CTFshow刷题日记-MISC-图片篇(下 24-51)文件结构与颜色通道

原创

OceanSec 于 2021-09-11 15:52:34 发布 773 收藏 9

分类专栏： # CTF 文章标签： python pycharm 爬虫 ctf

本文链接：https://blog.csdn.net/q20010619/article/details/120238821

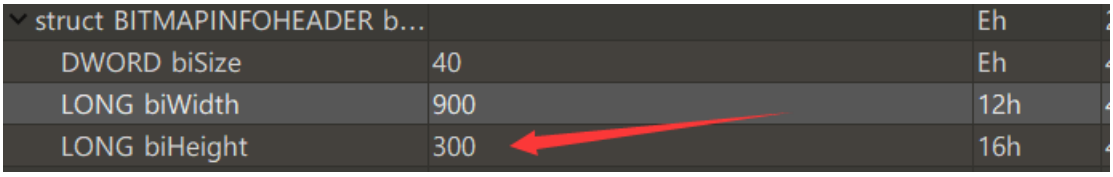
文件结构篇

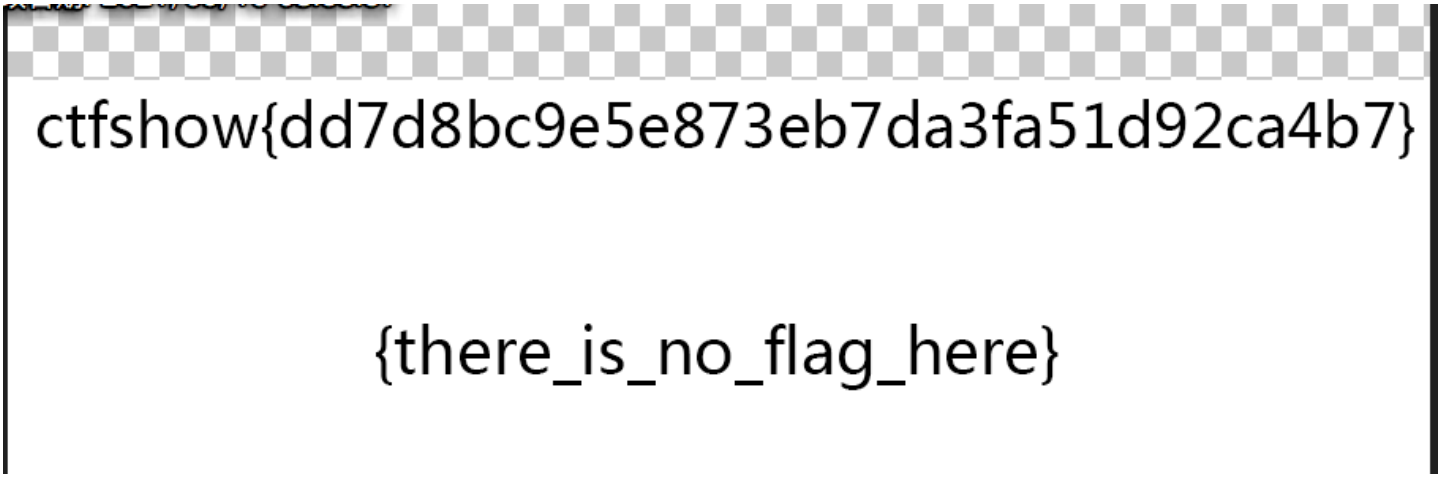
misc24-bmp改高度

提示：flag在图片上面

bmp格式文件

真flag在图片上面，改一下高度就可以看到了

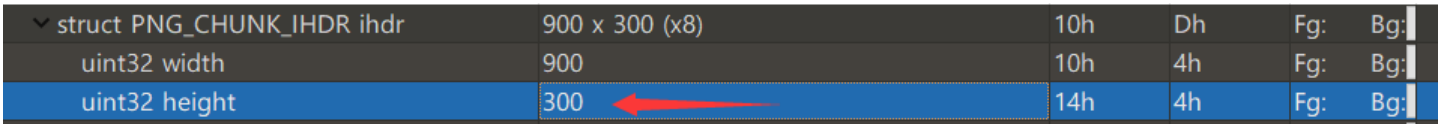




misc25-png改高度

提示：flag在图片下面

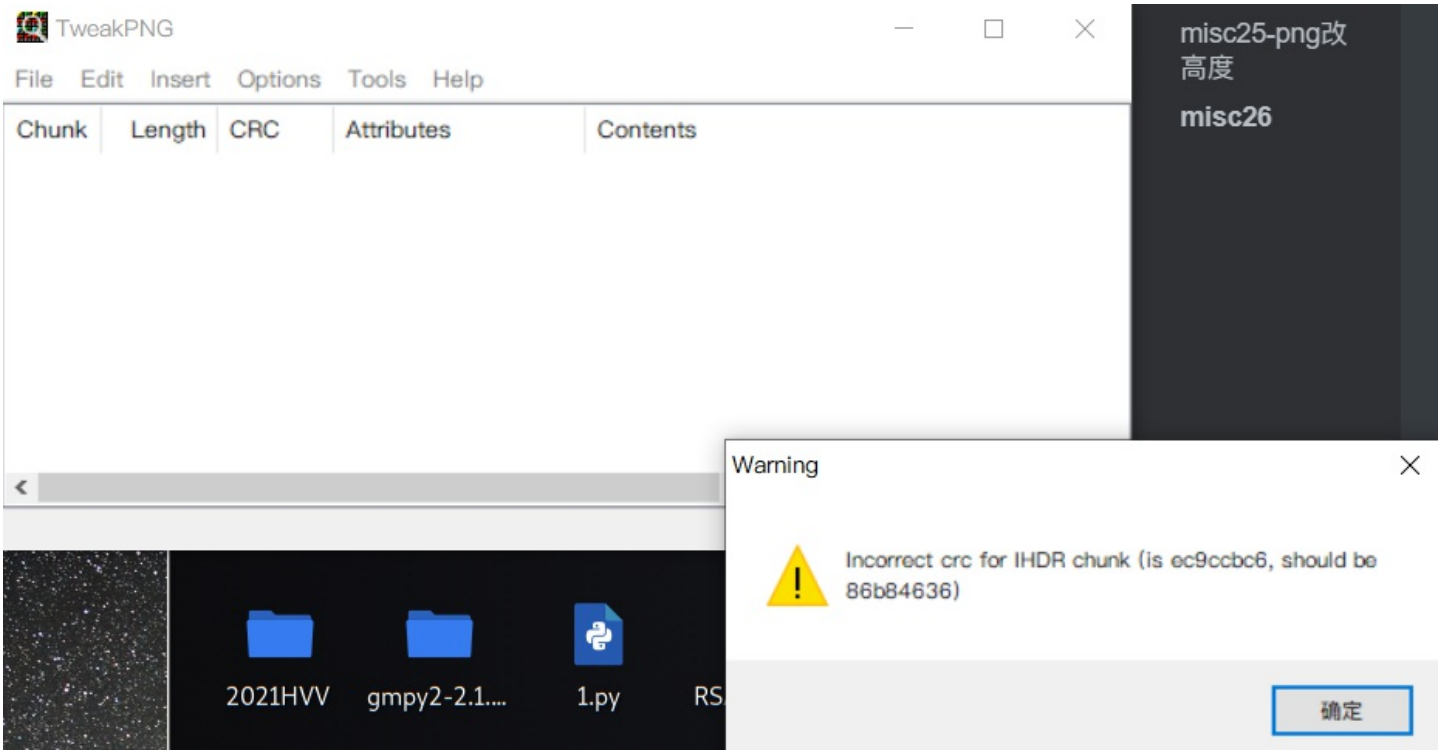
改高度即可



misc26-crc32

提示：flag还是在图片下面，但到底有多下面？

用tweakpng打开文件



通过python2脚本来获取图片高度

```
# -*- coding: utf-8 -*-
import binascii
import struct

#\x49\x48\x44\x52\x00\x00\x01\xF4\x00\x00\x01\xA4\x08\x06\x00\x00\x00

crc32key = 0xCBD6DF8A
for i in range(0, 65535):
  height = struct.pack('>i', i)
  #CRC: CBD6DF8A
  data = '\x49\x48\x44\x52\x00\x00\x01\xF4' + height + '\x08\x06\x00\x00\x00'

  crc32result = binascii.crc32(data) & 0xffffffff

  if crc32result == crc32key:
    print ''.join(map(lambda c: "%02X" % ord(c), height))
```

第二个脚本-推荐使用piccrc32.py

```
import binascii
import struct

crcbp = open("misc26.png", "rb").read()
for i in range(2000):
    for j in range(2000):
        data = crcbp[12:16] + struct.pack('>i', i)+struct.pack('>i', j)+crcbp[24:29]
        crc32 = binascii.crc32(data) & 0xffffffff
        if(crc32 == 0xEC9CCBC6):
            print(i, j)
            print('hex:', hex(i), hex(j))
```

```
PS C:\Users\q2723\Desktop> python2 .\test1.py
(900, 606)
('hex:', '0x384', '0x25e')
```

得出正确的高度

将0x25e转换为十进制，修改高度保存

{there_is_no_flag_here}

ctfshow{94aef1
+True height(hex) of this picture+
087a7ccf2e28e742efd704c}

所以说flag就是25e拼接上图片中给出的内容

misc27-jpg改高度

提示：flag在图片下面

修改高度

struct APP14 app14		2h	10h	Fg:	Bg:
struct DQT dqt		12h	86h	Fg:	Bg:
struct SOFx sof0		98h	13h	Fg:	Bg:
enum M_ID marker	M_SOF0 (FFC0h)	98h	2h	Fg:	Bg:
WORD szSection	17	9Ah	2h	Fg:	Bg:
ubyte precision	8	9Ch	1h	Fg:	Bg:
WORD Y_image	300	9Dh	2h	Fg:	Bg:
WORD X_image	900	9Fh	2h	Fg:	Bg:
ubyte nr_comp	3	A1h	1h	Fg:	Bg:

保存为图片

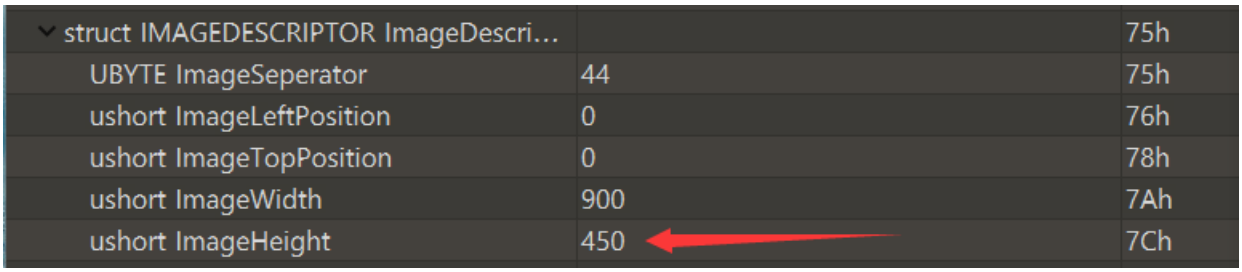
{there_is_no_flag_here}

ctfshow{5cc4f19eb01705b99bf41492430a1a14}

misc28-gif改高度

提示：flag在图片下面

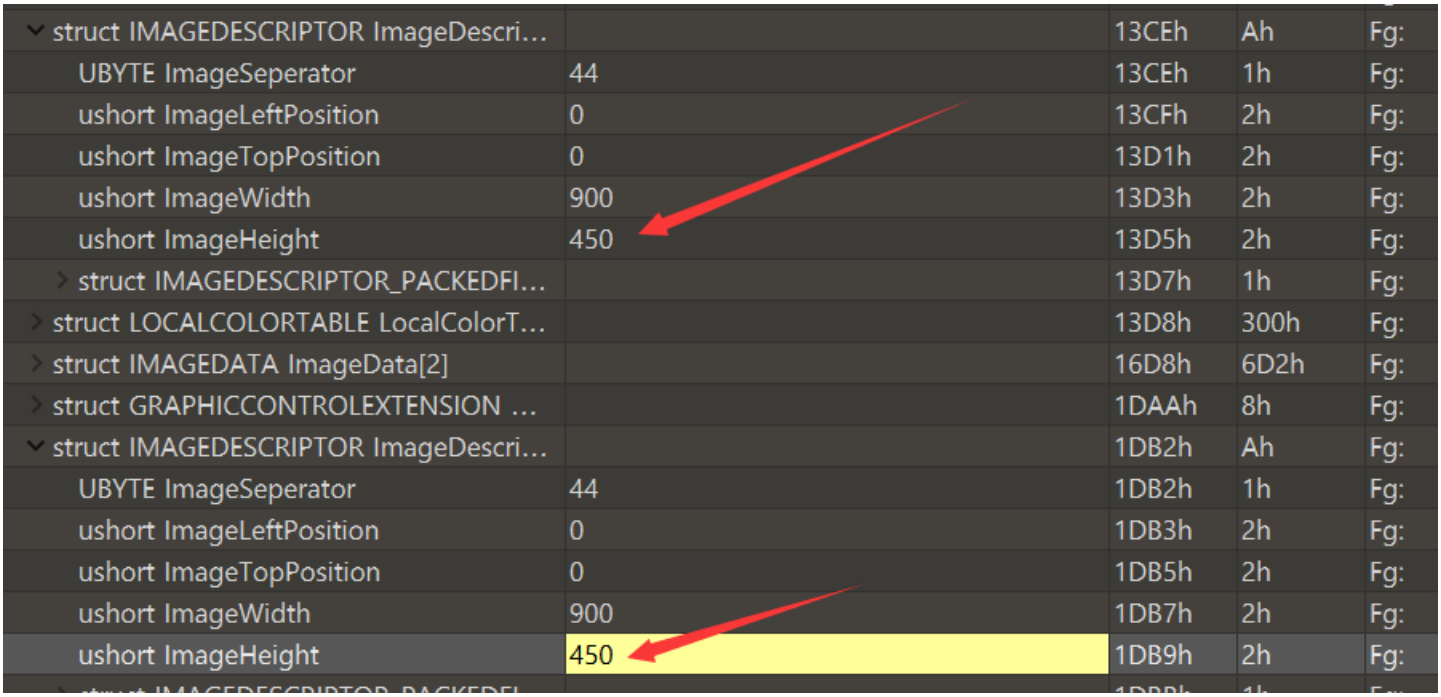
也是010改图片高度



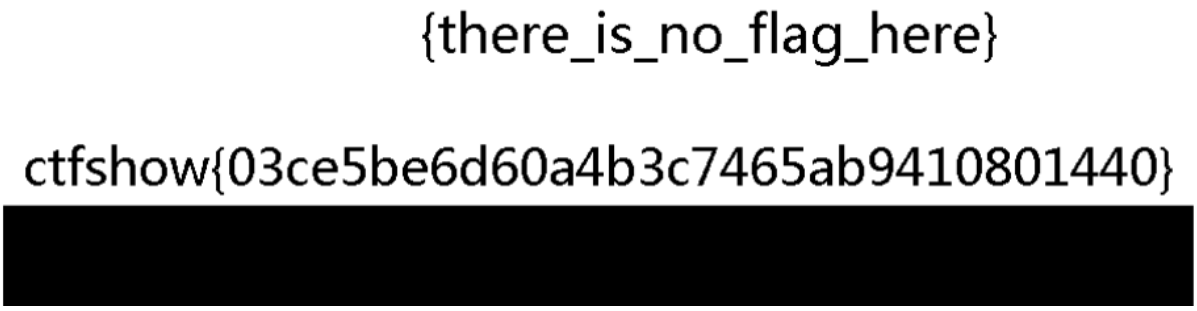
注意图片是gif格式，有不同的帧，所以在不同的块也有宽高的数值需要多试试，改这个有效的

misc29-多帧gif改高度

多帧gif动图

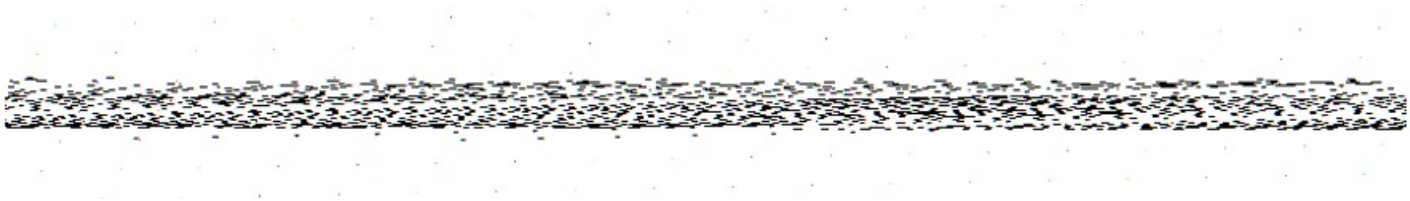


有很多的帧，需要都改一下高度，发现在第八帧有flag



misc30-bmp改宽度

提示：正确的宽度是950



发现图片是这样的，按照提示去修改宽度即可

misc31-bmp计算宽度

提示：高度是正确的，但正确的宽度是多少呢

bmp的文件格式

这张图片由900*150个像素，文件头占用53字节，文件尾在0x76F50位置，换算成10进制为487248

因为每个像素点由3个字节（十六进制的6位）表示，每个字节负责控制一种颜色，分别为蓝，绿，红，所以文件真是像素个数为

```
(487248-53)/3=162398
```

题目提示图片高度是正确的，所以说要计算宽度

宽度即为像素数除以高度：192398/150=1082（如果有余数要舍去）

将图片宽度改为1082保存图片打开即为flag

同样之前的misc24题已知宽度求高度也是同样的计算方法

misc32-png计算crc改宽度

知道高度爆破宽度，跟26题一样，可以用之前的脚本

新脚本-效果一样，这两个脚本都需要改crc的值

```
import zlib
import struct

# 同时爆破宽度和高度
filename = "misc32.png"
with open(filename, 'rb') as f:
    all_b = f.read()
    data = bytearray(all_b[12:29])
    n = 4095
    for w in range(n):
        width = bytearray(struct.pack('>i', w))
        for h in range(n):
            height = bytearray(struct.pack('>i', h))
            for x in range(4):
                data[x+4] = width[x]
                data[x+8] = height[x]
            crc32result = zlib.crc32(data)
            #替换成图片的crc
            if crc32result == 0xE14A4C0B:
                print("宽为：", end = '')
                print(width, end = ' ')
                print(int.from_bytes(width, byteorder='big'))
                print("高为：", end = '')
                print(height, end = ' ')
                print(int.from_bytes(height, byteorder='big'))
```

python3运行即可

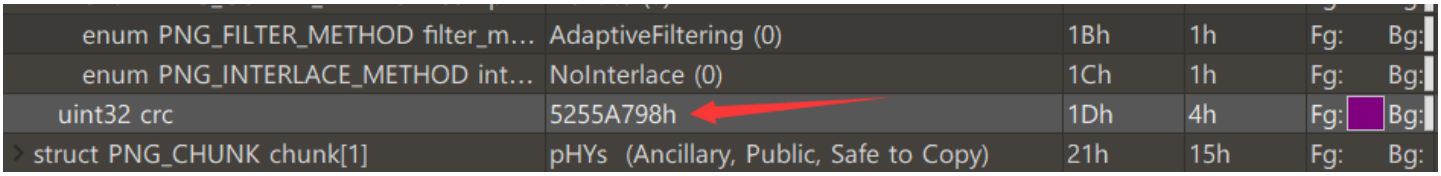
```
PS [redacted]\脚本\CRC32碰撞> python .\piccrc32-2.py
宽为： bytearray(b'\x00\x00\x04\x14') 1044
高为： bytearray(b'\x00\x00\x00\x96') 150
```

宽度为1044，修改宽度保存

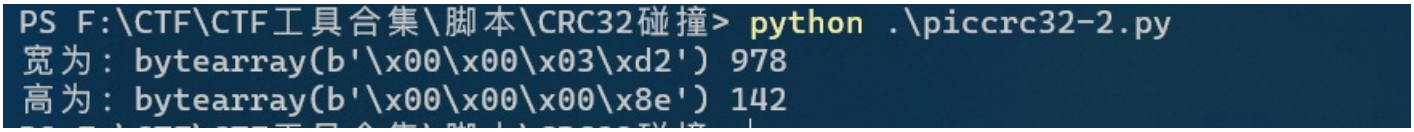
misc33-pngcrc计算宽高

高度和宽度都不对，使用32题的脚本计算下

在010中找到crc的值



更改脚本运行



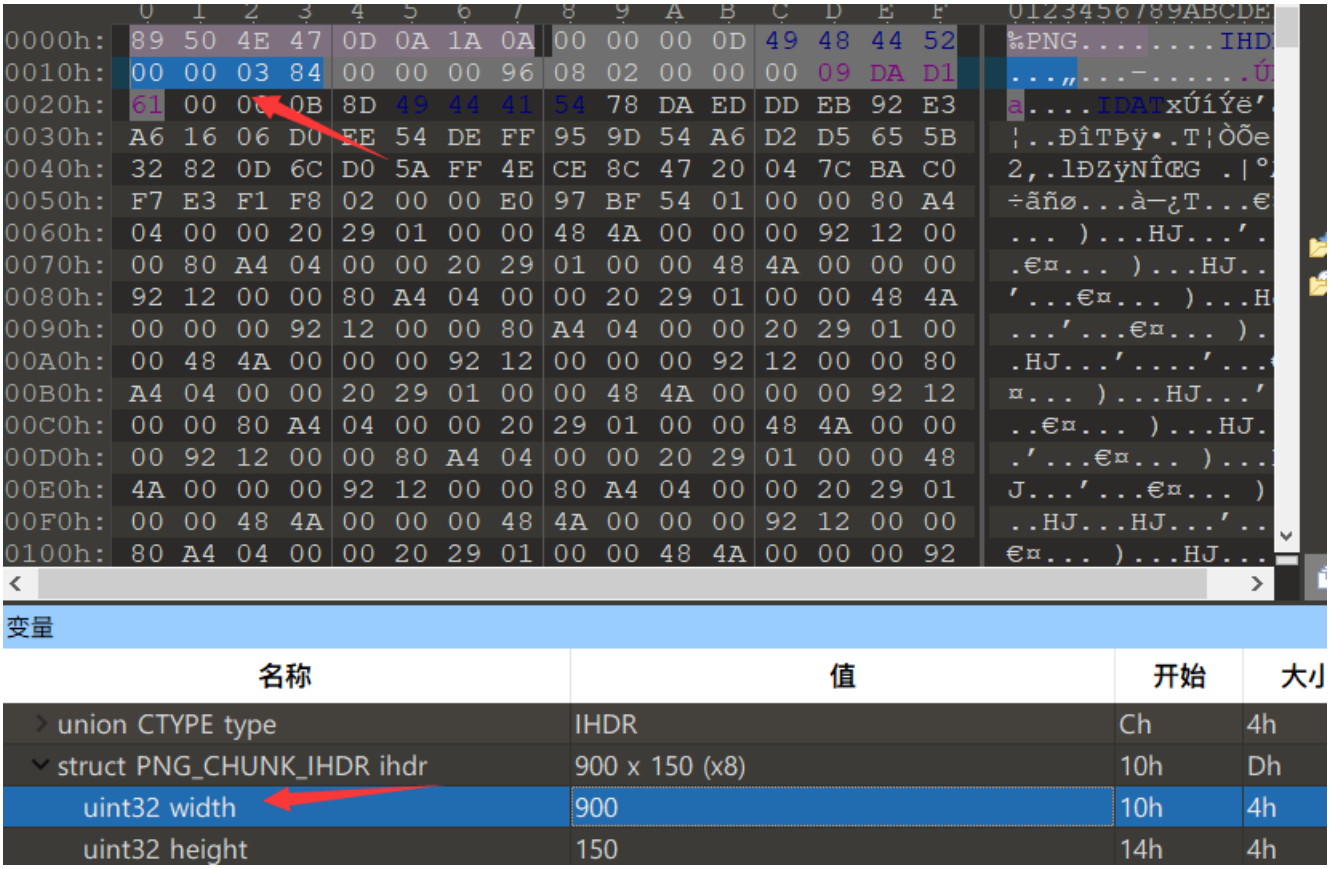
misc34-png爆破宽度

提示：出题人狗急跳墙，把IHDR块的CRC也改了，但我们知道正确宽度肯定大于900

```
import zlib
import struct
filename = "misc34.png"
with open(filename, 'rb') as f:
    all_b = f.read()
    #w = all_b[16:20]
    #h = all_b[20:24]
    for i in range(901,1200):
        name = str(i) + ".png"
        f1 = open(name,"wb")
        im = all_b[:16]+struct.pack('>i',i)+all_b[20:]
        f1.write(im)
        f1.close()
```

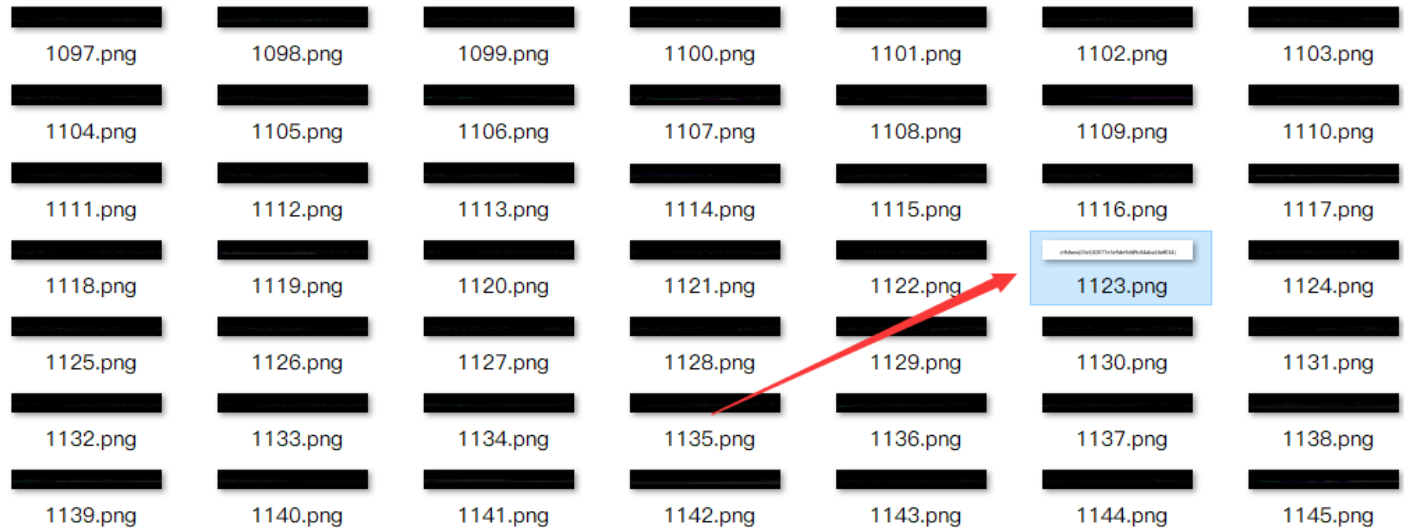
把生成的所有图片都保存下来了(建议在空文件夹里)，然后用眼看哪个是正常的。
最后得到正确的宽度是1123

这里的w和h需要打开图片去看下




```
struct --- 将字节串解读为打包的二进制数据
struct.pack(format, v1, v2, ...)
返回一个 bytes 对象，其中包含根据格式字符串 format 打包的值 v1, v2, ... 参数个数必须与格式字符串所要求的值完全匹配
```

运行结果



misc35-jpg爆破宽度

提示：出题人负隅顽抗，但我们知道正确宽度肯定大于900

```
#w = all_b[157:159]
#h = all_b[159:161]
```

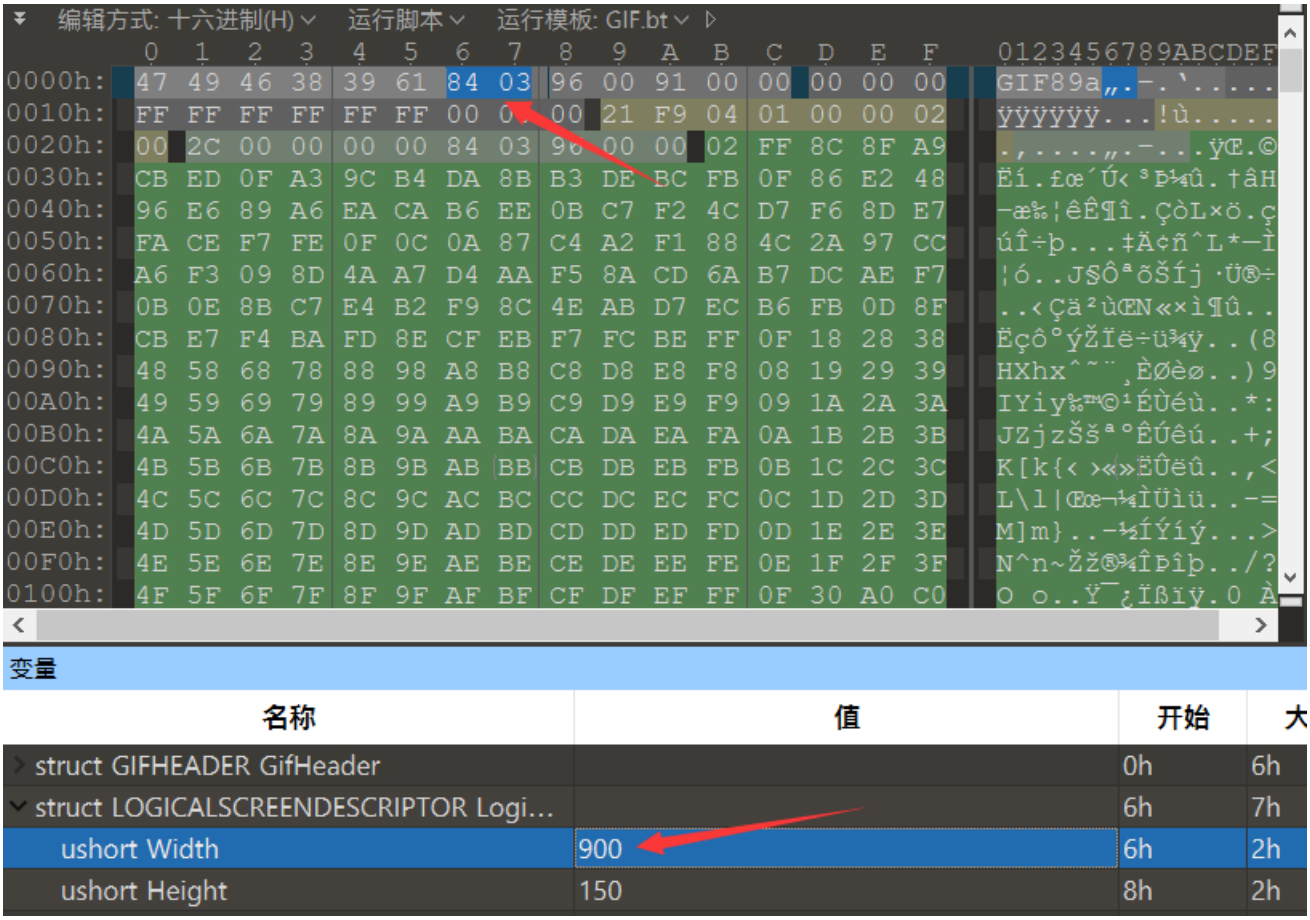
```
import zlib
import struct
filename = "misc35.jpg"
with open(filename, 'rb') as f:
    all_b = f.read()
    #w = all_b[159:161]
    #h = all_b[157:159]
    for i in range(901,1200):
        name = str(i) + ".jpg"
        f1 = open(name,"wb")
        im = all_b[:159]+struct.pack('>h',i)+all_b[161:]
        f1.write(im)
        f1.close()
```

需要把图片的基础高度调高一点

> 高度调到了600，宽度在993-1000这个范围内都可以得到flag

misc36-gif爆破宽度

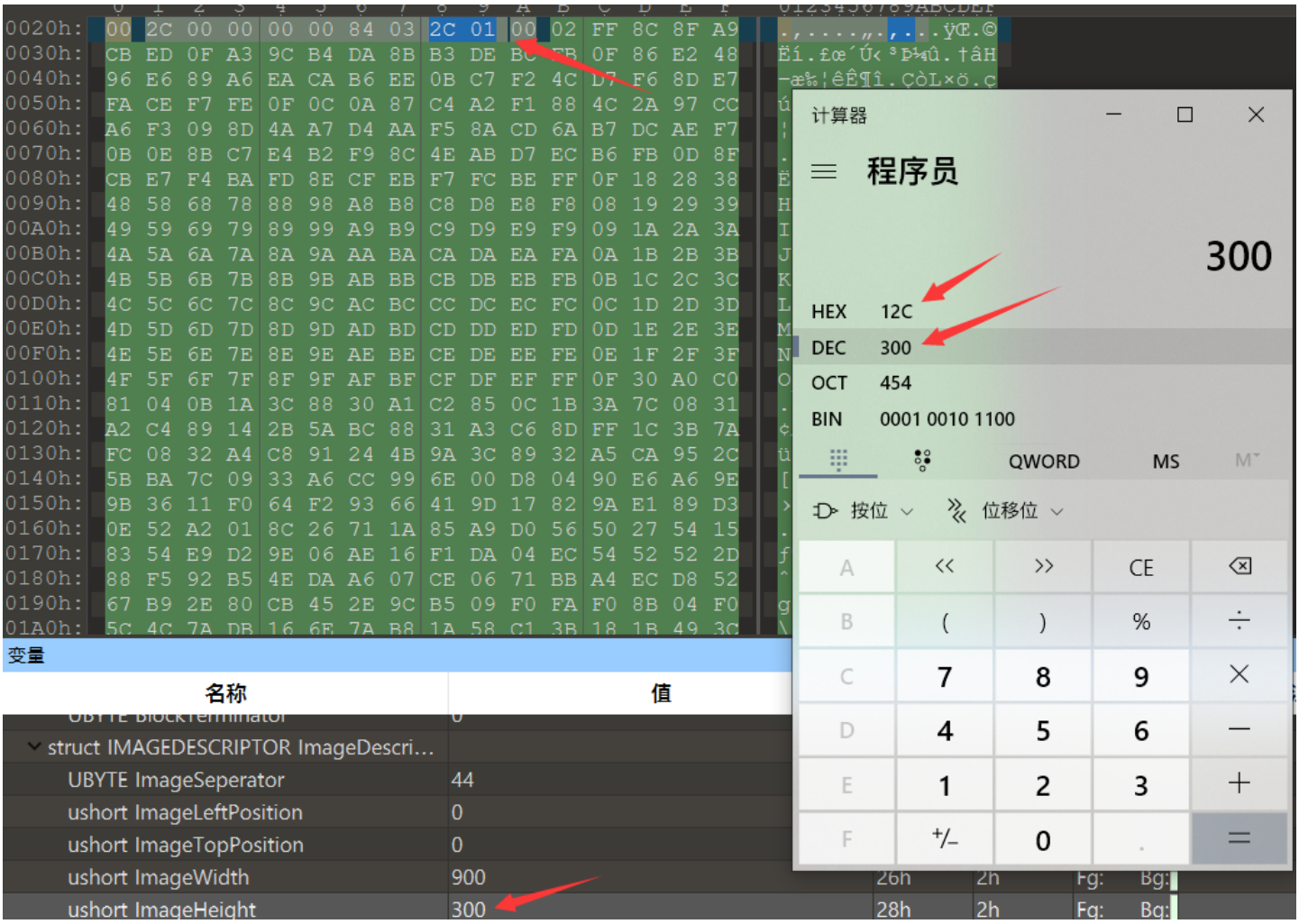
gif图片，提示正确宽度在920-950之间



改下脚本

```
import zlib
import struct
filename = "misc36.gif"
with open(filename, 'rb') as f:
    all_b = f.read()
    #w = all_b[6:8]
    for i in range(920,950):
        name = str(i) + ".gif"
        f1 = open(name,"wb")
        im = all_b[:38]+struct.pack('>h',i)[::-1]+all_b[40:]
        f1.write(im)
        f1.close()
```

注意高度存储时是倒着的，因为



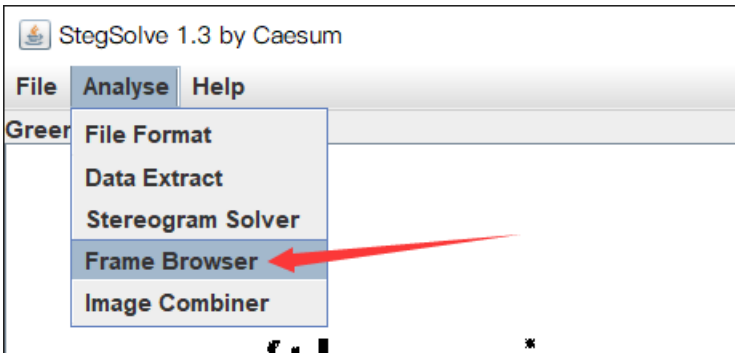
运行之后发现没有flag，需要提高图片高度再试

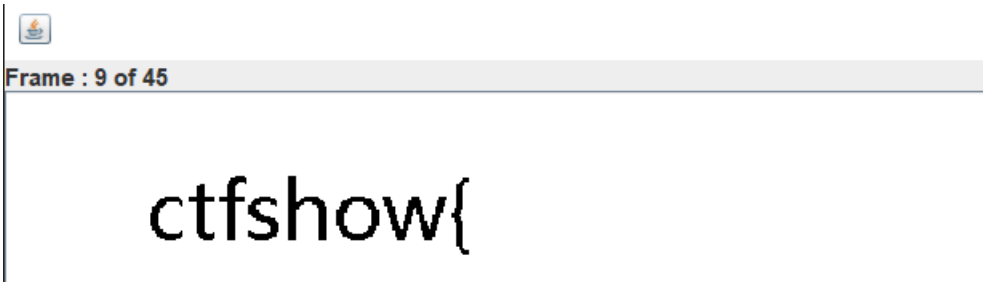
misc37-gif逐帧

提示flag再图片里

这是个gif动图，在某些帧里边发现了不一样的字符

用stegslove打开





flag藏在不同的帧里边

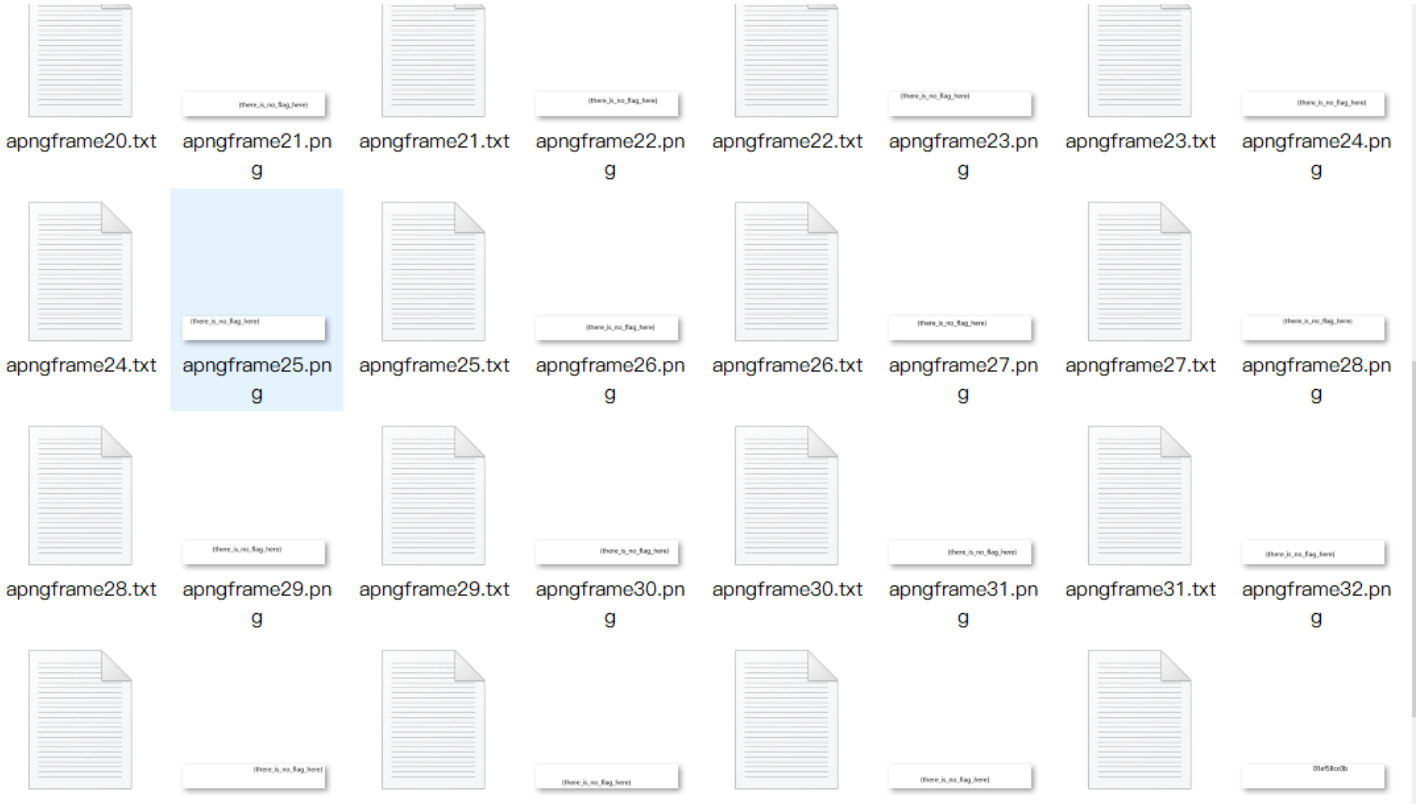
misc38-apng图片分离

提示：flag在图片里

是一个png图片，在win10自带的图片浏览器中并没有发现什么，使用chrome或者蜂蜜图片浏览器打开发现这是张可以动的png

使用apngdis.exe（APNG Disassembler）工具分离文件

```
PS C:\Users\[illegible]\Desktop> .\apngdis.exe .\misc38.png

APNG Disassembler 2.8

Reading '.\misc38.png'...
extracting frame 1 of 44
extracting frame 2 of 44
extracting frame 3 of 44
extracting frame 4 of 44
extracting frame 5 of 44
extracting frame 6 of 44
extracting frame 7 of 44
extracting frame 8 of 44
extracting frame 9 of 44
extracting frame 10 of 44
```



发现flag片段

misc39-gif时间差flag

提示：flag就像水，忽快忽慢地流

是一个gif，不过这里是利用不同帧之间的间隔时间来隐写的。
这里利用linux下的工具identify

```
安装命令：sudo apt-get install imagemagick
基本的命令格式：
    identify [options] input-fileidentify:命令名称
    options:参数
    input-file:文件名。
提取命令：identify -format "%T " misc39.gif > 1.txt
```

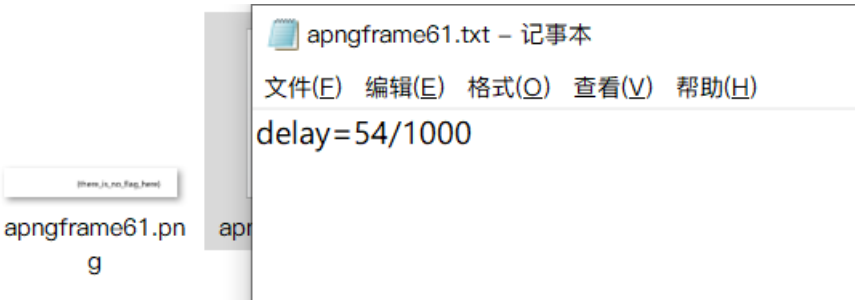
得到的一串36和37，考虑把37换成1、36换成0，就得到长度为287的二进制字符串，考虑每7位转一个字符(正常是8位一组)，得到flag

```
>>> str="37373636363737373737363736363737363637373637373736363737373736373636363737363737373737373637373737373736373
736373736373637363737363637363637373736363636373736363736373736373736373736373736373736373736373736373736373736373
736363736373736373637363737363637373637373636363736363737363737373736363637363737363736373736373736373736373637373
637373736363637373637363636373737363637363737363737363637373637363737373636363736373737363637373736363736373737363
637363737363636373737373636363637363737363636363737363736363637373637363737373636363736373737363637373736363736373
63736373736363637373737363636363736373736363637373637363737373636363736373737363637373736363736373737363637373736363
637363737363636373737373636363637363737363636363737363736363637373637363737373636363736373737363637373736363736373737
1000111100001011000001101000110101011100111111101"
>>> str=str.replace("37","1")
>>> str=str.replace("36","0")
>>> str
'1100011111010011001101110011111010001101111111011111111011011010101100100111000011000101100101100110110011001110010111001
0110101110011011000100110111111000101100101011001001101100111000110010001110010110110011001111100001011100111001010111000101
1000111100001011000001101000110101011100111111101'
>>> len(str)
287
```

```
s="1100011111010011001101110011111010001101111111011111111011011010101100100111000011000101100101100110110011
0011100101110010110101110011011000100110111110001011001010110010011011001110001100100011100101101100110011
1000010111001110010111000101100011111000010110000011010001101010111001111110"
flag=""
for i in range(41): #287//7
    flag += chr(int(s[7i:7(i+1)],2))
print(flag)
```

misc-40apng时间差flag

apng动图，使用APNG Disassembler工具，分离文件，同时会生成txt文件，记录详细信息



例如这个需要把54提取出来，转换成对应的ascii码

```
flag=""
for i in range(1,69):
    if i < 10:
        i = "0" + str(i)
    f = open("apngframe" + str(i) + ".txt")
    s = f.read()
    # delay=116/1000
    flag += chr(int(s.split("/")[0][6:]))
print(flag)
```

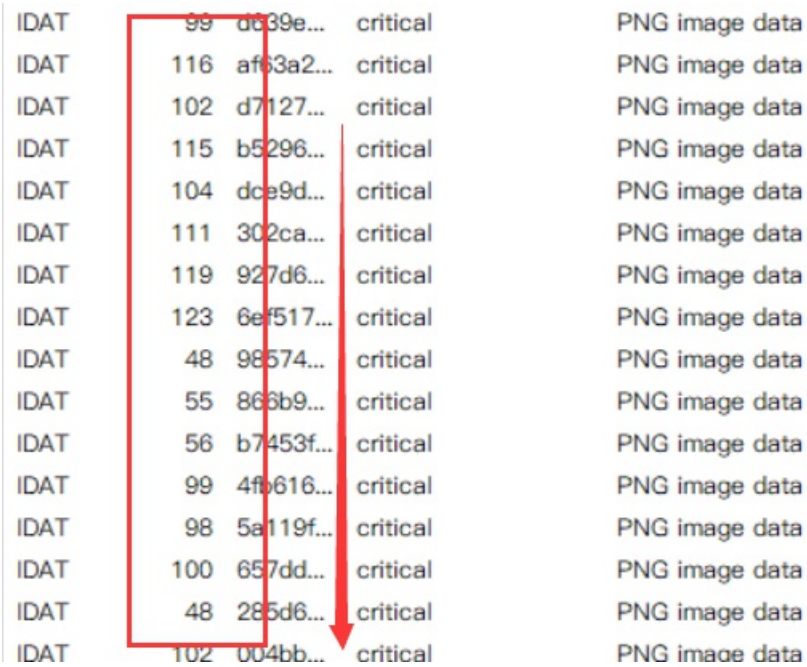
实际实在28张图片时出现flag，可以将range范围改为28,69

```
PS C:\...\新建文件夹> python .\test.py
ctfshow{95ca029...f0f6b1bdaca394a6fcb95b}
```

misc42-IDAT数据块长度

提示：flag有多长？2cm……不好意思打错了，41位

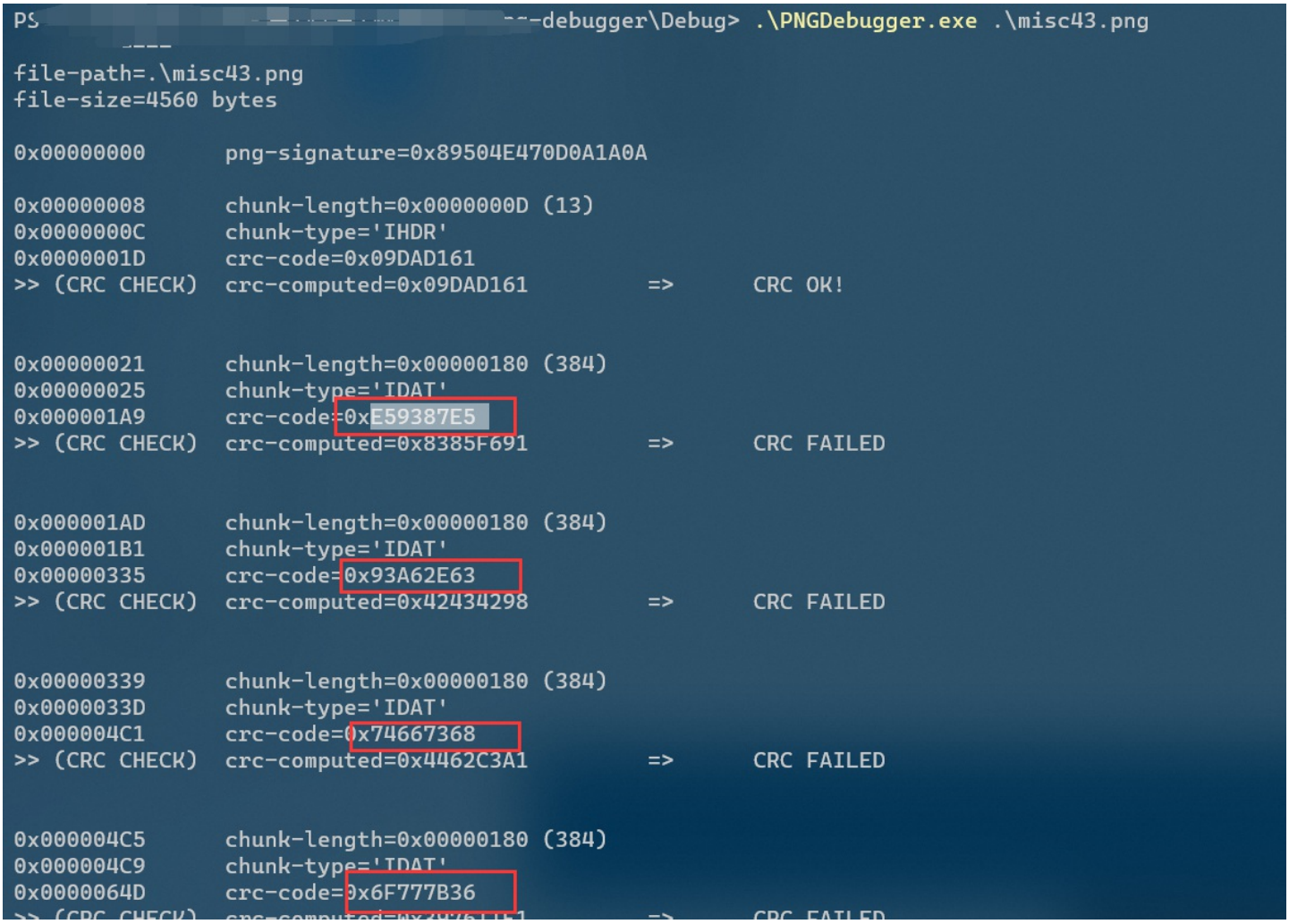
用tweakpng打开图片，发现IDAT数据块的长度转换成ASCII码就是flag



misc43-收集报错

提示：错误中隐藏着通往正确答案的道路

tweakpng打开图片发现很多报错提示，用pngdebugger看一下



发现很多报错代码，结合提示将其拼合转换成字符串

python整理下格式

```
string ="E59387E50x93A62E630x746673680x6F777B360x656232350x383966660x666635650x333930660x653662380x373530340x646263300x3839327D"
string=string.replace("0x","")
for i in range(0,len(string),2):
    print((string[i:i+2]),end=" ")
```

因为前两个字符是中文python字符集里没有，所以偷懒直接用工具

Ascii Encoding

Ascii Encoding

Text

哇哦.ctfshow{6eb2589ffff5e390fe6b87504dbc0892}

Bin

11100101 10010011 10000111 11100101 10010011 10100110 101110 1100011 1110100 1100110 1110011 1101000 1101111 1110111 1111011 110110
1100101 1100010 110010 110101 111000 111001 1100110 1100110 1100110 1100110 110101 1100101 110011 111001 110000 1100110 1100101 110110
1100010 111000 110111 110101 110000 110100 1100100 1100010 1100011 110000 111000 111001 110010 1111101

Oct

345 223 207 345 223 246 56 143 164 146 163 150 157 167 173 66 145 142 62 65 70 71 146 146 146 146 65 145 63 71 60 146 145 66 142 70 67 65 60 64 144 142
143 60 70 71 62 175

Dec

229 147 135 229 147 166 46 99 116 102 115 104 111 119 123 54 101 98 50 53 56 57 102 102 102 102 53 101 51 57 48 102 101 54 98 56 55 53 48 52 100 98 99 48
56 57 50 125

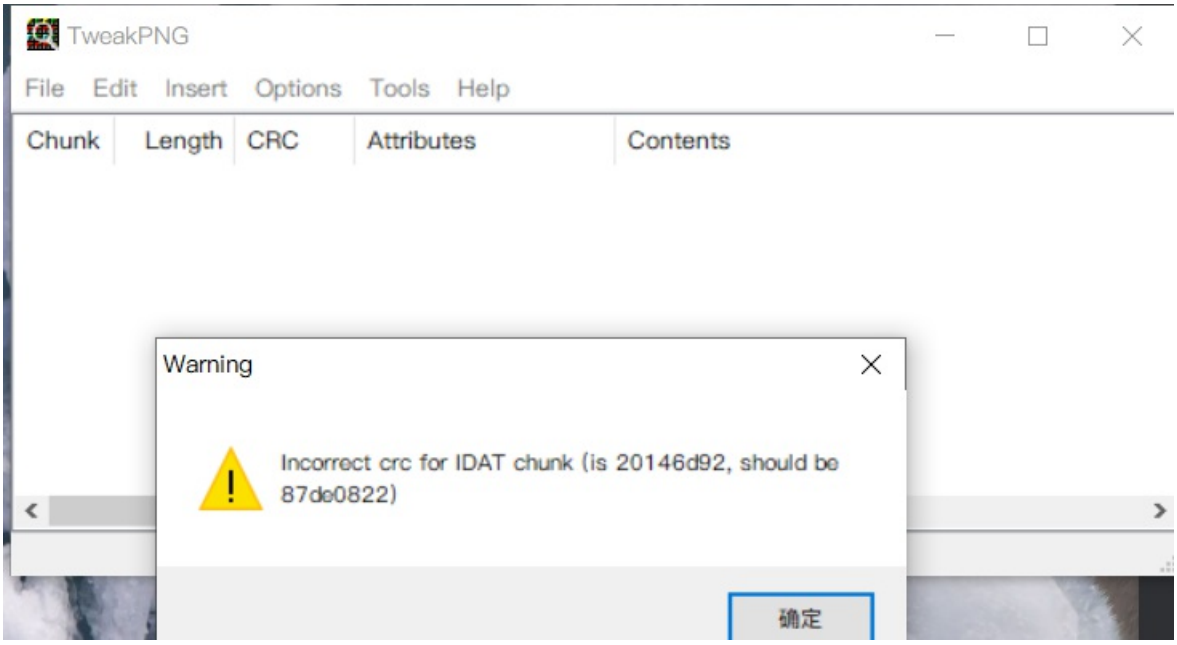
Hex

E5 93 87 E5 93 A6 2E 63 74 66 73 68 6F 77 7B 36 65 62 32 35 38 39 66 66 66 66 35 65 33 39 30 66 65 36 62 38 37 35 30 34 64 62 63 30 38 39 32 7D

misc44-转换报错提示

提示：错误中还隐藏着坑

tweakpng 依旧是一堆报错，而且这次IDAT的数据块非常的多



使用pngDebugger将输出导出为文档

> 然后写脚本，把 `CRC OK的替换成1，CRC FAILED替换成0`

```
f=open("1.txt","r")
s=f.read()
f.close()
flag=""
for i in s.split():
    if "OK!" == i:
        flag += "1"
    elif "FAILED" ==i:
        flag += "0"
print(flag)
#11111111111111101100011011101000110011001110011011010000110111101110111011110110110001101100011001100010110000
10110011000110011001100100110001001100110001110010011011000110011001100000011100001100110011000110011000100110011
00011011000110011001100100011001100110001011000100110010100110111001110000011001101100110001101100011100101100101
101111101
print(len(flag)) #344
for i in range(43):
    print(chr(int(flag[8*i:8*(i+1)],2)),end="")
```

misc45-转换格式

提示：有时候也需要换一换思维格式

其实图片中是是藏有了其他文件但是因为png和bmp格式图片像素点的读取方式不一样，在png格式下分离不出文件，需要先把图片从png格式转换成bmp格式，在进行binwalk提取得到flag.png

在线转换网站：https://cn.onlineconvert.com/pdf-to-bmp

misc46-49卷废了

misc46

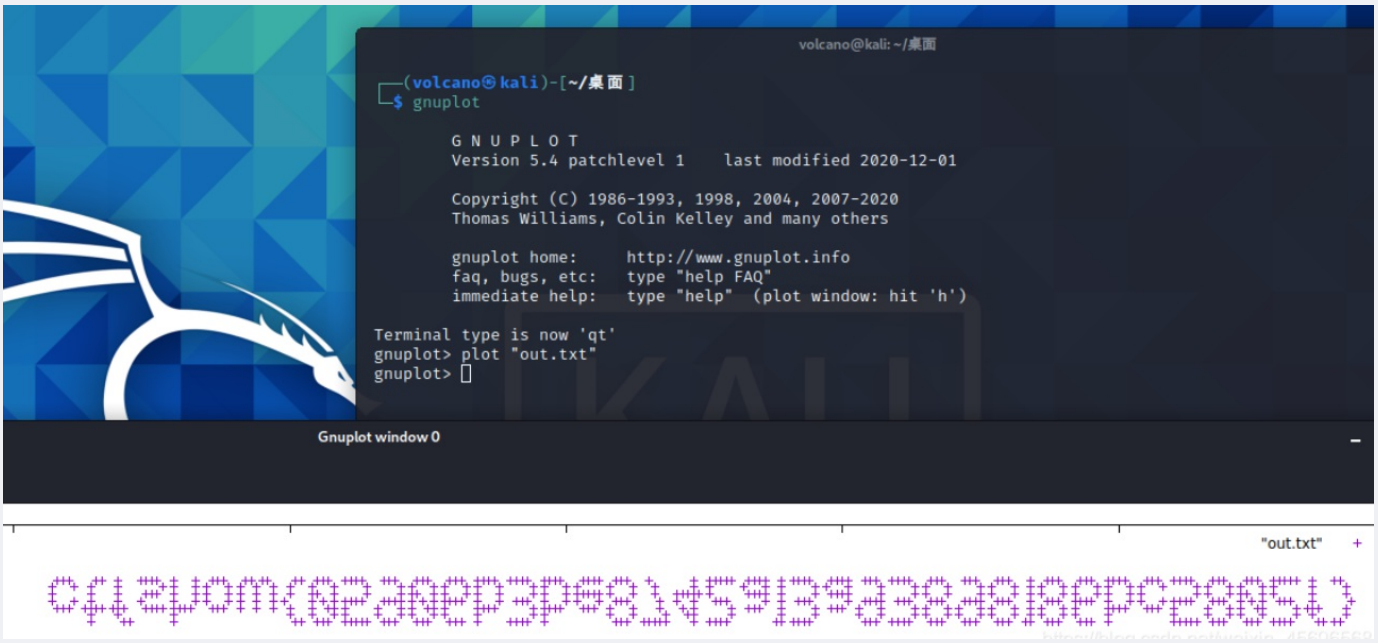
是一个gif
提取出它的详细信息：identify misc46.gif > /1.txt (这里直接在根目录生成1.txt，好找)
内容大概长这样，其中0+0、174+49、196+47这些是偏移量，这题就利用这个来作图。

```
) 查看(V)  帮助(H)
面/misc46.gif[0] GIF 900x150 900x150+0+0 8-bit sRGB 2c 0.040u 0:00.050
面/misc46.gif[1] GIF 450x50 900x150+174+49 8-bit sRGB 16c 0.040u 0:00.054
面/misc46.gif[2] GIF 450x50 900x150+196+47 8-bit sRGB 16c 0.040u 0:00.054
面/misc46.gif[3] GIF 450x50 900x150+256+49 8-bit sRGB 16c 0.040u 0:00.054
面/misc46.gif[4] GIF 450x50 900x150+293+52 8-bit sRGB 16c 0.040u 0:00.054
面/misc46.gif[5] GIF 450x50 900x150+220+49 8-bit sRGB 16c 0.040u 0:00.054
```

写一个很简单的脚本把坐标提取出来

```
f = open("1.txt","r")
x = f.readlines()
f.close()

f = open("out.txt","w")
for i in x:
    f.write(i.split("+")[1])
    f.write(" ")
    f.write(i.split("+")2)
    f.write("\n")
f.close()
```

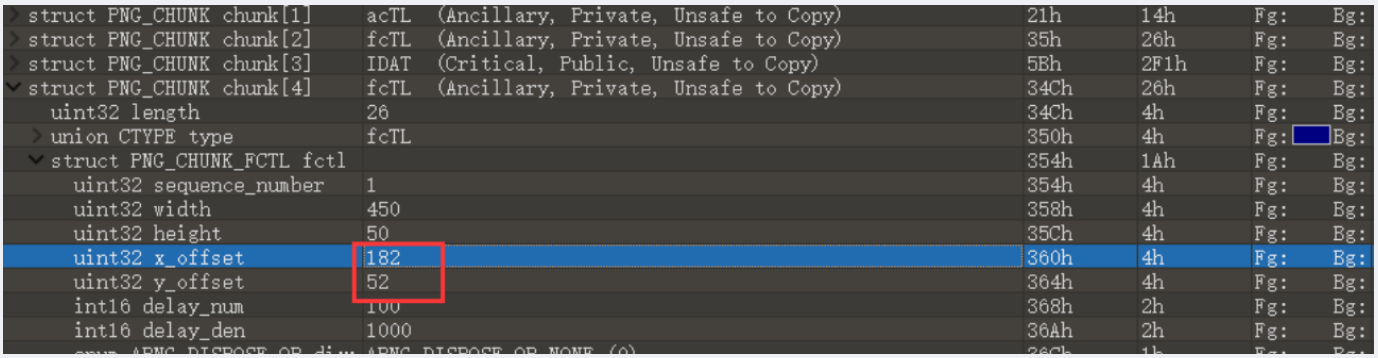
再利用gnuplot作图，画出来的结果有点模糊，自行调整



再翻转一下得到flag

misc47

给了一个png，打开发现没内容，用浏览器打开，确认是apng
思路和上题一致，不过稍微复杂一点，先通过这篇文章了解一下apng文件结构，简单来说就是每一个IDAT块前面都会有一个fcTL块，它其中就包含水平垂直偏移量
如下，对应坐标点就是(182,52)

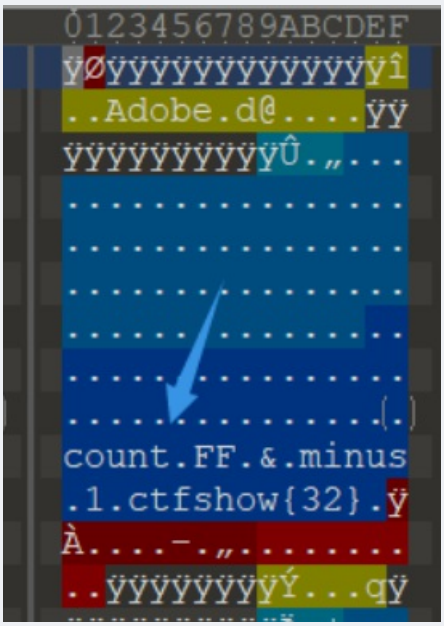


之前的脚本有的师傅使用的时候有点小问题，所以改了一下

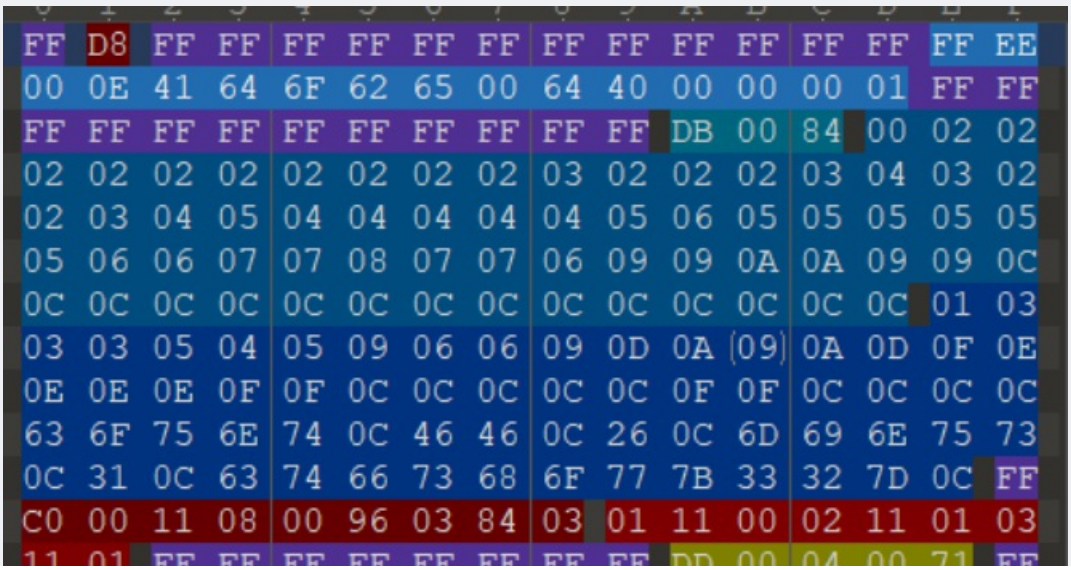
```
from PIL import Image

im = Image.new('RGB', (400, 80), 255)
f = open('1.txt','r') #把图片的十六进制导出，保存为1.txt
c = f.read()
f.close()

lt = c.split("6663544C")
for i in range(2,len(lt)):
    x = int(lti,16)
    y = int(lti,16)
    im.putpixel((x,y),0)
im.show()
```

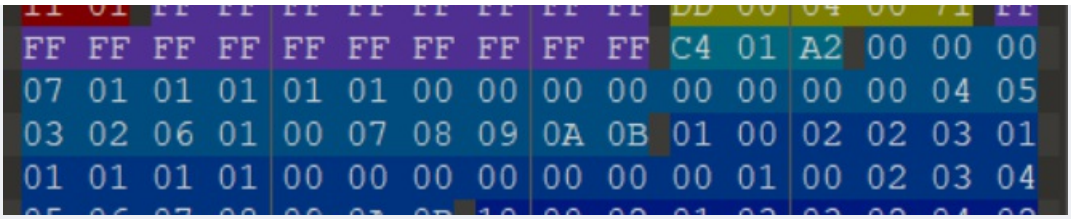
misc48

用010editor打开，发现有提示
1、统计FF的数量，再减去1
2、ctfshow{}中包含32个字符

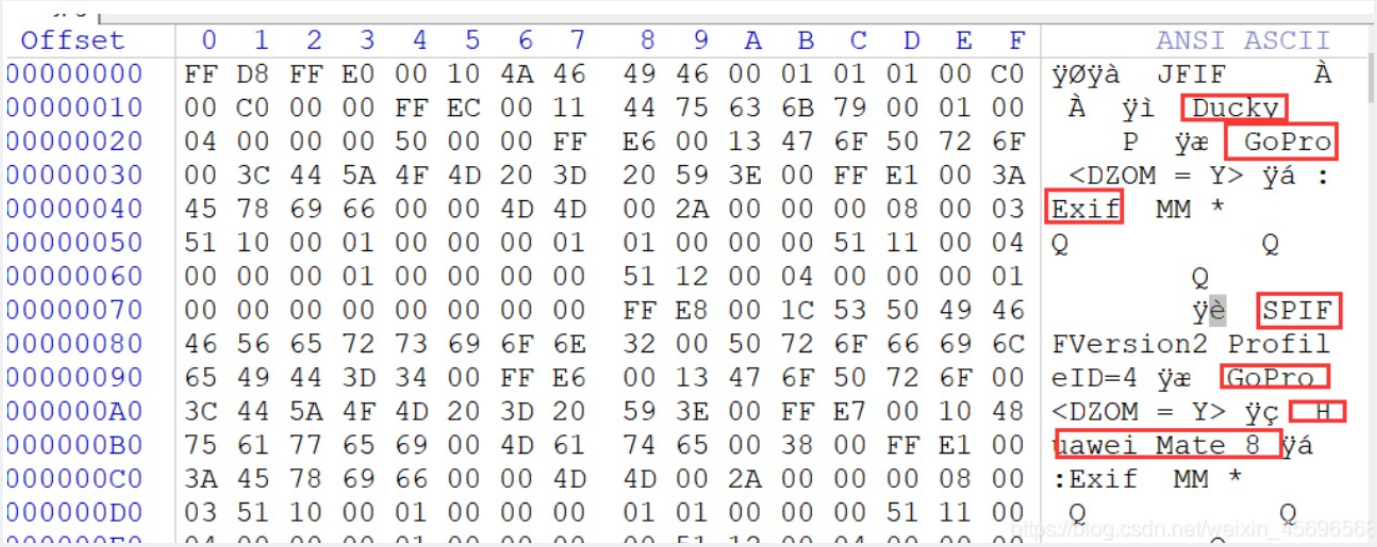


提示了，但没有完全提示，因为第一条提示，其实指的是统计每两个有意义块之间的FF的数量再减一
图中紫色的就是，开头的那个FF也算，因为只有一个，减去1后就是0；接下来是12、11、0...



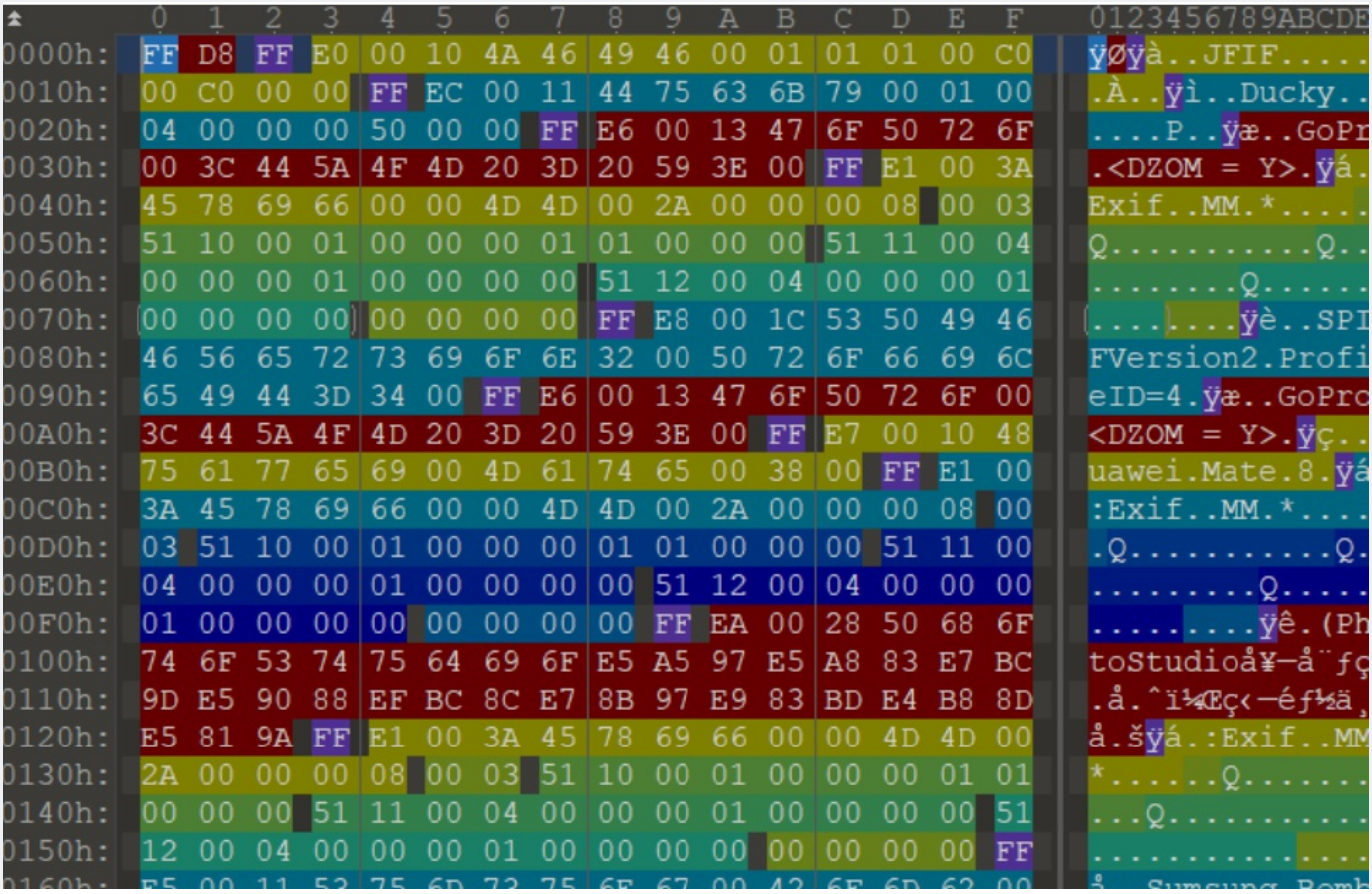


因为flag长度是32位，所以只统计前32个，即：
0 12 11 0 7 10 13 13 9 0 9 13 0 13 6 0 10 9 2 1 0 1 10 8 11 5 12 7 2 2 3 10
分别转十六进制后，再连接在一起，得到：ctfshow{0cb07add909d0d60a92101a8b5c7223a}
misc49
提示：它们一来就是十六种。本题略脑洞，可跳过
八神的脑洞题，靠我自己想是不行的，果断参考wp
用winhex打开，能看到很多字符串，这其实是八神给的提示，虽然我没get到



Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI ASCII
00000000	FF	D8	FF	E0	00	10	4A	46	49	46	00	01	01	01	00	C0	ÿØÿà JFIF À
00000010	00	C0	00	00	FF	EC	00	11	44	75	63	6B	79	00	01	00	À ÿì Ducky
00000020	04	00	00	00	50	00	00	FF	E6	00	13	47	6F	50	72	6F	P ÿæ GoPro
00000030	00	3C	44	5A	4F	4D	20	3D	20	59	3E	00	FF	E1	00	3A	<DZOM = Y> ÿá :
00000040	45	78	69	66	00	00	4D	4D	00	2A	00	00	00	08	00	03	Exif MM *
00000050	51	10	00	01	00	00	00	01	01	00	00	00	51	11	00	04	Q Q
00000060	00	00	00	01	00	00	00	00	51	12	00	04	00	00	00	01	Q
00000070	00	00	00	00	00	00	00	00	FF	E8	00	1C	53	50	49	46	ÿè SPIF
00000080	46	56	65	72	73	69	6F	6E	32	00	50	72	6F	66	69	6C	FVersion2 Profil
00000090	65	49	44	3D	34	00	FF	E6	00	13	47	6F	50	72	6F	00	eID=4 ÿæ GoPro
000000A0	3C	44	5A	4F	4D	20	3D	20	59	3E	00	FF	E7	00	10	48	<DZOM = Y> ÿç H
000000B0	75	61	77	65	69	00	4D	61	74	65	00	38	00	FF	E1	00	uawei Mate 8 ÿá
000000C0	3A	45	78	69	66	00	00	4D	4D	00	2A	00	00	00	08	00	:Exif MM *
000000D0	03	51	10	00	01	00	00	00	01	01	00	00	00	51	11	00	Q Q

重点是这些字符串前面，都出现过FFE? 这种格式的数据，搜索一下发现有挺多的



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDE
0000h:	FF	D8	FF	E0	00	10	4A	46	49	46	00	01	01	01	00	C0	ÿØÿà..JFIF.....
0010h:	00	C0	00	00	FF	EC	00	11	44	75	63	6B	79	00	01	00	.À..ÿì..Ducky..
0020h:	04	00	00	00	50	00	00	FF	E6	00	13	47	6F	50	72	6F	....P..ÿæ..GoPr
0030h:	00	3C	44	5A	4F	4D	20	3D	20	59	3E	00	FF	E1	00	3A	.<DZOM = Y>.ÿá.
0040h:	45	78	69	66	00	00	4D	4D	00	2A	00	00	00	08	00	03	Exif..MM.*.....
0050h:	51	10	00	01	00	00	00	01	01	00	00	00	51	11	00	04	Q...........Q..
0060h:	00	00	00	01	00	00	00	00	51	12	00	04	00	00	00	01	........Q.....
0070h:	00	00	00	00	00	00	00	00	FF	E8	00	1C	53	50	49	46	........ÿè..SPI
0080h:	46	56	65	72	73	69	6F	6E	32	00	50	72	6F	66	69	6C	FVersion2.Profi
0090h:	65	49	44	3D	34	00	FF	E6	00	13	47	6F	50	72	6F	00	eID=4.ÿæ..GoPro.
00A0h:	3C	44	5A	4F	4D	20	3D	20	59	3E	00	FF	E7	00	10	48	<DZOM = Y>.ÿç..H
00B0h:	75	61	77	65	69	00	4D	61	74	65	00	38	00	FF	E1	00	uawei.Mate.8.ÿá.
00C0h:	3A	45	78	69	66	00	00	4D	4D	00	2A	00	00	00	08	00	:Exif..MM.*.....
00D0h:	03	51	10	00	01	00	00	00	01	01	00	00	00	51	11	00	.Q...........Q..
00E0h:	04	00	00	00	01	00	00	00	00	51	12	00	04	00	00	00	.........Q.....
00F0h:	01	00	00	00	00	00	00	00	00	FF	EA	00	28	50	68	6F	.........ÿê.(Ph
0100h:	74	6F	53	74	75	64	69	6F	E5	A5	97	E5	A8	83	E7	BC	toStudioå¥—å¨ƒç¼
0110h:	9D	E5	90	88	EF	BC	8C	E7	8B	97	E9	83	BD	E4	B8	8D	.å.ˆï¼Œç‹—éƒ½ä¸
0120h:	E5	81	9A	FF	E1	00	3A	45	78	69	66	00	00	4D	4D	00	å.šÿá.:Exif..MM.
0130h:	2A	00	00	00	08	00	03	51	10	00	01	00	00	00	01	01	*.....Q.....
0140h:	00	00	00	51	11	00	04	00	00	00	01	00	00	00	00	51	...Q...........Q
0150h:	12	00	04	00	00	00	01	00	00	00	00	00	00	00	00	FF	.....

```
0170h: 37 00 FF E3 00 13 4D 45 54 41 00 00 4B 6F 64 61   7.ÿã..META..Koda
0180h: 6B 00 54 2D 30 33 00 FF EF 00 10 47 72 61 70 68   k.T-03.ÿï..Graph
0190h: 43 6F 6E 76 00 EA 02 71 00 FF E5 00 11 53 75 6D   Conv.ê.q.ÿå..Sum
01A0h: 73 75 6E 67 00 42 6F 6D 62 00 37 00 FF ED 00 38   sung.Bomb.7.ÿí.8
01B0h: 50 68 6F 74 6F 73 68 6F 70 20 33 2E 30 00 38 42   Photoshop 3.0.8B
01C0h: 49 4D 04 04 00 00 00 00 00 00 38 42 49 4D 04 25   IM........8BIM.%
01D0h: 00 00 00 00 00 10 D4 1D 8C D9 8F 00 B2 04 E9 80   ......Ô.ŒÙ..².é.
01E0h: 09 98 EC F8 42 7E FF EA 00 28 50 68 6F 74 6F 53   .˜ìøB~ÿê.(PhotoS
```

把所有十六进制数保存在2.txt中，用一个小脚本处理一下

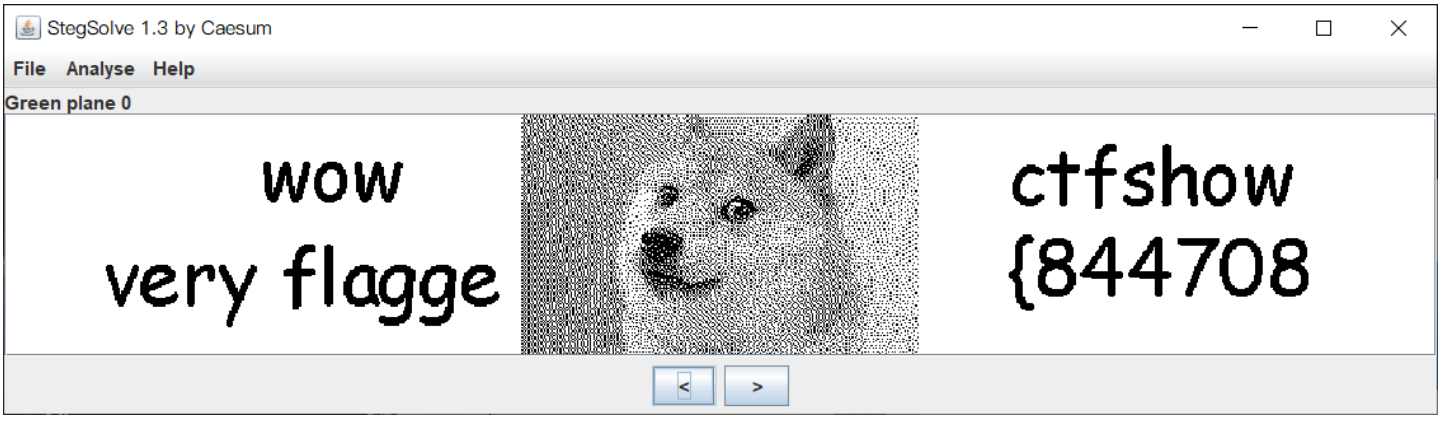
```
f=open("2.txt","r")
txt=f.read().replace("\n","")
f.close()

l=txt.split("FFE")
flag=""
for i in range(1,len(l)):
    flag += li
print(flag.lower()[:32]) #得到的结果套上ctfshow{}
```

其实就是把FFE后面的那个字符提取出来，再连接在一起，一共32位()，这就是flag。
这里写脚本的时候有个小失误，把这种也统计进去了，所以只有前32位是符合格式的正确的。

颜色通道

misc50-stegslove一把梭

既然提示了颜色通道



剩余的在后边几个通道

总结

1. 以上misc图片题基本包含ctf中的题型，但是没有涉及到隐写的内容，f5，outguess，盲水印等都没有出现
2. misc图片题有固定的套路需要多细心观察，有耐心，做题的过程中也发现了很多之前没有注意的盲点
3. 我们一起成长，一起进步

参考链接

https://blog.csdn.net/weixin_45696568/article/details/115261347

感谢z.volcano师傅写的文章给我很大的启发