whaleCTF MISC_Writeup(姿势大全)

原创

Pad0y 于 2019-04-21 16:47:34 发布 1876 收藏 6

分类专栏： WhaleCTF Learning 文章标签： Writeup 隐写 MISC

本文链接：https://blog.csdn.net/qq_34356800/article/details/83753759


WhaleCTF 同时被 2 个专栏收录

3 篇文章 0 订阅

订阅专栏


Learning

4 篇文章 0 订阅

订阅专栏

这两天抽空把whaleCTF的MISC部分的题目整理了下完整的Writeup，时间比较急写得简陋了些，如有错误欢迎各位师傅指正，持续更新~~

Find

直接上stegsolve神器，会发现一张反色的二维码，保存下来再反色即可，得到flag

被我吃了

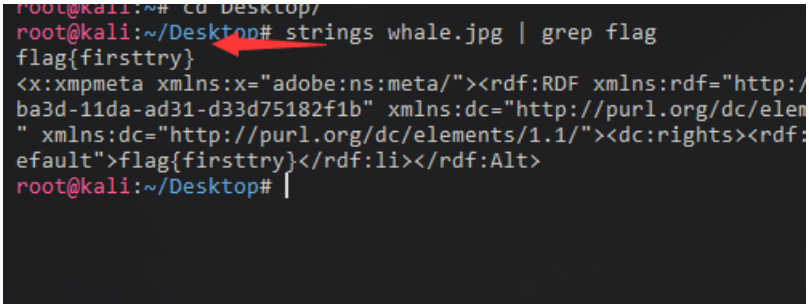
winhex下可以看到图片包含了一个zip，分离至新文件保存为zip即可得到flag

合体鲸鱼

和上一题同一个套路，jpg包含两张图片，分离即可

亚种

各种姿势检测下也没检测出什么，直接上strings大法，strings whale.jpg | grep flag



下雨天

分析下文件类型是一张gif，改后缀丢到gif分离器（stegsolve神器也可以），在第五帧得到flag(妹砸好评~)

这是什么

在winhex下搜索FFD9，发现没有结尾，拉到结尾发现有字符串，unicode——>ascii, getflag!

```
CF 53 D6 AC 50 4B E5 E8  D9 26 26 26 23 31 30 32  ÏSÖ¬PKåèÙ&&&#102
3B 26 23 31 30 38 3B 26  23 39 37 3B 26 23 31 30  ;&#108;&#97;&#10
33 3B 26 23 31 32 33 3B  26 23 31 31 32 3B 26 23  3;&#123;&#112;&#
36 39 3B 26 23 35 31 3B  26 23 31 30 37 3B 26 23  69;&#51;&#107;&#
38 31 3B 26 23 31 32 32  3B 26 23 31 30 39 3B 26  81;&#122;&#109;&
23 39 37 3B 26 23 37 37  3B 26 23 37 38 3B 26 23  #97;&#77;&#78;&#
31 32 35 3B                                       125;
```

IHDR

firefox加载不出来，用py把图片下载下来(win10自带的edge可以加载出来)
根据题目提示，跟图片提示IHDR块有关。py下载脚本如下：

```
import requests
response = requests.get('http://whalectf.xin/files/850e2779d4afe36c2094333c56c0cf84/HARD.png')
cat_img = response.content
with open('HARD.png','wb') as f:
    f.write(cat_img)
```

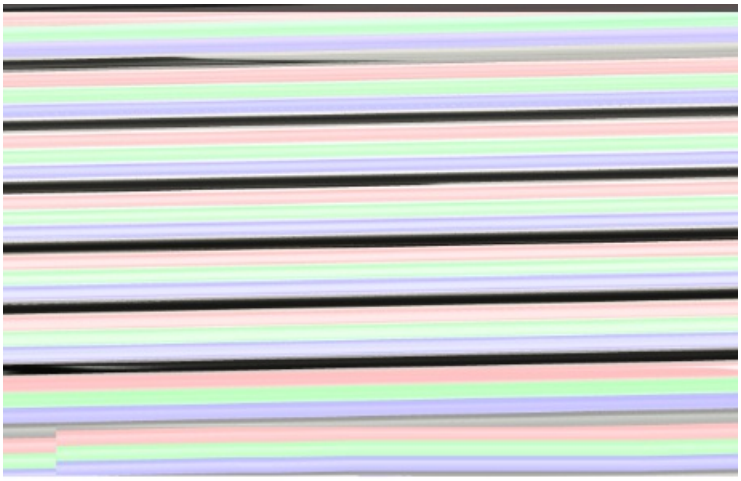
对一张正常的图片，通过修改其宽度或者高度隐藏信息，使计算出的CRC校验码与原图的CRC校验码不一致；windows的图片查看器会忽略错误的CRC校验码，因此会显示图片，但此时的图片已经是修改过的，所以会有显示不全或扭曲等情况，而Linux下的图片查看器不会忽略错误的CRC校验码，因此用Linux打开修改过宽或高的png图片时，会出现打不开的情况，也基本可以确定图片宽和高被动了手脚，话不多说，上脚本爆破。
参考链接如下：PNG(文件头IHEDR)图片隐写

```python
# -*- coding: utf-8 -*-
# __Author__:Pad0y
import binascii
import struct
crc32key = 0xCAF304E6  # CRC校验
for i in range(0, 65535):
 height = struct.pack('>i', i)
   data = '\x49\x48\x44\x52\x00\x00\x0C\xF0' + height + '\x08\x06\x00\x00\x00'
   crc32result = binascii.crc32(data) & 0xffffffff
   if crc32result == crc32key:
    print ''.join(map(lambda c: "%02X" % ord(c), height))
```

得到高度,修改即可,getflag!

```
In [33]: !python CRC_crack.py
00000F90
```

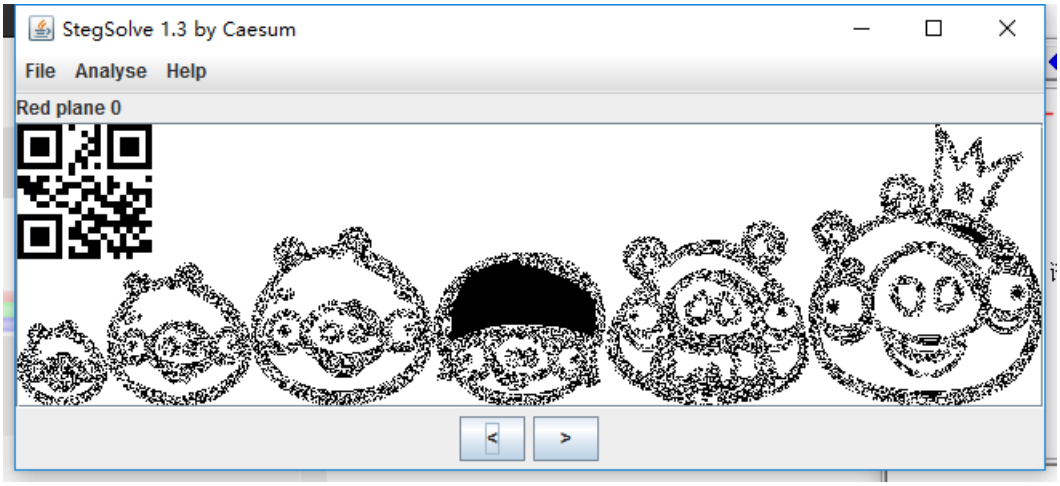
```
89 50 4E 47 0D 0A 1A 0A  00 00 00 0D 49 48 44 52  %PNG        IHDR
00 00 0C F0 00 00 0F 90  08 06 00 00 00 CA F3 04     ð          Êó
E6 00 00 00 01 73 52 47  42 00 AE CE 1C E9 00 00  æ    sRGB ®Î é
00 04 67 41 4D 41 00 00  B1 8F 0B FC 61 05 00 00    gAMA  ±  üa
00 09 70 48 59 73 00 00  0E C4 00 00 0E C4 01 95    pHYs   Ä   Ä •
2B 0E 1B 00 00 FF A5 49  44 41 54 78 5E A4 FD 69  +    ÿ¥IDATx^¤ýi
B3 A5 D9 79 9E 89 ED CC  3C 99 59 13 80 42 55 01  ³¥Ùyž‰íÌ<™Y €BU
C4 40 91 50 88 30 A2 21  93 AD B6 14 21 DA 41 49  Ä@'P^0¢!"-¶ !ÚAI
5F 1C EE 08 B7 BE 28 2C  3A 7A F0 07 FF 39 FD 00    î ·¾(,:zð ÿ9ý
```



FLAG{ihDR_ALSO_FUN}

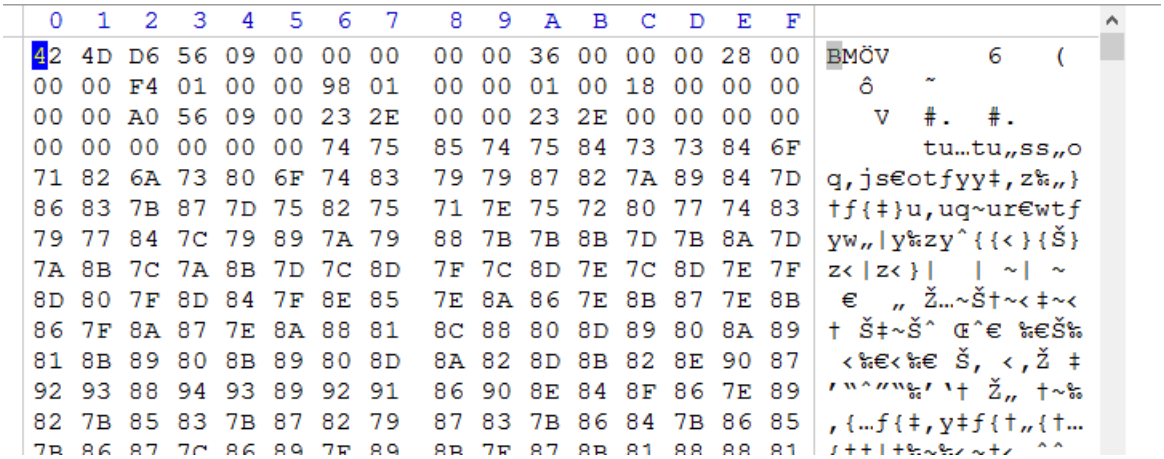
愤怒的小猪

丢到神器有个二维码，get!

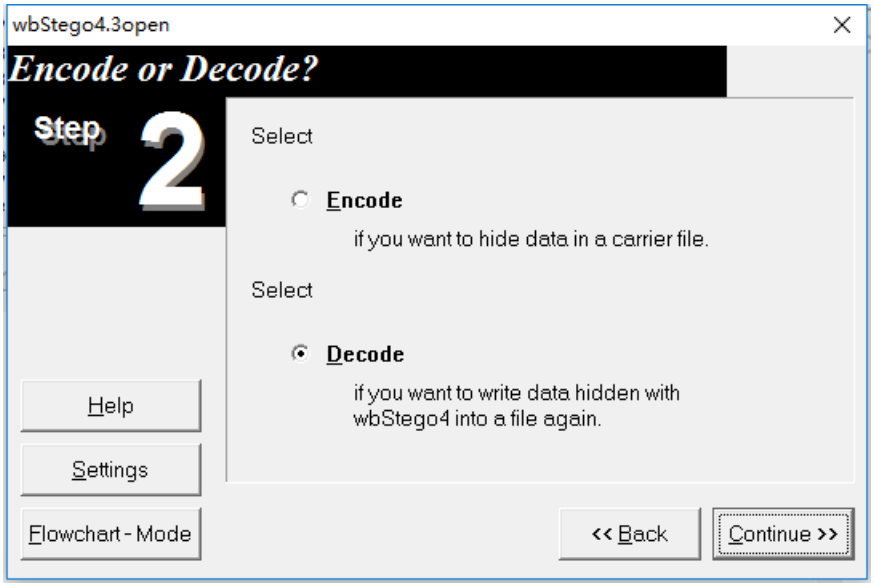


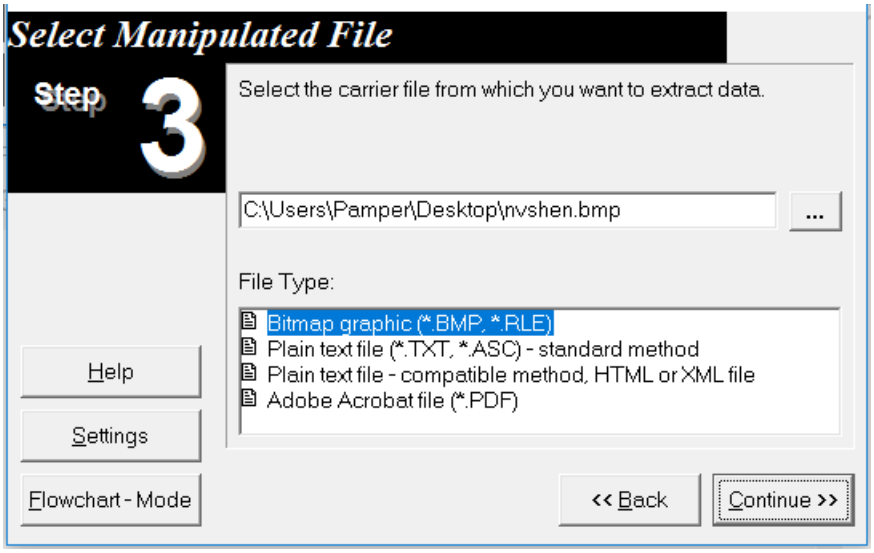
底片

丢到winhex，发现是bmp，比较少遇到bmp图片的隐写，骚分析一波，什么也得不到（--）目测是LSB隐写了，这里用wbs43open解密

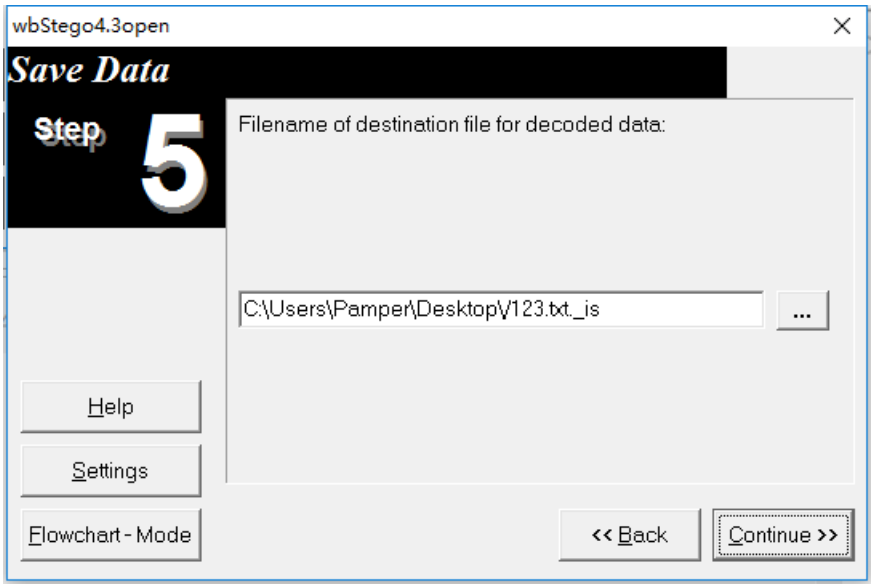


```
   0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
42 4D D6 56 09 00 00 00  00 00 36 00 00 00 28 00  BMÖV      6   (
00 00 F4 01 00 00 98 01  00 00 01 00 18 00 00 00    ô   ˜
00 00 A0 56 09 00 23 2E  00 00 23 2E 00 00 00 00     V  #.  #.
00 00 00 00 00 00 74 75  85 74 75 84 73 73 84 6F        tu…tu„ss„o
71 82 6A 73 80 6F 74 83  79 79 87 82 7A 89 84 7D  q‚js€otƒyy‡‚z‰„}
86 83 7B 87 7D 75 82 75  71 7E 75 72 80 77 74 83  †ƒ{‡}u‚uq~ur€wtƒ
79 77 84 7C 79 89 7A 79  88 7B 7B 8B 7D 7B 8A 7D  yw„|y‰zyˆ{{‹}{Š}
7A 8B 7C 7A 8B 7D 7C 8D  7F 7C 8D 7E 7C 8D 7E 7F  z‹|z‹}|  | ~| ~
8D 80 7F 8D 84 7F 8E 85  7E 8A 86 7E 8B 87 7E 8B   €  „ Ž…~Š†~‹‡~‹
86 7F 8A 87 7E 8A 88 81  8C 88 80 8D 89 80 8A 89  † Š‡~Šˆ Œˆ€ ‰€Š‰
81 8B 89 80 8B 89 80 8D  8A 82 8D 8B 82 8E 90 87   ‹‰€‹‰€ Š‚ ‹‚Ž ‡
92 93 88 94 93 89 92 91  86 90 8E 84 8F 86 7E 89  ’“ˆ”“‰’‘† Ž„ †~‰
82 7B 85 83 7B 87 82 79  87 83 7B 86 84 7B 86 85  ‚{…ƒ{‡‚y‡ƒ{†„{†…
```

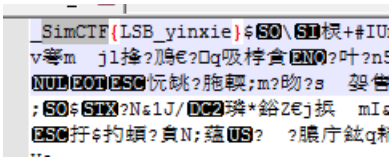




这里密码为空



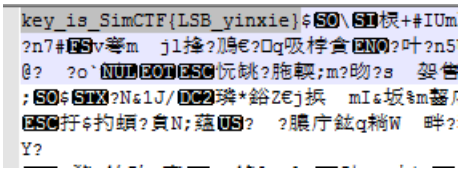
得到的文件用notepad++打开就能看到flag，这里有点邪门，怎么提交都显示错误（气哭.jpg）可能因为前面下划线还有东西，换个姿势继续日



由于图片是bmp类型，像素在图片中是倒叙存储，即以b,g,r的方式存储，而且当高度为正时图片中最后一行像素是存储在第一行的位置（对于正常二位坐标系，当以左下角为原点高度才会为正），用PIL的getdata读出的像素只是像素点在图片的相对位置，并非在文本文件的相对位置，所以以文本文件的方式读取图片

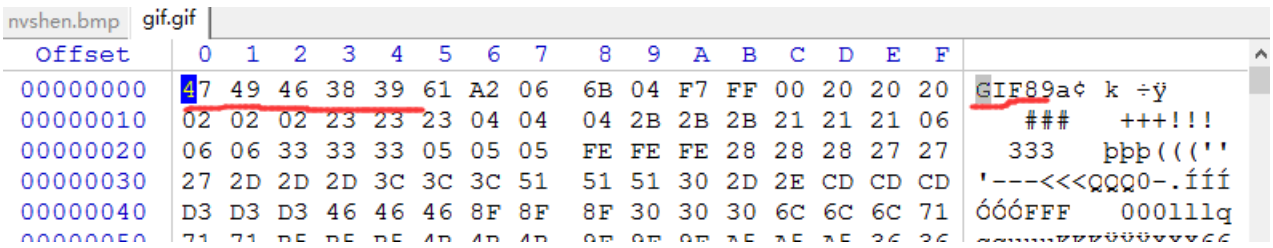
```
bmpfi=open("nvshen.bmp","rb")
bmpstr=bmpfi.read()
bmpfi.close()
bfOffBits=int(bmpstr[13:9:-1].encode("hex"),16)
str1=""
for j in xrange(bfOffBits,len(bmpstr)):
  str1+=bin(ord(bmpstr[j]))[-1]
i=0
lst=""
while i<len(str1):
  str2=str1[i:i+8]
  lst+=chr(int(str2,2))
  i=i+8

fi=open("flag",'wb')
fi.write(lst)
fi.close()
```

得到的flag再用notepad打开（会心一笑.jpg）
python大法好！



真是动图

打开链接图片又是无法显示的问题，把链接丢到能下载东西的软件就行了，得到一张损坏的GIF，缺少头部所致，补齐后分离每一帧图片，把灰色部分连起来base64解码即可Getflag。



模糊的图片

题目也提示了用ps去处理，仔细看人像背后有些字，这里需要锐化图片才能让图片清晰，电脑上没ps，于是用下在线版的ps，没法满足需求（蛋疼），上代码

```
>>> from PIL import Image
>>> img = Image.open('1.png')
>>> x = img.size[0]
>>> y = img.size[1]
>>> x,y
(399,378)
>>> for i in range(x-2):
 for j in range(y-2):
  a = img.getpixel((i,j))[0] + img.getpixel((i,j))[1]+img.getpixel((i,j))[2]
  b = img.getpixel((i,j+1))[0] + img.getpixel((i,j+1))[1]+img.getpixel((i,j+1))[2]
  c = img.getpixel((i,j+2))[0] + img.getpixel((i,j+2))[1]+img.getpixel((i,j+2))[2]
  if(a>b and c>b) or (a<b and c <b):
   pass
  else:
   img.putpixel((i,j),(255,255,255))
>>> img.show()
```

错误压缩

似曾相识的题目，binwalk里可以看到0x15AFFB后面是zlib的压缩数据

```
root@kali:~/Desktop# binwalk sctf.png

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PNG image, 1000 x 562, 8-bit/color RGBA, non-inte
rlaced
91            0x5B            Zlib compressed data, compressed
1421307       0x15AFFB        Zlib compressed data, default compression
```

pngcheck下图片，可以看到正常的块的length是在65524的时候就满了，而倒数第二个IDAT块长度是45027，最后一个长度是138，很明显最后一个IDAT块是有问题的，因为他本来应该并入到倒数第二个未满的块里。

```
  zlib: deflated, 32K window, fast compression
  chunk IDAT at offset 0x10008, length 65524
  chunk IDAT at offset 0x20008, length 65524
  chunk IDAT at offset 0x30008, length 65524
  chunk IDAT at offset 0x40008, length 65524
  chunk IDAT at offset 0x50008, length 65524
  chunk IDAT at offset 0x60008, length 65524
  chunk IDAT at offset 0x70008, length 65524
  chunk IDAT at offset 0x80008, length 65524
  chunk IDAT at offset 0x90008, length 65524
  chunk IDAT at offset 0xa0008, length 65524
  chunk IDAT at offset 0xb0008, length 65524
  chunk IDAT at offset 0xc0008, length 65524
  chunk IDAT at offset 0xd0008, length 65524
  chunk IDAT at offset 0xe0008, length 65524
  chunk IDAT at offset 0xf0008, length 65524
  chunk IDAT at offset 0x100008, length 65524
  chunk IDAT at offset 0x110008, length 65524
  chunk IDAT at offset 0x120008, length 65524
  chunk IDAT at offset 0x130008, length 65524
  chunk IDAT at offset 0x140008, length 65524
  chunk IDAT at offset 0x150008, length 45027
  chunk IDAT at offset 0x15aff7, length 138
  chunk IEND at offset 0x15b08d, length 0
No errors detected in sctf.png (28 chunks, 36.8% compression).
```

```
0015AF70  2F 05 D0 73 C5 5C E6 62  3E F3 57 80 FE B4 20 49  / ÐsÅ\æb>óW€þ´ I
0015AF80  34 90 9B 26 A0 33 D2 9F  91 9D 9E D3 F3 79 9C C5  4 ›& 3Ò Ÿ ' žÓóyœÅ
0015AF90  D9 73 B2 10 67 F8 09 E8  13 E5 D9 62 3E 73 9E BF  Ùs² gø è åÙb>sž¿
0015AFA0  9C 31 3E CD 73 9A 13 D0  1F 0B A0 F3 B4 68 12 A0  œ1>Ísš Ð   ó´h
0015AFB0  4F 97 E6 D1 36 CF C6 74  7E 16 A6 B2 E8 F3 96 9A  C—æÑ6ÏÆt~ ¦²èó–š
0015AFC0  20 A0 2E 05 FA 44 C3 FF  2E D0 69 5B 0A A0 97 FF  . úDÃÿ.Ði[  —ÿ
0015AFD0  17 40 AF C3 48 6B A5 00  3A 4F ED 26 05 FA 54 63  @¯ÃHk¥ :Oí& úTc
0015AFE0  95 00 FA 54 0D 21 BD BA  02 FF 3F 01 E7 98 5E 68  • úT !½º ÿ? ç˜^h
0015AFF0  95 8F CD 00 00 00 8A 49  44 41 54 78 9C 5D 91 01  • Í   ŠIDATxœ]'
0015B000  12 80 40 08 02 BF 04 FF  FF 5C 75 29 4B 55 37 73  €@  ¿ ÿÿ\u)KU7s
0015B010  8A 21 A2 7D 1E 49 CF D1  7D B3 93 7A 92 E7 E6 03  Š!¢} IÏÑ}³"z'çæ
0015B020  88 0A 6D 48 51 00 90 1F  B0 41 01 53 35 0D E8 31  ^ mHQ   °A S5 è1
0015B030  12 EA 2D 51 C5 4C E2 E5  85 B1 5A 2F C7 8E 88 72  ê-QÅLâå…±Z/ÇŽ^r
0015B040  F5 1C 6F C1 88 18 82 F9  3D 37 2D EF 78 E6 65 B0  õ oÁ^ ,ù=7-ïxæe°
0015B050  C3 6C 52 96 22 A0 A4 55  88 13 88 33 A1 70 A2 07  ÃlR–" ¤U^ ^3¡p¢
0015B060  1D DC D1 82 19 DB 8C 0D  46 5D 8B 69 89 71 96 45  ÜÑ, ÛŒ F]‹i‰q–E
0015B070  ED 9C 11 C3 6A E3 AB DA  EF CF C0 AC F0 23 E7 7C  íœ Ãjã«ÚïÏÀ¬ð#ç|
0015B080  17 C7 89 76 67 D9 CF A5  A8 00 00 00 00 49 45 4E  Ç‰vgÙÏ¥¨    IEN
0015B090  44 AE 42 60 82                                    D®B`,
```

CRC校验码

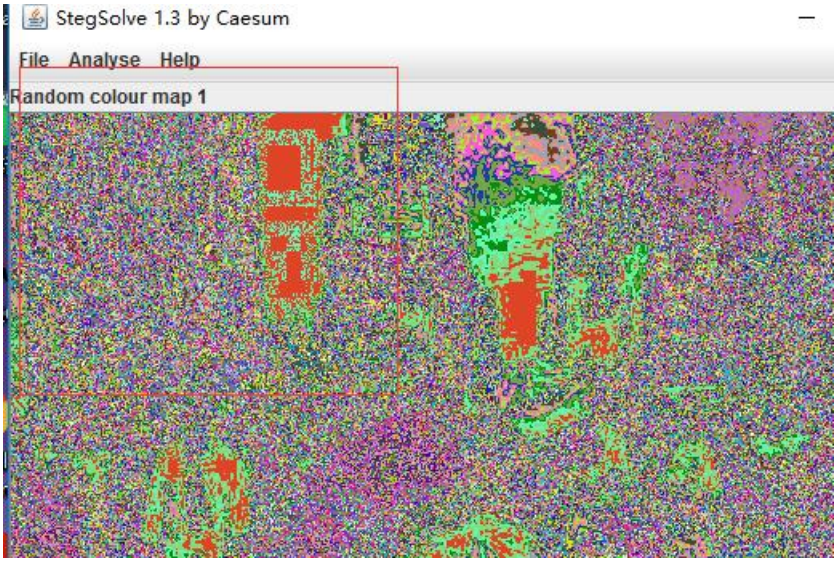
winhex可以看到最后四个字节是CRC校验码，所以有效数据是789C-7667范围内的字节，写个脚本提取出来

```python
# -*- coding: utf-8 -*-
# __Author__:Pad0y
from PIL import Image
import zlib
img = open('sctf.png', 'rb').read()[0x15AFFB:0x15B085]  # 获取zlib压缩数据
data = zlib.decompress(img)
binstr = str(data,encoding='utf8')
# print(len(binstr))  长度是625，刚好可以绘制成 25*25的图片
pic = Image.new('RGB', (25, 25))  # 绘制一张25*25的图
i = 0
for y in range(0, 25):
    for x in range(0, 25):
        if binstr[i] == '1':
            pic.putpixel([x, y], (0, 0, 0))
        else:
            pic.putpixel([x, y], (255, 255, 255))
        i = i+1

pic.save('flag.png')
```

得到一张二维码，getflag！

最低位的亲吻

看题目是最低位，由此联想到LSB，先想办法获取最低位，再把获取结果表示出来。但是二进制最低位只可能是0或1，由0,1表示出来的像素点自然想到是二维码，LSB的套路。如何证明？丢到神器大概可以发现有个二维码的图片。

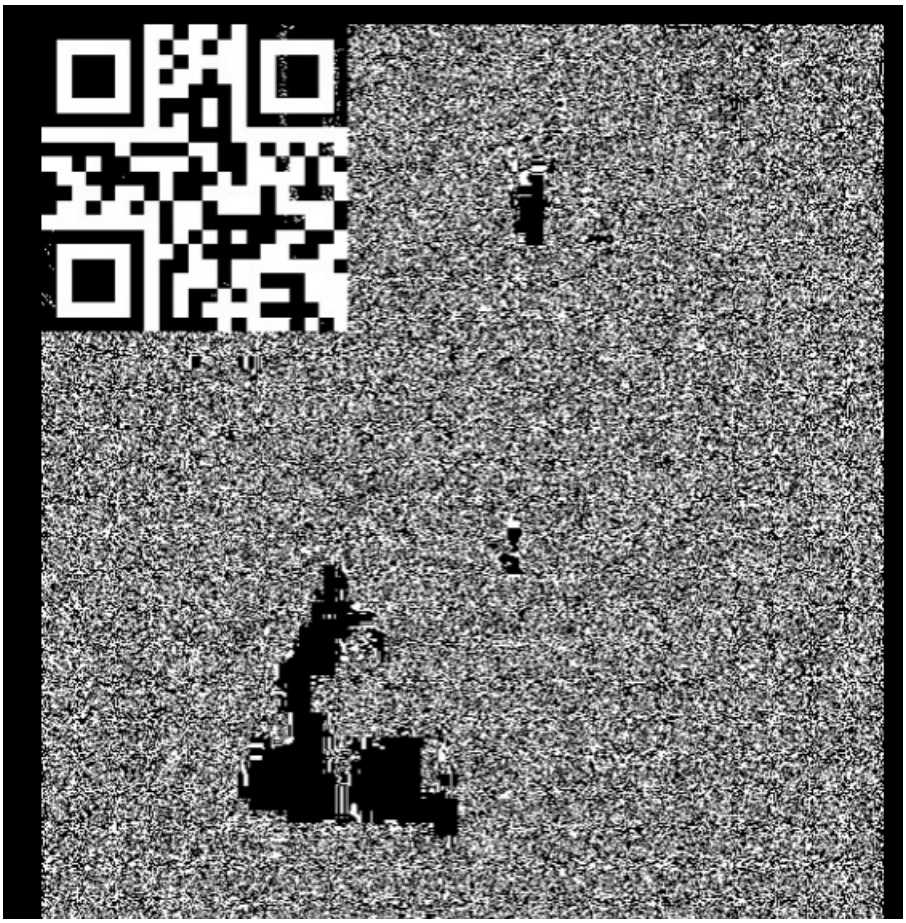


这题目骚姿势比较多，大佬们用matlab三行代码就解决了，也可以把这张图片转化为32bit的png图片（用画图工具另存为png就可以）在丢到stegsolve就能看到完整的二维码,扫一下得到flag

flag{i love u}

py实现如下

```
from PIL import Image
lena = Image.open('01.bmp')
im=Image.new('1',lena.size)
pixsels = lena.load()
width=lena.size[0]
height=lena.size[1]

for x in range(0,width):
  for y in range(0,height):
      g=pixsels[x,y]
      im.putpixel((x,y),int(bin(g)[-1]))


im.save("2.bmp")
```



舞会搭档

看了下到此为止只有7个人做出来，应该算是MISC部分最难的一题，上神器，太大无法加载出来，上脚本PIL库无法加载这么大的图片也报错，心态爆炸。一开始想着如何恢复看似二维码的图片(然而并不是)，还参考了如何用PS修复一张模糊的二维码？然并卵，想在一天内AK隐写题果然很蛋疼（还要算上写博客的时间）

```
In [3]: img = Image.open('Question1.jpg')
d:\program files\python36\lib\site-packages\PIL\Image.py:2546
5 pixels, could be decompression bomb DOS attack.
  DecompressionBombWarning)

In [4]:
```

直接丢到notepad，发现一些有趣的东西

此前一直没有注意题目的描述部分"教了李磊四个简单的动作：上、下、左、右",对应如下：

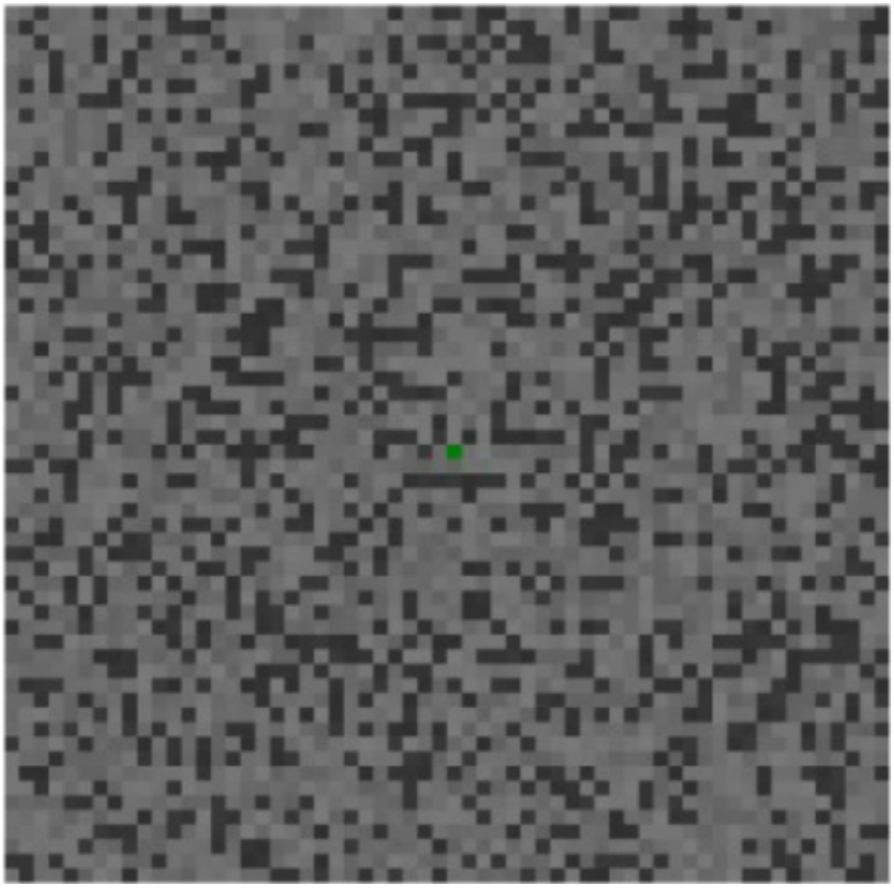
^[[A：上

^[[B：下

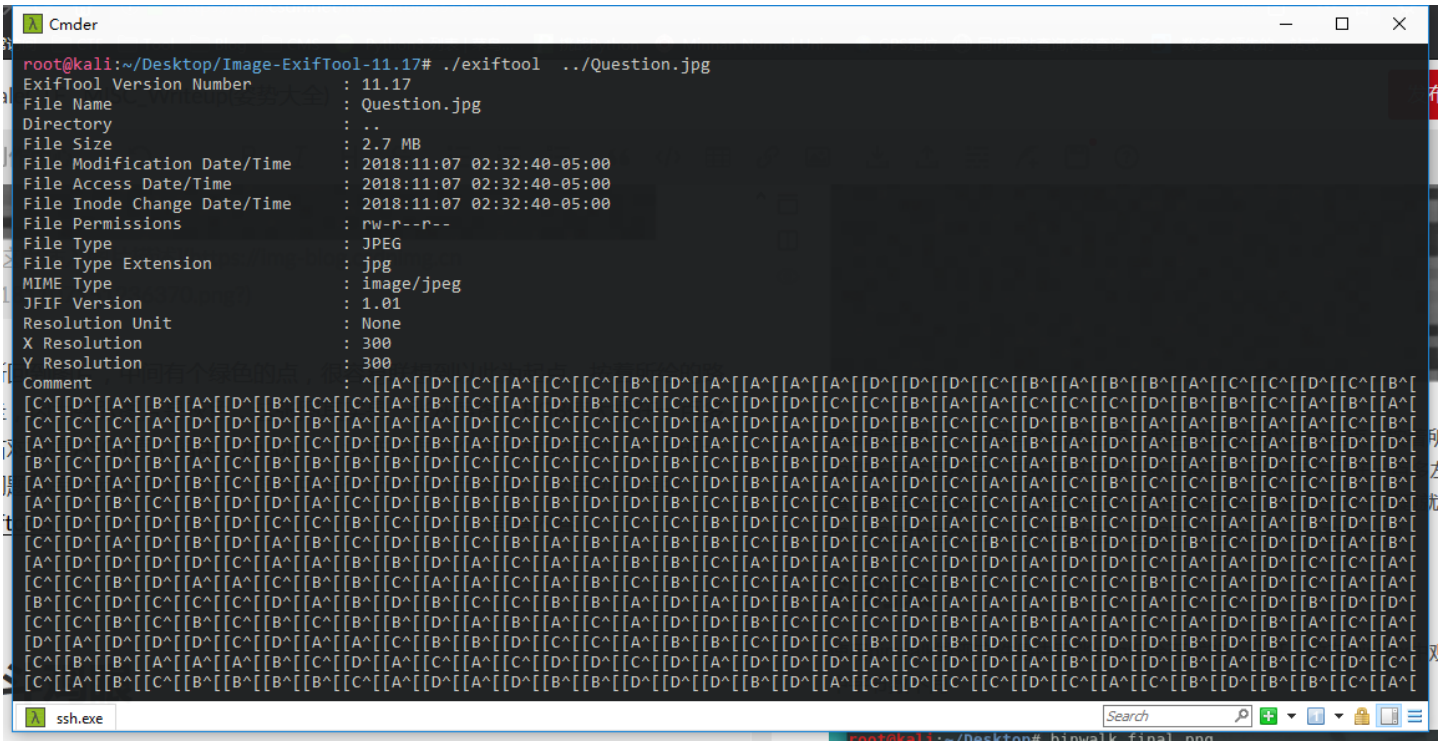
^[[C：右

^[[D：左

（觉得熟悉不，终端想拿到上一条命令按↑有时候就会出现^[[A）



重新回到图片，中间有个绿色的点，很容易联想到以此为起点，按着所给的路径走，因为整张图片是灰色，这张图其实是一张灰度图，RGB值大概在一百多左右，可以对应ASCII值（笑容逐渐荡漾），这样根据每个路径就能得到，剩下的一个问题就是移动时间和方向，图像中有个comment元数据，这里可以借助exiftools查看内容




```
# -*- coding: utf-8 -*-
# __Author__: Pad0y
from PIL import Image
import sys
```

```
import sys
comment = """
^[[A^[[D^[[C^[[A^[[C^[[C^[[B^[[D^[[A^[[A^[[A^[[A^[[D^[[D^[[D^[[C^[[B^[[A^[[B^[[B^[[A^[[C^[[C^[[D^[[C^[[B^[[C^[[D
^[[A^[[B^[[A^[[D^[[B^[[C^[[C^[[A^[[B^[[C^[[A^[[D^[[B^[[C^[[C^[[C^[[D^[[D^[[C^[[C^[[B^[[A^[[A^[[C^[[C^[[C^[[D^[[B
^[[B^[[C^[[A^[[B^[[A^[[C^[[C^[[C^[[A^[[D^[[D^[[D^[[B^[[A^[[A^[[A^[[D^[[C^[[C^[[C^[[C^[[D^[[A^[[D^[[A^[[D^[[D^[[B
^[[C^[[C^[[D^[[B^[[B^[[A^[[A^[[B^[[A^[[A^[[C^[[B^[[A^[[D^[[A^[[D^[[B^[[D^[[D^[[C^[[D^[[D^[[B^[[A^[[D^[[D^[[C^[[A
^[[D^[[A^[[C^[[A^[[A^[[B^[[B^[[C^[[A^[[B^[[A^[[D^[[A^[[B^[[C^[[A^[[B^[[D^[[D^[[B^[[C^[[D^[[B^[[B^[[B^[[B^[[A^[[B
^[[B^[[D^[[C^[[C^[[C^[[C^[[D^[[B^[[C^[[B^[[B^[[D^[[B^[[A^[[D^[[C^[[A^[[B^[[D^[[D^[[A^[[B^[[D^[[C^[[B^[[B^[[A^[[D
^[[A^[[D^[[B^[[C^[[B^[[A^[[D^[[D^[[D^[[B^[[D^[[B^[[C^[[D^[[C^[[D^[[B^[[A^[[A^[[A^[[D^[[C^[[A^[[A^[[C^[[B^[[C^[[C
^[[B^[[C^[[C^[[B^[[B^[[A^[[D^[[B^[[C^[[B^[[D^[[D^[[A^[[C^[[D^[[B^[[B^[[B^[[B^[[D^[[D^[[B^[[C^[[B^[[C^[[C^[[B^[[C
^[[C^[[C^[[A^[[C^[[C^[[A^[[C^[[C^[[B^[[D^[[C^[[D^[[D^[[D^[[D^[[D^[[B^[[D^[[C^[[C^[[B^[[C^[[D^[[B^[[D^[[C^[[C^[[C
^[[C^[[B^[[D^[[C^[[D^[[B^[[D^[[A^[[C^[[C^[[B^[[C^[[D^[[C^[[A^[[A^[[B^[[D^[[B^[[D^[[A^[[B^[[D^[[D^[[A^[[B^[[A^[[D
^[[D^[[D^[[D^[[C^[[A^[[A^[[B^[[B^[[D^[[A^[[C^[[A^[[A^[[B^[[B^[[C^[[A^[[D^[[A^[[B^[[C^[[D^[[D^[[A^[[D^[[D^[[C^[[A
^[[A^[[D^[[C^[[C^[[A^[[C^[[C^[[B^[[D^[[A^[[A^[[C^[[B^[[B^[[C^[[A^[[A^[[C^[[A^[[B^[[C^[[B^[[C^[[C^[[A^[[C^[[C^[[C
^[[B^[[C^[[C^[[C^[[C^[[B^[[C^[[A^[[D^[[C^[[A^[[A^[[B^[[C^[[D^[[C^[[C^[[C^[[D^[[A^[[B^[[D^[[B^[[C^[[B^[[C^[[B^[[A
^[[D^[[A^[[D^[[B^[[A^[[C^[[A^[[A^[[A^[[A^[[B^[[C^[[A^[[C^[[C^[[D^[[B^[[D^[[D^[[C^[[C^[[B^[[C^[[B^[[C^[[B^[[C^[[B
^[[B^[[D^[[A^[[B^[[A^[[C^[[A^[[D^[[D^[[B^[[C^[[C^[[C^[[D^[[B^[[A^[[B^[[A^[[A^[[C^[[A^[[D^[[B^[[A^[[C^[[A^[[D^[[A
^[[D^[[D^[[D^[[C^[[D^[[A^[[A^[[C^[[B^[[B^[[D^[[C^[[C^[[A^[[B^[[B^[[C^[[D^[[C^[[B^[[D^[[B^[[D^[[C^[[C^[[D^[[D^[[B
^[[D^[[B^[[C^[[A^[[A^[[C^[[B^[[B^[[A^[[A^[[A^[[B^[[C^[[D^[[A^[[C^[[A^[[C^[[D^[[D^[[C^[[D^[[A^[[C^[[A^[[C^[[A^[[C
^[[D^[[D^[[A^[[B^[[D^[[D^[[B^[[A^[[B^[[C^[[D^[[C^[[C^[[A^[[B^[[C^[[B^[[B^[[B^[[B^[[C^[[A^[[D^[[A^[[D^[[B^[[B^[[D
^[[D^[[D^[[B^[[D^[[A^[[C^[[D^[[C^[[C^[[D^[[C^[[A^[[C^[[B^[[D^[[B^[[B^[[C^[[A^[[B^[[A^[[C^[[D^[[D^[[D^[[C^[[C^[[D
^[[D^[[A^[[B^[[D^[[D^[[D^[[C^[[C^[[B^[[D^[[D^[[B^[[B^[[A^[[B^[[B^[[C^[[A^[[A^[[A^[[C^[[D^[[D^[[A^[[D^[[A^[[B^[[C
^[[C^[[C^[[B^[[D^[[D^[[D^[[D^[[C^[[D^[[D^[[B^[[C^[[D^[[B^[[B^[[C^[[D^[[B^[[C^[[C^[[D^[[B^[[D^[[C^[[A^[[C^[[C^[[D
^[[B^[[D^[[B^[[D^[[A^[[B^[[B^[[B^[[A^[[D^[[C^[[C^[[C^[[C^[[C^[[A^[[D^[[B^[[C^[[A^[[B^[[D^[[B^[[D^[[B^[[B^[[B^[[D
^[[C^[[B^[[B^[[B^[[C^[[B^[[A^[[D^[[C^[[C^[[A^[[D^[[A^[[B^[[A^[[D^[[D^[[B^[[A^[[D^[[B^[[C^[[B^[[A^[[D^[[B^[[C^[[D
^[[C^[[A^[[B^[[D^[[D^[[D^[[C^[[B^[[B^[[A^[[D^[[D^[[B^[[D^[[D^[[C^[[C^[[D^[[D^[[A^[[B^[[C^[[D^[[D^[[C^[[C^[[D^[[A
^[[C^[[A^[[C^[[A^[[D^[[B^[[C^[[A^[[C^[[B^[[C^[[B^[[A^[[D^[[B^[[D^[[A^[[D^[[C^[[A^[[B^[[B^[[D^[[C^[[A^[[C^[[A^[[D
^[[D^[[B^[[C^[[D^[[C^[[B^[[C^[[C^[[C^[[B^[[A^[[D^[[B^[[A^[[A^[[D^[[A^[[D^[[D^[[C^[[B^[[D^[[D^[[C^[[D^[[B^[[B^[[A
^[[A^[[C^[[A^[[A^[[A^[[A^[[D^[[C^[[D^[[A^[[B^[[C^[[A^[[A^[[C^[[D^[[C^[[D^[[C^[[D^[[A^[[C^[[B^[[C^[[D^[[C^[[B^[[A
^[[D^[[B^[[B^[[B^[[B^[[B^[[C^[[A^[[A^[[A^[[A^[[D^[[C^[[C^[[A^[[C^[[B^[[D^[[C^[[D^[[A^[[A^[[B^[[D^[[C^[[A^[[A^[[C
^[[B^[[D^[[B^[[C^[[D^[[D^[[C^[[B^[[C^[[A^[[D^[[A^[[D^[[A^[[B^[[D^[[D^[[B^[[A^[[D^[[A^[[D^[[C^[[A^[[C^[[A^[[C^[[A
^[[B^[[D^[[D^[[B^[[B^[[B^[[D^[[D^[[A^[[A^[[C^[[B^[[B^[[B^[[C^[[C^[[A^[[C^[[B^[[B^[[D^[[B^[[B^[[C^[[B^[[D^[[D^[[A
^[[A^[[B^[[D^[[C^[[C^[[C^[[C^[[B^[[A^[[C^[[C^[[C^[[D^[[D^[[D^[[B^[[C^[[D^[[B^[[D^[[D^[[B^[[B^[[D^[[C^[[C^[[D^[[D
^[[A^[[A^[[D^[[C^[[C^[[C^[[B^[[C^[[A^[[D^[[D^[[C^[[A^[[A^[[D^[[C^[[C^[[D^[[D^[[A^[[B^[[B^[[D^[[D^[[A^[[B^[[B^[[D
^[[C^[[B^[[D^[[A^[[B^[[C^[[C^[[B^[[D^[[C^[[C^[[C^[[C^[[C^[[C^[[D^[[A^[[B^[[D^[[A^[[A^[[B^[[D^[[C^[[B^[[D^[[D^[[A
^[[C^[[C^[[B^[[A^[[A^[[C^[[C^[[B^[[D^[[A^[[A^[[B^[[A^[[A^[[D^[[C^[[C^[[C^[[B^[[A^[[B^[[D^[[B^[[A^[[B^[[A^[[B^[[A
^[[B^[[A^[[A^[[A^[[B^[[D^[[D^[[D^[[B^[[C^[[D^[[C^[[A^[[D^[[B^[[A^[[A^[[D^[[A^[[D
"""
comment = comment.replace('^','').replace('[[','')
# A is up
# B is down
# C is right
# D is left

x, y = 4846 + 61, 6900 + 61
size = 122
image = Image.open('Question.jpg')
data = image.load()
string = []
for new_place in comment:
    # new_place = 'A'
    if ( new_place == 'A' ): y -= size
    if ( new_place == 'B' ): y += size
    if ( new_place == 'C' ): x += size
    if ( new_place == 'D' ): x -= size

    try:
        string.append(chr(data[x, y][0]))
        # print data[x,y]
```

```
    except:
        pass

# print len(string)
sys.stdout.write( "".join(string) )
image.close()
```

如果解出答案字符串大于32位，则提交32位md5值（小写），提交格式：flag{xxxx}

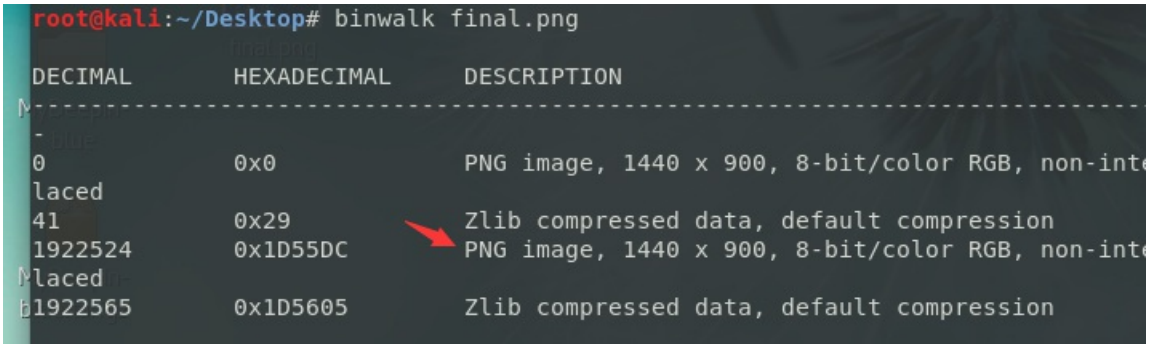
根据题目提示把得到的字符串用MD5加密就能得到flag了(记得转换成小写)

2019-01-13补充：对于小伙伴们的issues如下

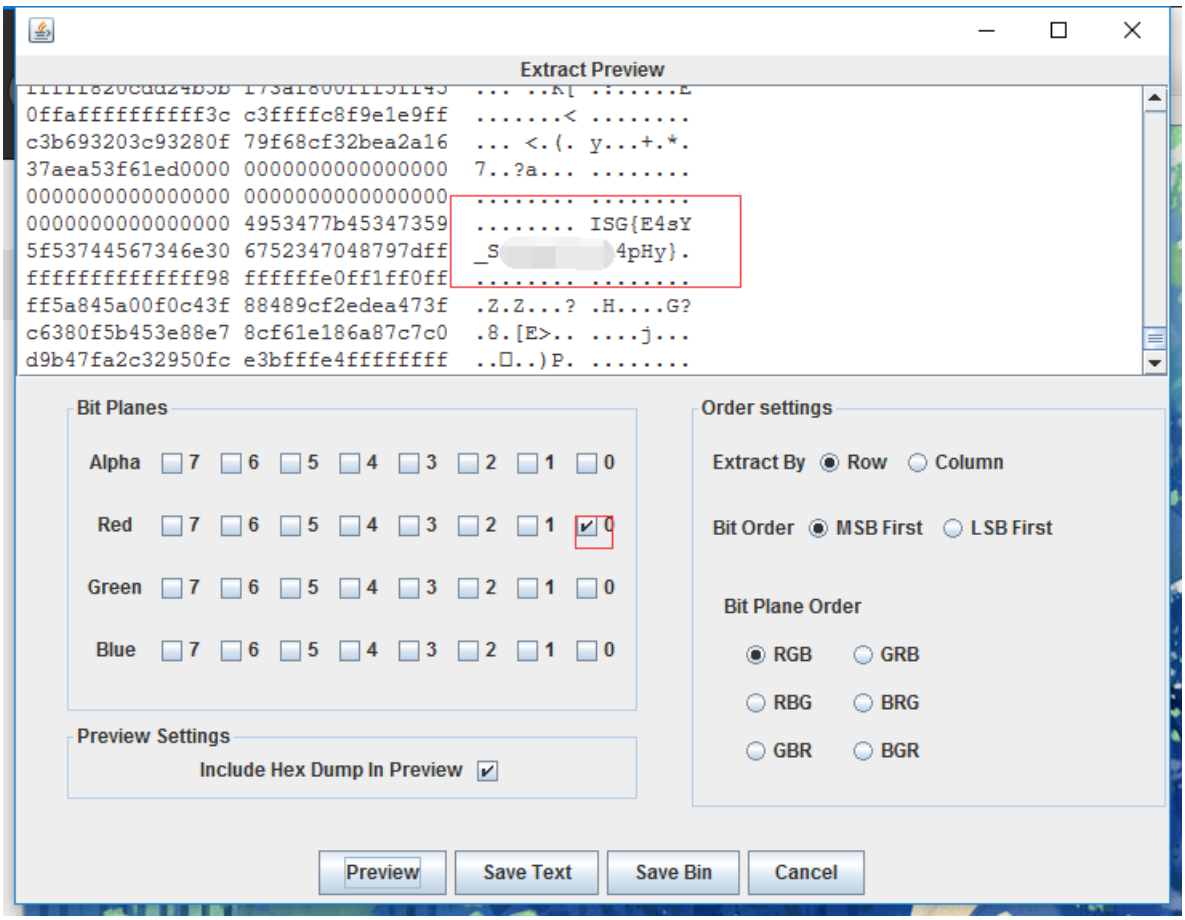
- Q1：结果为什么需要去掉最后一位？
  A：结果的每一位都对应一位comment元数据，提取出来的有1000个（直接对ABCD四个计数），取得的结果有1001位，故需要舍掉再进行md5
- Q2：^[[A这种是什么？
  A：这个是ascii控制码，以ESC起始为标记。ANSI的终端机标准里面，光标上、下、右、左移动的指令分别规定为<esc>[A，<esc>[B，<esc>[C，<esc>[D，其中方括号跟字母之间还可以插入一个数字来表示要移动几位，<esc>在ANSI标准里面排行第27（0x1B），是个不可见字符。

斗鸡眼

使用binwalk发现文件是由两张png组合而成的，分开，放在winhex中观察发现两个文件中字节数不同



分别是1d55db和1d5648于是将使用stegsolve对图片做减法，发现得到的图片最下角有一条红色的线，猜想相对大一些的图片可能是在结尾加入了密码字符而导致两个文件字节数不相同，将做完减法的图片放入stegsolve观察发现在Red plane 0处线短的很厉害，基本可以确定在图像结尾处添加了对r使用了低位加密的像素点，将大一点（1d5648）的图像放入stegsolve，用stegsolve的data extract选择red0得到结果观察字符串尾部，即可发现key



AK

隐写