugku.com/U2FsdGVkX18tl8Yi7FaGiv6jK1SBxKD30eYb52onYe0=  
AES Key： @#@#¥%..... ¥ ¥%%.....&¥

士兵：二维码真的扫不出来吗？？肯定可以扫出来

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

里面有一个文本，文本内容就是这样

摩斯密码加密解密

1 .../-/-/..-/-.../-..-/-..-/-./-/-..-/-.../-..-/-..-/-.../-/-/.../-/.../-..-/-..-/-.../-..-/-..-/-...-

生成摩斯密码 解密摩斯密码 交换内容 清空 下载加密/解密代码 复制加密/解密代码

1 HTTP://ENCODE.CHAHUO.COM/

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

发现有摩斯密码，直接解码，得出的字符好像是一条网址

HTTP://ENCODE.CHAHUO.COM/



打开网站居然是一个在线加密解密网站，收藏一波

把网址后面的base64作为密文粘进去，填上key，可解出明文：（注意：这个是AES解密）



# 在线加密解密 encode & decode

加密前字符串

U2FsdGVkX18tI8Yi7FaGiv6jK1SBxKD30eYb52onYe0=

密钥

@#@#¥%..... ¥ ¥%%.....&¥

SHA1

SHA224

SHA256

SHA384

SHA512

UrlEncode

UrlDecode

AES加密

**AES解密**

TripleDES解密

base64加密

base64解密

结果

momoj2j.png

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

将解出的明文和网址前半部分网址合起来  
即<http://c.bugku.com/momoj2j.png>

得到一个图片链接，下载后发现是个反色的二维码：



使用stegsolve取反后用QR Research可扫出flag:





[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

QR Research

文件(F) 工具(T) 帮助(H)

Folder icon, Save icon (A:), Monitor icon, Mouse icon, Question mark icon, Camera icon, QR icon, Paintbrush icon

纠错等级: H(30%)  
掩码: Auto  
版本: Auto  
尺寸: 4

已解码数据 1:  
位置:(6.7,4.0)-(697.4,4.2)-(6.7,693.4)-(699.4,693.5)  
颜色正常,正像  
版本:2

纠错等级:H,掩码:6

内容:

KEY{nitmzhen6}



[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

KEY{nitmzhen6}

## Bugku-MISC-就五层你能解开吗

Challenge

361 Solves



# 就五层你能解开吗

## 300

链接: <http://pan.baidu.com/s/1i4TQoz7> 密码: w65m

提示: 第一层: CRC32 碰撞

第二层: 维吉尼亚密码

第三层: sha1 碰撞

第四层: md5 相同文件不同

第五层: RSA

Flag

Submit

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

下载附件

Challenges: Cryptography 500.7z

保存到网盘

下载(247KB)

保存到手机

举报

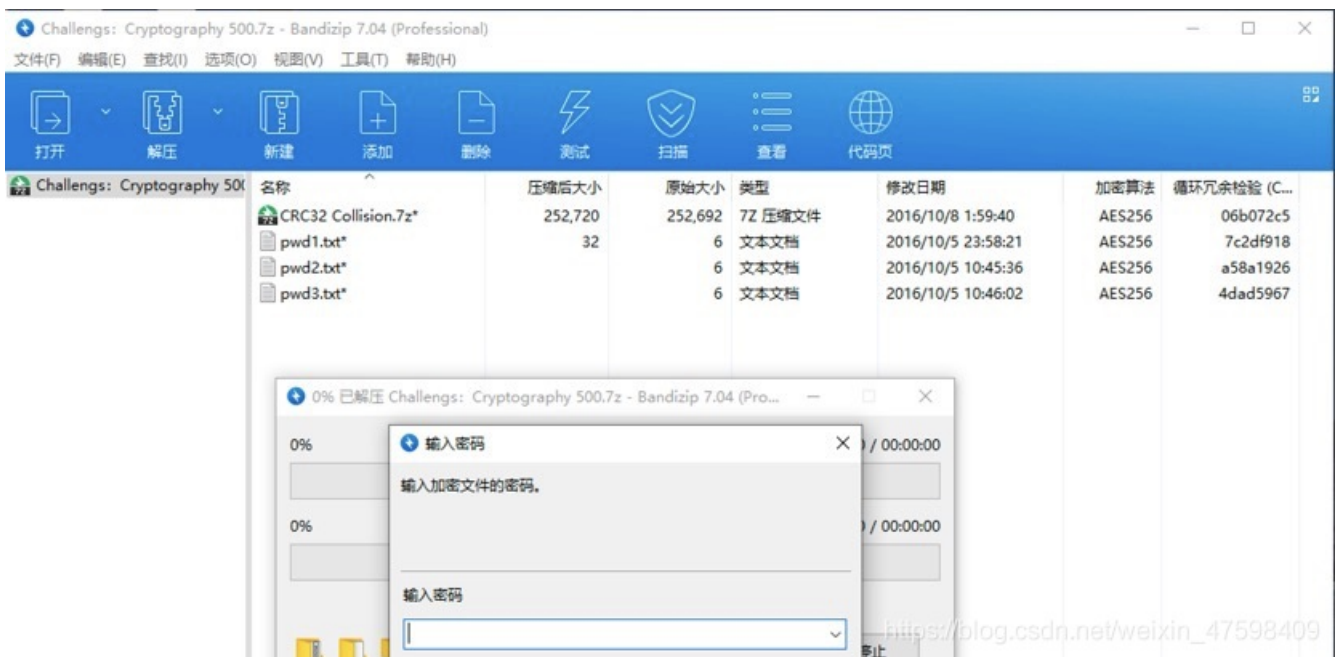
生成链接

© 2017-03-18 15:59 生成时间: 永久有效





解压居然要密码



然后根据题目的指示

提示：第一层：CRC32 碰撞

第二层：维吉尼亚密码

第三层：sha1 碰撞

第四层：md5 相同文件不同

第五层：RSA

先直接用CRC32碰撞脚本跑一波

脚本传送门<https://github.com/theonlypwner/crc32>

依次碰撞三个txt文件

```
[root@root~18:38:55~Tony]
└─┬─> ~/桌面/crc32# python crc32.py reverse 0x7C2DF918
4 bytes: {0x1c, 0x00, 0x1c, 0xa1}
verification checksum: 0x7c2df918 (OK)
```



```
alternative: VQT_ls (OK)
alternative: X28BT9 (OK)
alternative: _GLQzV (OK)
alternative: goMEPt (OK)
alternative: nyUKFQ (OK)
alternative: t_s4f3 (OK)
alternative: xQxVkx (OK)
alternative: yQ9gpa (OK)
```

test\_data.py

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

从结果上猜测最有可能的组合是: `_CRC32_i5_n0t_s4f3`

使用这个密码成功解压7z, 再解压里面的7z, 看到一个keys.txt, 里面有10000个密钥, 一个ciphertext.txt, 里面有一段密文。还有个tips.txt:

> Challenges: Cryptography 500 > CRC32 Collision

名称	修改日期	类型	大小
 ciphertext.txt	2016/10/6 23:21	文本文档	1 KB
 Find password.7z	2016/10/8 1:59	7Z 压缩文件	3 KB
 keys.txt	2016/10/7 2:25	文本文档	411 KB
 tips.txt	2016/10/7 3:37	文本文档	1 KB

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

```
keys.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
ZVVOKMTHJNWFPFRLDWGWOPMDHMDDDMSHWJMNNKXZ
ZSWKILXTWXOTAEISBTOBMRWBCODLSAOEHLKDGWDB
KAMQYTOMBEOGOIMROKOTXPXUTOREHDHLDIWCVPY
ITBENEXUYTDZPXAWIWIADYNMDGFEOSLCYMYXSKP
QFLXKEBGTEFGAISNRPUJJIORMICAFPDDQPKXVJG
OBQPOELJUZFLCGFPAHEYISILNKVIOCLQGLBFNYT
XCVHPWOQZRELISPTILVLLHRLUDFMMRSMCAUBVRD
CJGVUAQWHEYCIURHDCAOBVCDSVYTRZSPEFATZRG
FNNTXXGRMYTFPPQAKKDIISDEHBQGMIAAMJTIRMAO
NPKTNZDGGJEAHRVMANIJTZEBJVWUAWTSKVBUFPQ
NXQRVPJPBAMVTFJUTBFKSTYZJZDTNGXHCJPIHBDZ
SHINMBHZYLJELSOHIBBWZMPOQTEWYFQQXPSNNIZW
WMVSZTKFGGWAEGUMMKNLWRXBMGYFMZPJVFETHPTL
DBTWAQAAAYVGTLLILUEOVBNJUTSHHLVJRIAWJQLMW
WAPAAACVIGCBMNYIOLZNAKOEORLQFLNYDDJOVFKBG
WVCJCTLXGKSEJBMFLXHAUTNNUDLNWAPUDYOEOR
VYLNBTBDKDEXRBXGRKQFLRVBCZTYKDGLVMCITQP
LSOAMNTLIOLRRROIWHTAXVNVCKQAKFWGAJKEHPOL
XMFFUYVCRQHXEVBNRTEJHQAOGEDPTIEVTMITBNSP
BHSBMCFLHBUKSRDLZPMDQPTFKCECDJUOFHREDMWX
PDTPCQCDVFFTGRDDWQNEYQWGRWODHPEUKXIIYKMF
GWUDVMWVGNAGHEKVMHFEKFZPSGMRBRHWISKLZFRE
SFTUUBRUPFLLEZANYORAFYLDEFBNNXCYKZPXNNSO
```

```
ciphertext.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
rla xymijgpf ppsoto wq u nncwel ff tfqlgnxwzz sgnlwduzmy vcyg ib bhfbe u tnaxua ff satzmpibf vszqen eyvlatq cnzhk dk hfy mnc
```

tips.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

你知道维吉尼亚密码吗?  
我们给了keys.txt, 唯一的密钥就在其中, 那么解密ciphertext.txt里的密文吧!  
解压密码就在明文里, 祝你好运!

Do you know the Vigenère Ciphers?  
We gave the keys.txt, Only have a key in it, So decrypts ciphertext.txt!  
Unzip Password in plaintext, good luck to you!

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

写个python批量解密维吉尼亚密码:

```
from pycipher import Vigenere
keys = []
cipher = 'rla xymijgpf ppsoto wq u nncwel ff tfqlgnxwzz sgnlwduzmy vcyg ib bhfbe u tnaxua
ff satzmpibf vszqen eyvlatq cnzhk dk hfy mnciuzj ou s yygusfp bl dq e okcvpa hmsz vi
wdimyfqqjqubzc hmpmbgxifbgi qs lciyaktb jf clntkspy drywuz wucfm'
with open('./keys.txt', 'r') as f:
    keys = f.readlines()
plain = ''
with open('./plain.txt', 'w') as f:
    for i in keys:
        i=i.replace('\n','')
        plain = Vigenere(i).decipher(cipher)
        f.write(plain+'\n')
```



```
/root/桌面/Challenges : Cryptography 500/CRC32 Collision/plain.txt - Mousepad
文件(F) 编辑(E) 搜索(S) 视图(V) 文档(D) 帮助(H)
警告：您正在使用 root 账户，操作不当可能会损害您的系统。
|SQF JOAPXCXCJKANXILAKYZYQTXZCCQTYEKEL JMPVHOQBPNTRMPCBTWKWCFYFYOUNOCCPOBSQGWOSLVARJSQOJSTKDGIKCTSHI
STENQBLQKSRWPOGBNDCTBWGVCXCUBFCHZCNTTDPFOTATMOPFZFKNIXTPEIQTHWEWSMCUAAFVFEYZZPYRMRUQKOHXTOUZDCLJC
HLOHATUXFLLJBKCCAMCBQYFCLXOBMCJABKPAXEDIDLKNWGYMUYEAUTPQRRQBWYDDBMOBLXMOHMAFDADBGEBOACXOSMIHLZYMWD)
JSZTLILPIWCQAVOXGARMNKEJSIZAPRYAEPLYCHIRFSVZHVPEXJVHTEBLXFFMTKCKIXZAOMBOKRWDIDIKILMALROGVXZOKKKKCI
BGP AOIHDNLAJPKWGXLWLEGUINZXDATABIDXIMCEJAXGLGKVLSCYTAITJUOMKLGKSDOXDSVEWJZTRIAJTAZCHORKUAMXHZSQTS
DKKIKIXAMQAENMJEOPMWFVUOTYVVKLROAQHMVMUNZKGGVBPBDTVGVVWFUAWLVSPJNVKMRWJXAAIUGCDXPKRARZRXCXBFOK/
UJFQJQUTHYBEHAZBVOFZCCVFTRCAHTZTULXCYEBDQJBWFDYIWLUVAJMPMTTIZICTGJGCAGOGCHANIHEABWTCXJZXKUZYCRPVLRL
PCUCFSITKIBRNKUCHTOUZHUBTKHAGZTRJSWGABALCQIBFMIZVUIGTHQYUCFMFVRIKHZBCHXLDBMWBTOVHJFBLKXJYPXZQMDI
MYXEBPCSURMKADYEMNMVFVZSXXKZHXQLUEEOINSSIYTKXCGHJEFBTMLHVRBMLVXTNZPZGSTZAGPTOJSLLRKLBYPFHJHSJKSUDA)
EWQELNFDXJBPIBTHOJILUOYVYVQVJZFUODCVFUDQAWMKHAJSMWUGBKGVFOWLAOWWL FVJYAXGUFNALQDJDPLDLWIUKKYNSSQPRKI
EOKGDZXUFPTUWNFZVVLKVUEXVMCMGZTEEEIOSYPHAOGMKZDJUCMLPWSNMAZKBUCYLBCMFUWSKGTUYUPADHXNDGCLSQSGCBJRC
ZESKMLBKIEWLEAAMGVYOBNIOSBJVAAVJYFJMRTKVEOQIYFZXRPCXJNAXADYUBLJEHBJUVDJPAQOSNTDYXFRMUEBVFJOPWVWC
VZFFZTYEAJJPLMUHCMDJRWFVFSFHAHGBCLITDSKZVRZBLVGCTPWCGEVHVTRRDXWDWIUHAGBEQRKEIYGZOUSSMZCBVKNKGNVBQRY)
OKHBYWIJGRKJWHGIUSCZMATCLTYIYKHUYNBNJOGKKKDHUJMYVEDAPQTLXQZSARDBIYYHFKEIPMSPKGDNDUIYFLATSHHGWSQI
VLLXYKNBANEDCUGFDXDUDZYINAPAISSIDEJBUPRARLHDUXRQPAUVDTTUCRUJZWDPPAHNVWJGUGALRTUEYEYTSNOBBMJCPZJ
VQYOWTXMAFNLGRCDZJUTUPJKSCUGJQWMMKZIVLMPRQUUSGBBPSGCZAPCEUXZUNKAHCUFETKSMKNBHMUIJLVWCADNGKJQGYZYI
WNPXKTHGWMBSYRRNXMAPCWHVCMHJCKALBVOGJJRSNLQTGLVLZUJRAEBOROIWFWBSMHIXNORDGTMFJKVGCRCXCKXJNYQIGNXT)
GTMXZPYBYUYBALSPXUQSHJCBPFJAUFGENSSKEVCTIDIMTNNONPRKNZJULUWSFKSQPFIVXTMGYXYGEOFMZEMISPLCRWIQWVO/
UZVSEONHPZYSLXNGXDLLGXCIYHCQAXMQNBPDYMARQZRYABRWEMRJEGAUIZLXAJWPCZSPETDAGTAKETSIZEXQYQVQMMID)
OSTWUKDYZOLFXPLTGHFXYJDLJBDWVYRGGGJWVJMSFCTYHNOBFWQVWGMSPDYHUKYRDPDZLHTBYCJWCRVYDMTGDMDTDMHSLI
```

从得到的10000个明文中，搜索“Vigenere”，找到唯一一条有意义的明文：

```
MIYBNKVESATVVONOHEHSHASLCQRMDAZBETEERZPSIIUHJXZTHNMMOXACFMSDNAQMSFRMCVCPKVPJXVPLLBLINTYVFBQTFDJYK!
THEVIGENERECIPHERISAMETHODOFENCRYPTINGALPHABETICTEXTBYUSINGASERIESOFDIFFERENTCAESARCIPHERSBASEDON1
ELSIKUHYVTRMHHKXQVOCDAVPSBXPJDNKQVSTALOOEYWNIMIFOUYOVEVUCBDQKPSAKADPFYPWRSMDEFPIITXYLWQPWWDUTE
x 查找(N) : Vigenere 下一个 上一个 全部高亮 区分大小写(C)
```

```
THEVIGENERECIPHERISAMETHODOFENCRYPTINGALPHABETICTEXTBYUSINGASERIESOFDIFFERENTCAESARCIPHERSBASEDONTHELETTERSOF AKEYWORDITISASIMPLEFORMOFPOLYALPHABETICSUBSTITUTIONSOPASSWORDISVIGENERECIPHERFUNNY
```

从最后可以得到下一个压缩包的密码是“vigenere cipher funny”，需要全部改为小写。

名称	修改日期	类型
 Easy SHA1.7z	2016/10/8 1:59	7Z 压缩
 U need unzip password.txt	2016/10/7 3:26	文本文档

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

解压得到一个加密7z和一个txt:

U need unzip password.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

恭喜!

现在我们遇到一个问题,我们有一个zip文件,但我们不知道完整的解压密码。  
幸好我们知道解压密码的一部分sha1值。  
你能帮我们找到的密码吗?

不完整的密码: "\*7\*5-\*4\*3?" \*代表可打印字符

不完整的sha1: "619c20c\*a4de755\*9be9a8b\*b7cbfa5\*e8b4365\*" \*代表可打印字符

人生苦短, 我用Python。

Congratulations!  
Now we run into a problem,We have a zip file, but we don't know the complete unzip password.  
Fortunately, we know that part of the unzip password of sha1 value.  
can you help us to find the password?

Incomplete password is "\*7\*5-\*4\*3?" \* in the range of ASCII printable characters

Incomplete sha1 is "619c20c\*a4de755\*9be9a8b\*b7cbfa5\*e8b4365\*" \* in the range of ASCII printable characters

Life is short, you need Python.

[https://blog.csdn.net/weixin\\_47598409](https://blog.csdn.net/weixin_47598409)

写个python跑一下:

```
import string
import hashlib
printable = string.printable
for k1 in printable:
    for k2 in printable:
        for k3 in printable:
            for k4 in printable:
                key = k1 + '7' + k2 + '5-' + k3 + '4' + k4 + '3?'
                sha1 = hashlib.sha1(key.encode('utf-8'))
                flag = sha1.hexdigest()
                if '619c20c' and 'a4de755' and '9be9a8b' and 'b7cbfa5' and 'e8b4365' in flag:
                    print(key)
```

挨个试跑出来的结果:



```
[root@root~19:21:38~Tony]
~/桌面/Challenges : Cryptography 500/CRC32 Collision/Find password# python q.py
8725-{4}3?
3?B5-J4
k7T5-)4l3?
p7.5-"4'3?
F7M5-i4@3?
I7~5-s4F3?
*705-q4w3?
```

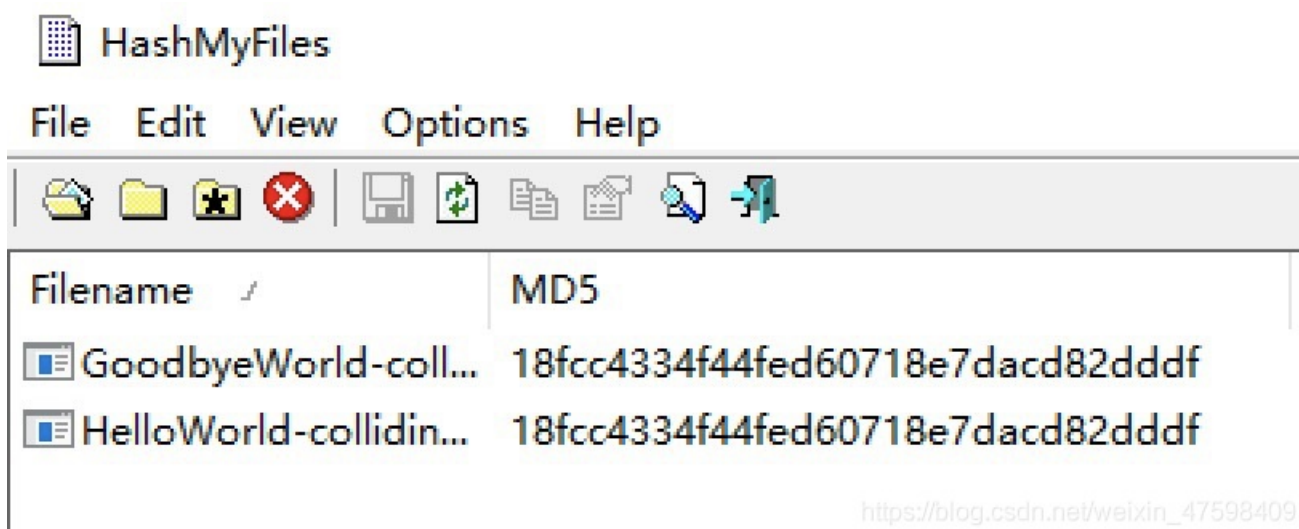
逐个测试最后发现“I7~5-s4F3?”是真正的密码。解压后看到一个7z和一个txt:

```
/root/桌面/Challenges : Cryptography 500/CRC32 Collision/Find password/Easy SHA
文件(E) 编辑(E) 搜索(S) 视图(V) 文档(D) 帮助(H)
警告：您正在使用 root 账户，操作不当可能...

Hello World ;- )
MD5校验真的安全吗？
有没有两个不同的程序MD5却相同呢？
如果有的话另一个程序输出是什么呢？
解压密码为单行输出结果。

Hello World ;- )
MD5 check is really safe?
There are two different procedures MD5 is the same?
If so what is the output of another program?
The decompression password is a single-line output.
https://blog.csdn.net/weixin_47598409
```

然后从网上找到了这样的两个软件:



输出分别是:

```
C:\Users\Administrator\Desktop\qq>HelloWorld-colliding.exe
Hello World ;- )
```





