CTFshow DJBCTF MISC(大吉杯) WP

原创

七月7yue 于 2021-01-24 22:11:25 发布 2019 收藏 3

分类专栏： MISC CTF 文章标签： 信息安全

本文链接：https://blog.csdn.net/qyCraner/article/details/113097425

MISC 同时被 2 个专栏收录

4 篇文章 0 订阅

订阅专栏

CTF

5 篇文章 0 订阅

订阅专栏

只做了misc的菜鸡，来写写misc部分的writeup，狸神的FM到最后也看不懂，太难了呜呜呜。

博客原文：http://www.7yue.top/djbctf/

十八般兵器

hint1：JPHS

hint2：用Notepad++打开试试？

hint3：前十种兵器对应10进制，后八种对应8进制

根据hint1，先利用JPHS将18张图片均解密一下，密码为空，每张图片解密后的数据末尾都有一段数字，这幅图是删掉了部分空格，实际需要往下拉。

flag{flag_1s_Not_herE}

0963

根据题目的武器顺序，前十张图片的数字组合起来十进制转十六进制。后八张图片的数字组合起来八进制转十六进制，asc转码一下得到flag

```
flag{CTFshow_10_bA_Ban_b1ng_Q1}
```

请问大吉杯的签到是在这里签吗

请问大吉杯的签到是在这里签吗

100

flag为全部小写字母，没有空格

@感谢cheyenne师傅供题

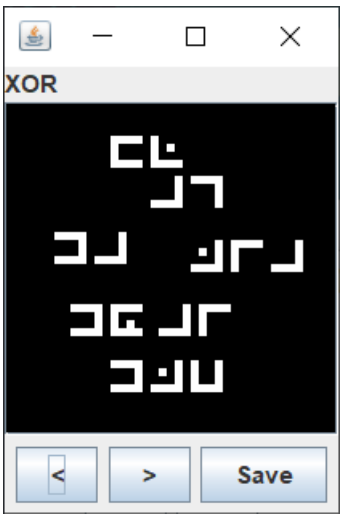
1.png

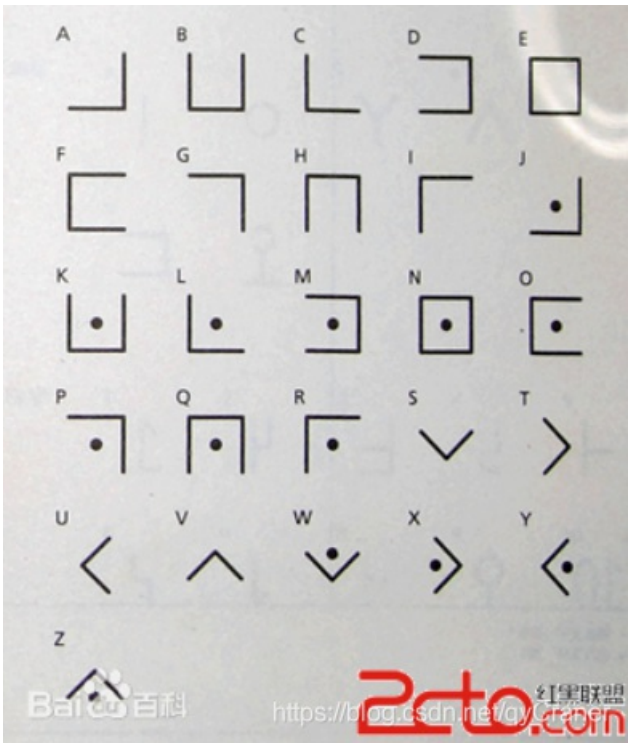
开局一张二维码，一直分离能分离出四张套娃的二维码，第二章二维码实际已经提示了到这里就要停下来了。（八神真好，处处都有提示）

第二张二维码用stegsolve查看最低通道，均能得到新的一张二维码图片，但没法扫码。
第二张图片原图：

最低位通道图：

两张图异或一下，即可发现酷似猪圈密码的字符：





解密后即可得到flag：

```
flag{dajiadoaidjb}
```

牛年大吉

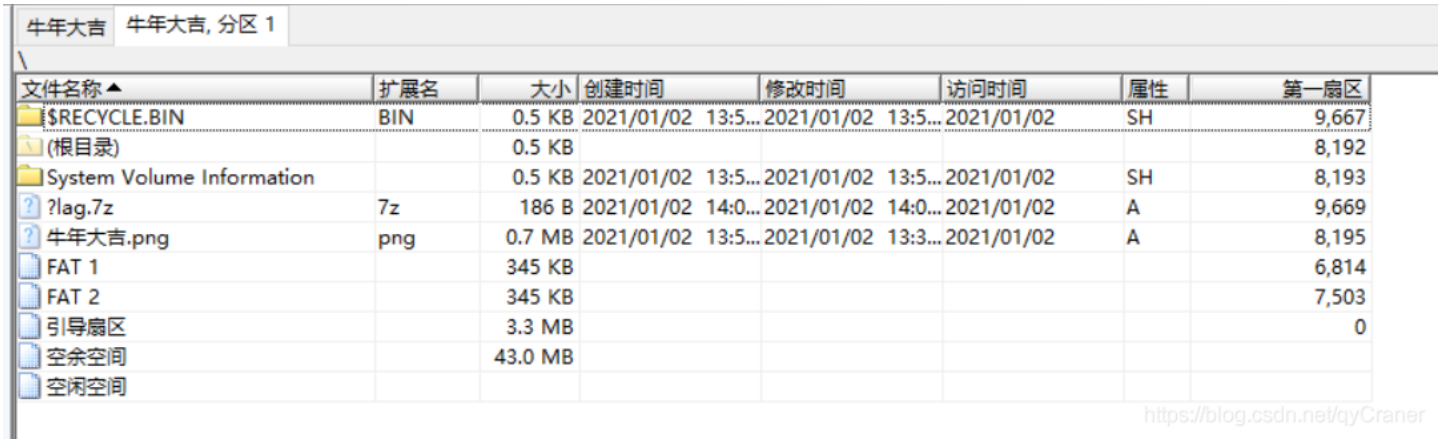
牛年大吉

100

题目下载 蓝奏云下载地址：
https://wws.lanzous.com/i1Ac0jybrvc 百度云下载地址：
https://pan.baidu.com/s/14EXw7U4w0Am0oP_xRXfbqQ 提取码：ns2k

感谢i_kei师傅供题

hint1：不要格式化哟，看看引导扇区是不是丢东西了
hint2：压缩包密码在图片文件头里
下载附件得到一个vhd文件，用winhex装载



牛年大吉 | 牛年大吉, 分区 1

\

文件名称	扩展名	大小	创建时间	修改时间	访问时间	属性	第一扇区
$RECYCLE.BIN	BIN	0.5 KB	2021/01/02 13:5...	2021/01/02 13:5...	2021/01/02	SH	9,667
(根目录)		0.5 KB					8,192
System Volume Information		0.5 KB	2021/01/02 13:5...	2021/01/02 13:5...	2021/01/02	SH	8,193
?lag.7z	7z	186 B	2021/01/02 14:0...	2021/01/02 14:0...	2021/01/02	A	9,669
牛年大吉.png	png	0.7 MB	2021/01/02 13:5...	2021/01/02 13:3...	2021/01/02	A	8,195
FAT 1		345 KB					6,814
FAT 2		345 KB					7,503
引导扇区		3.3 MB					0
空余空间		43.0 MB					
空闲空间							

能看到有一个7z和一个png图片。7z需要密钥，密钥即为png文件头**89504E47**，猜，就硬猜。一开始hint2放错位置了，导致一直做不出来，后来有了hint2之后并不算难题。
解开7z压缩包即可得到flag：

```
flag{CTFshow_The_Year_of_the_Ox}
```

童话镇

童话镇

100

一曲童话镇，多少断肠人？

https://ctfshow.lanzous.com/iA8HFkn4q9c

感谢@阿狸师傅供题

View Hint

View Hint

View Hint

View Hint

hint1：离别

hint2：思念

hint3：爱

hint4：印象

开局一个mp3文件+四个感觉没用任何作用的提示。
把mp3文件binwalk一下得到一个加密的zip文件，爆破密码得 **67373**
得到两个txt文件，看起来很像是机器学习之类的。
学习样本，答案只有两种，0或1：

```
1   0   [-15, 15, 28, -1, -15, -1]
2   1   [16, -34, 23, -8, 13, -2]
3   0   [63, -37, 0, 62, 19, 25]
4   1   [50, 16, 52, 46, 5, -28]
5   1   [38, 3, 33, -44, -37, 39]
6   0   [-34, 15, -24, 1, 48, 1]
7   0   [-8, 10, 34, -39, -11, 21]
8   1   [46, -25, 2, -5, -6, 50]
9   0   [29, 11, -4, -10, -7, -11]
10  0   [62, -7, 40, 49, -40, 27]
11  1   [-39, 21, 42, 14, 3, 55]
12  1   [37, -28, -1, 54, 40, -36]
13  1   [-5, -17, -38, 31, -62, -64]
14  0   [18, 60, 44, -24, -13, 42]
15  0   [-38, 11, -35, -20, -4, 55]
16  0   [35, 27, 55, -32, 42, -17]
17  0   [58, 59, 8, 34, 8, 31]
18  0   [-36, 25, -5, -12, 6, 17]
19  1   [48, -53, 53, -48, 36, -18]
```


需要解密的密文：

```
1   [5, 64, 50, 34, 55, 18]
2   [-10, -8, -44, 20, -24, -19]
3   [-27, 63, -22, -31, -50, -9]
4   [43, -28, 53, 18, 41, -39]
5   [11, 25, -58, 7, -31, 0]
6   [7, -35, 48, -35, 3, 48]
7   [-55, -55, -5, -31, 34, 30]
8   [-8, 55, -57, -44, -15, -59]
9   [1, 23, -50, 1, 62, -37]
10  [56, -3, 35, -14, -52, 54]
11  [-36, -42, -11, -15, 3, -20]
12  [-47, -45, -21, -28, 42, 39]
13  [-3, -3, 44, 18, 49, 59]
```


利用knn算法来解，脚本如下：

```
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from ast import literal_eval
from PIL import Image
x_train = []
y_train = []
x_test = []
f1 = open("t.txt","r")
f2 = open("flag.txt","r")
while 1:
 s = f1.readline()
 if not s:
  break
 s = s.strip('\n')
 p1 = literal_eval(s.split("\t")[1])
 p2 = literal_eval(s.split("\t")[0])
 x_train.append(p1)
 x_test.append(p2)
 #print(x_train)
 #print(x_test)
while 1:
 s = f2.readline()
 if not s:
  break
 s = s.strip('\n')
 s = literal_eval(s)
 y_train.append(s)
 #print(y_train)
x_train = np.array(x_train)
y_train = np.array(y_train)
x_test = np.array(x_test)

clf = KNeighborsClassifier(n_neighbors = 1)
clf.fit(x_train, x_test)
y_test = clf.predict(y_train)

f3 = open("3.txt","w")
for i in y_test:
 f3.write(str(y_test[i]))
f3.close()
```

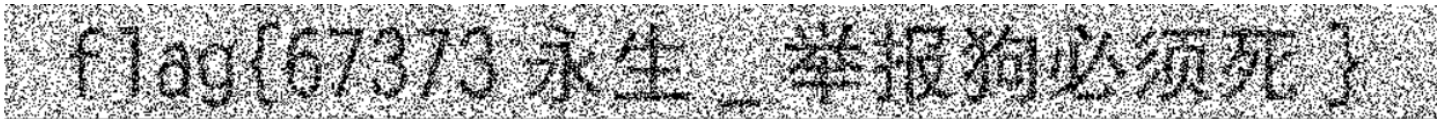
得到答案：

```
1   0100111000001000000001000101101100100011000100100100001000010000111001001101000000010000000100000001111
    1000001000100001100000010000000001000100100110001000110010000001111100011100011101010010000100010000
    0100000010001011000101001100000000000000011100010000000100010011001000110000011101101000001101000000010
    0100000000000001010001001000010000110111000000000000000000001000100100000001001000000000010011100101000
    0100000100010000010000000110100010101000011000000000010100000001001001100000000010100110000100000010000
    0000100001000011101010010001000011100000000010001100000001000000000000000000000001000010010001100001
    0001101010100000010001110010000000000000000011100100101010000001101000101000010110001010001010101011000
    0001000100101000000110101000001110000010010000001111010000000110000100001000101000100100000100100101010
    1001000000000011011000000010010001001001000100001000000001000000000000001000100001010000110001010000000
    0001000011010010001100010000000000010001000000000000000000000001000000000000010101001010100010101000100
    0000001100000100110110001001100001010000000000100100001000001000000000000011000010000100000000001000
    1000000000010011000000000000000011110000010000000000010000101001000000110001000000101010000001000011110
    0000001001100000000100100010000000000000001000010100000000000010111000000101000000001010001000100000010
    1000101011000000001000000001001010011111100000100000100000001110111000000000110000001000100011100000001
    0100001101011100101010001000111000010000110001110000001010000010000000000100000000100000000010000000000
    0010110001110011011010000101010001000100000000001000000000001001010001010110010001010000100000101001010
    0000000001101001000110000001000000111010000001000110000000110000101111001001000000000000000001000000000
    0100010100010100000001001000100001000000010001100001000001111000000000010110101001000010000000000000000
```

一共有78289个数据，78289正好等于79*991，且都是01字符，大概率就是一张图片了，写个脚本画图：

```
from PIL import Image
fp = open("3.txt","r").read()
pic = Image.new("L", (991, 79))
i = 0
for y in range(79):
 for x in range(991):
  if fp[i] == '0':
   pic.putpixel([x,y], 255)
  else:
   pic.putpixel([x,y], 0)
  i += 1
pic.show()
```

跑一下即可得到flag



色图生成器

色图生成器

100

> 欢迎使用色图生成器
> 已获取flag，正在为您生成色图.....
> 色图生成完毕，准备传输
> 正在传输色图.....
> ERROR！检测到屏蔽系统，传输被中断
> 准备为色图打码
> 正在生成马赛克.....
> 打码完成，准备添加冗余数据.....
> 添加完成，正在打包.....
> 打包完成，准备传输.....
> 传输完成，请点击下方链接下载您的色图

感谢@cheyenne师傅供题

View Hint

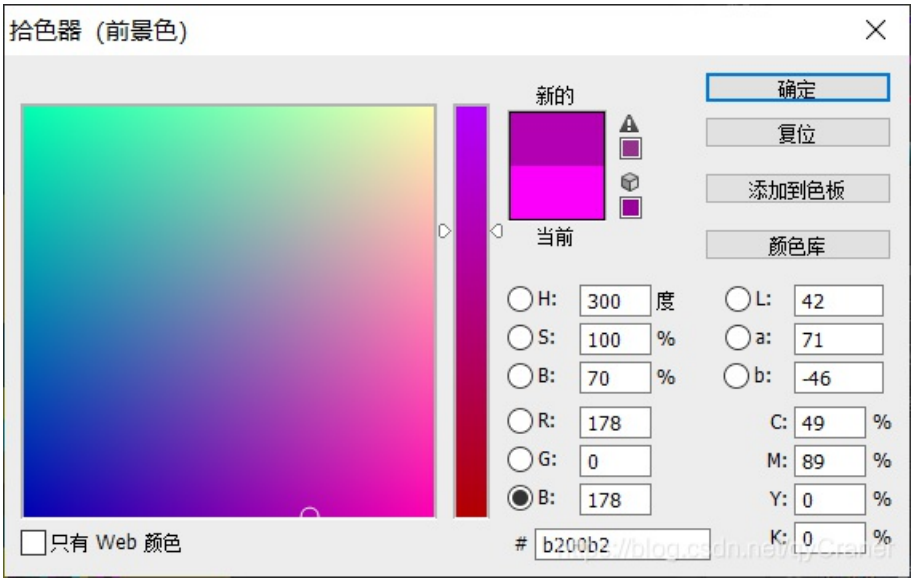
View Hint

View Hint

setu.zip

hint1：颜色很重要，但github更重要
hint2：第一步图片很重要，txt不重要
hint3：看一看马赛克部分的RGB值，有没有想到什么呢

下载附件得到一张图片，和一个txt文件，图片中间有一大块颜色各异的色块，根据hint3也能想到提取像素。每个色块的像素都是RGB中随机两个通道有同样的数据，另外一个为0

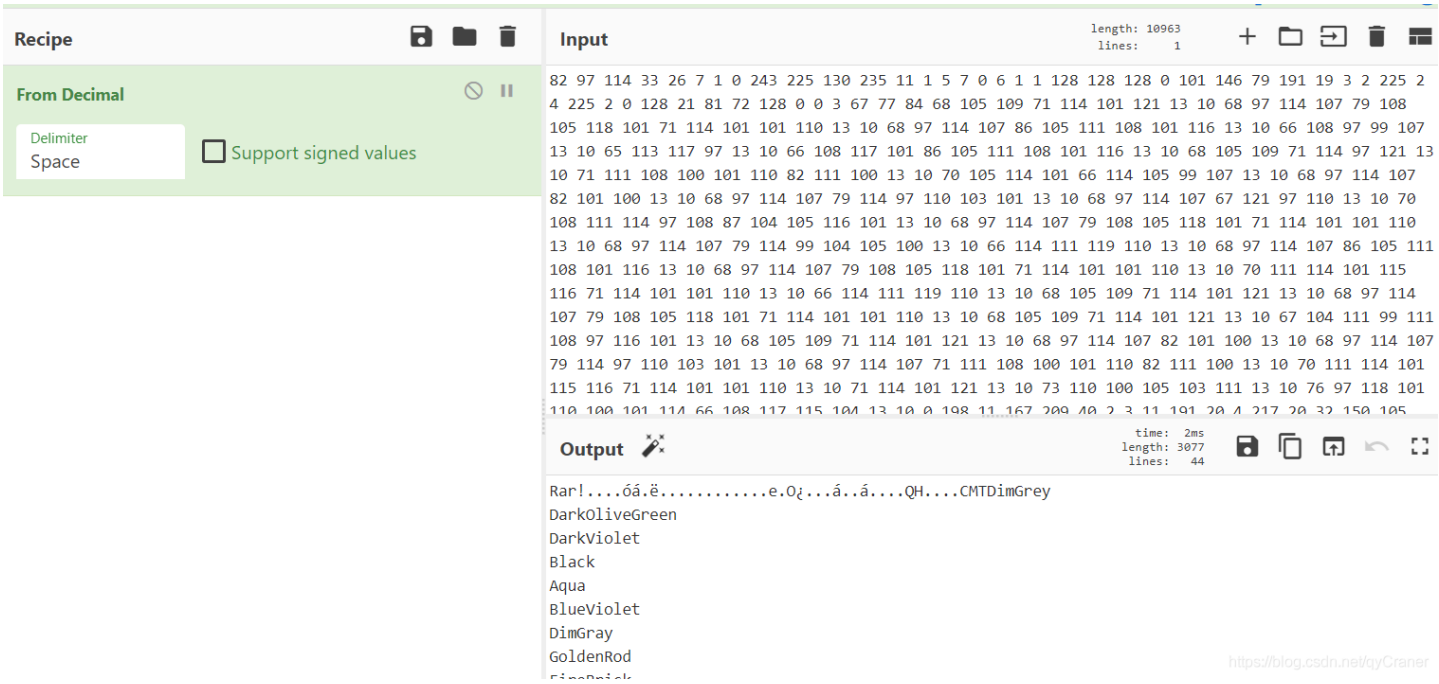


脚本如下：

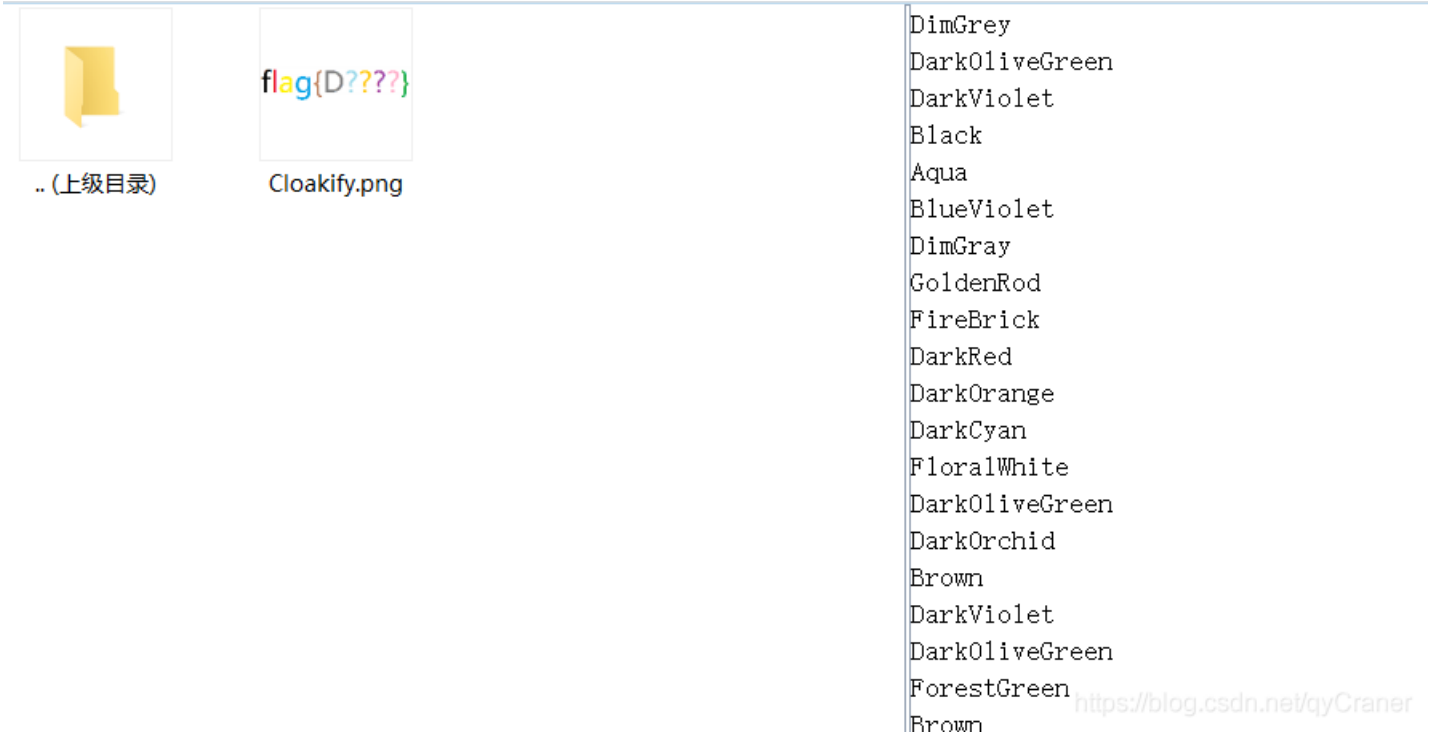
```
from PIL import Image
pic = Image.open('setu.png').convert('RGB')

fp = open("1.txt","w")
for y in range(0,17):
 for x in range(0,181):
  piv = pic.getpixel((50+5*x, 80+20*y))
  if piv[0] != 0:
   fp.write(str(piv[0]))
   fp.write(" ")
  else:
   fp.write(str(piv[1]))
   fp.write(" ")
fp.close()
```

得到一串十进制字符串，转码即可得到一个rar文件。



rar文件里有一张名为Cloakify.png的图片，备注里又有一串颜色数据。



将图片binwalk一下，得到一个加密的压缩包文件。
根据hint1，github搜索Cloakify，得到解密工具，字典为题目附件的colors.txt，密文为压缩包注释。

```
qiyue@ubuntu:~/Desktop/Cloakify-master$ python decloakify.py colors11 colors
D3arD4La0P1e45eD4iDa1Wo
```

解密得到压缩包密钥：**D3arD4La0P1e45eD4iDa1Wo**
解开得到一个pyc文件，在线反编译一下，得到：

```
from PIL import Image
import re, hashlib, random
flag = 'flag{jiu_bu_gao_su_ni}'
if re.fullmatch('^flag{[A-Z][0-9a-zA-Z]{4}}$', flag):
    m = hashlib.md5()
    m.update(flag.encode('ascii'))
    m = m.hexdigest()
    col = []
    for i in range(0, 24, 2):
        tmp = int(m[i:i + 2], 16)
        tmp += random.randint(-5, 5)
        col += [tmp]

    img = Image.new('RGB', (1024, 512))
    for i in range(4):
        timg = Image.new('RGB', (256, 512), tuple(col[i * 3:i * 3 + 3]))
        img.paste(timg, (i * 256, 0))

    img.save('C:/Users/Administrator/Desktop/setu.png')
```

这是对最初题目附件的setu.png的背景进行加密。
直接写个解密脚本一把梭：

```
import re
import hashlib
list = ['139', '102', '162', '24', '85', '57', '160', '37', '239', '200', '154', '30']
for a in range(48,123):
 for b in range(48,123):
  for c in range(48,123):
   for d in range(48,123):
    flag = 'flag{D' + chr(a) + chr(b) + chr(c) + chr(d) + '}'
    if re.fullmatch('^flag{[A-Z][0-9a-zA-Z]{4}}$', flag):
     m = hashlib.md5()
     m.update(flag.encode('ascii'))
     m = m.hexdigest()
     j = 0
     for i in range(0,24,2):
      p = int(list[j])
      if int(m[i:i+2], 16) - p > -5 and int(m[i:i+2], 16) - p < 5:
       j = j + 1
       continue
      elif i == 22:
       print(flag)
       break
      else:
       break
```

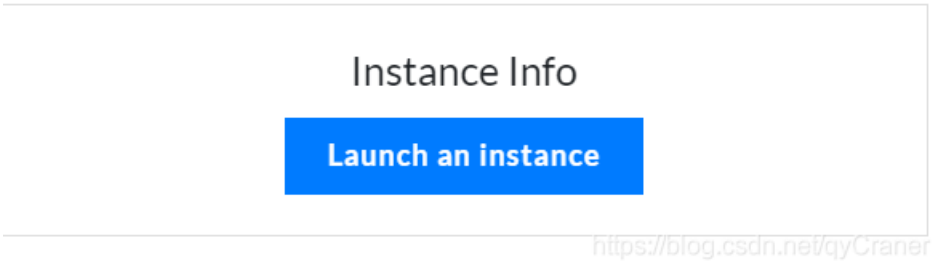
很快就能跑出来，跑出来即可得到flag：

```
flag{D4n1U}
```

拼图v2.0

有手就行，没手的可以拿眼睛去瞪

感谢@nimda师傅供题



手动拼图，一开始拼了一个小时只拼了90%，自动退出了很难受。第二次拼了50分钟得到flag：

碑寺六十四卦

碑寺六十四卦

100

这是从一处寺庙遗址中得到的碑文拓片，你能从中发现什么吗？ https://ctfshow.lanzous.com/iSFN4kn5jna

感谢@cheyenne师傅供题

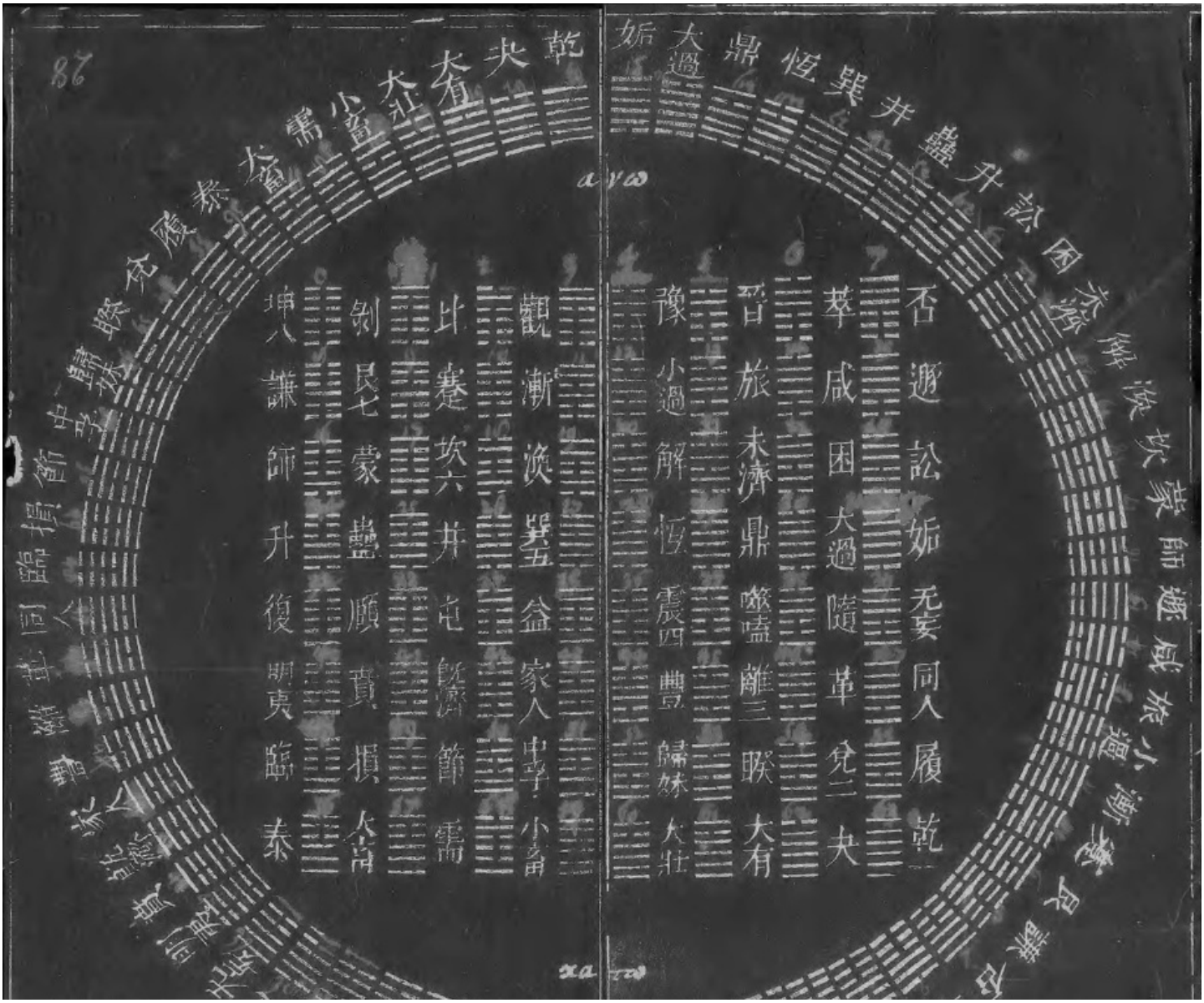
View Hint

View Hint

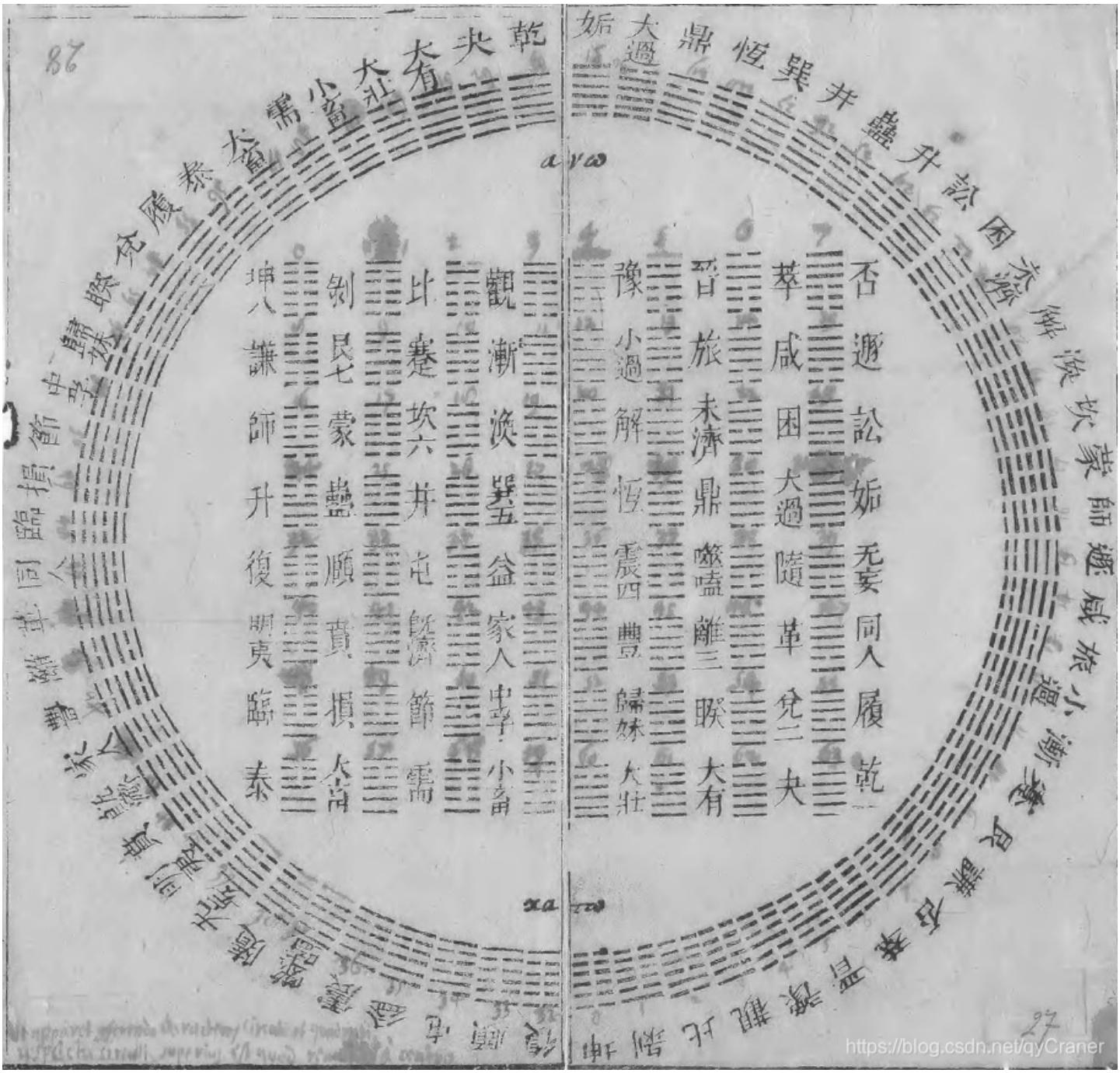
hint1：为什么碑文上空白的地方，拓片上却是黑黑一片呢？

hint2：如果说每个方块对应一个字符，可是替换表在哪里？

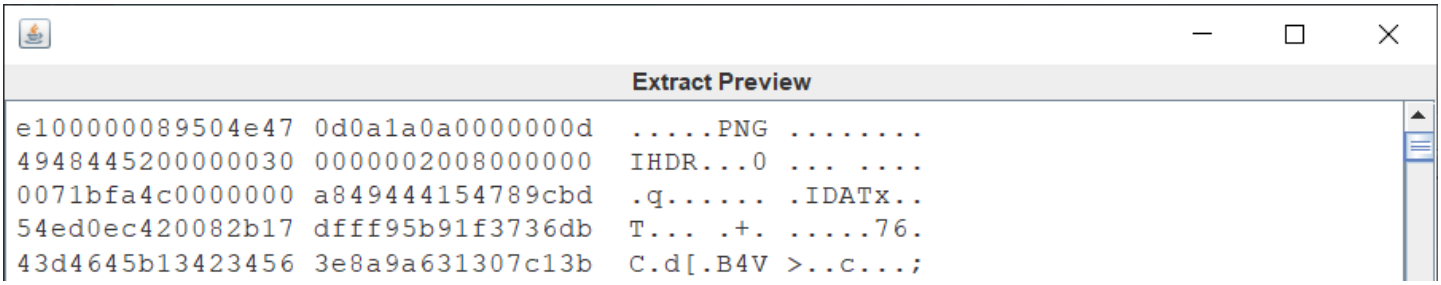
下载附件得到一张图，注：以下图都不是原图，原图请到官网自取

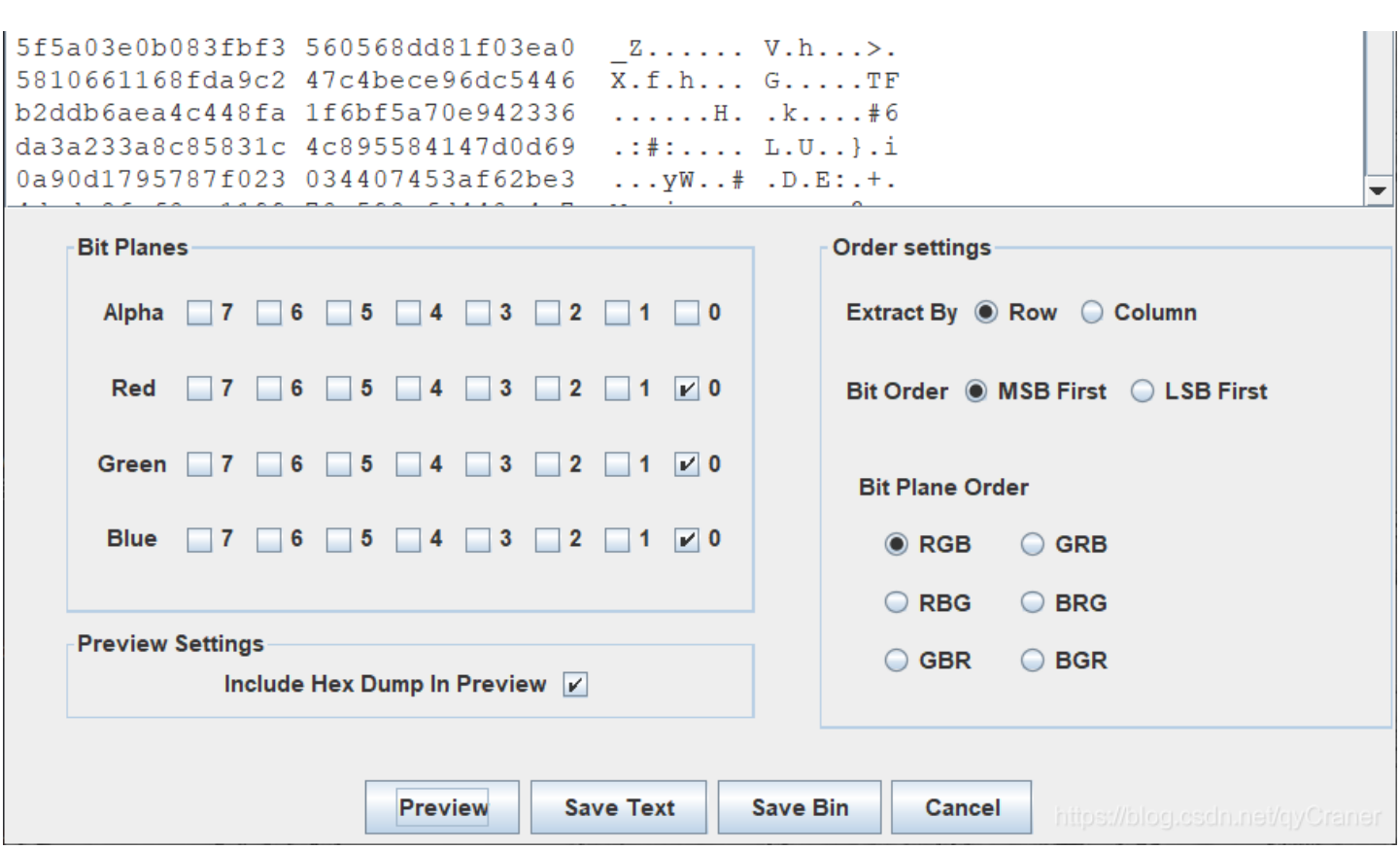


反色后得到：



反色后用stegsolve解一下最低位通道得到另一张图片：





对照原图的数据，一个一个进行对比，得到每个图案的数据为：

5,37,26,32,8,44,11,30,53,27,39,34,51,3,52,46,18,33,46,40,7,56,40

根据题目的64卦，一共有64个数据，联想到base64也是有64个数据，解密一下即可得到flag，脚本如下：

```
a = [5,37,26,32,8,44,11,30,53,27,39,34,51,3,52,46,18,33,46,40,7,56,40]
b = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'
flag = ''
for i in a:
 flag += b[i]
print(flag)
```

解得：

FlagIsLe1bnizD0uShuoH4o

最终flag为：

```
flag{Le1bnizD0uShuoH4o}
```

AA86

在一台旧电脑上（大约在16位操作系统还能跑的年代）发现了这个文件，挖掘它的秘密

感谢@ThTsOd师傅供题

2021/1/23 20:30 附件错了，sorry，已更新附件

View Hint

AA86.txt

hint1：请仔细阅读题目描述(5毛一条，去掉括号)
根据提示可得，关键是16位操作系统。其实只要安装一个16位操作系统这题直接就能跑出来，难点变成了如何安装16位操作系统。
谷歌 jjencode AA86 可得：

COM

Not the ActiveX thingy, but the old headerless binaries from DOS (and Windows XP or before)

aa86

Aa86 is a .COM file encoder by Yosuke Hasegawa that encodes binaries using only symbols characters, with a decoder.

so for example, a simple `hello world` binary is encoded as: `%@"%"@,~,%,!`_^[^_^]-;>`_^[^_^]%"!,^,:`_^[^_^]-@{-`{-?:`_[^_^]_-``-``-@@`_^[^_^]-`~-``-@$`_^[^_^]-``-``-@@`_^[^_^]-`~-``-@#`_^[^_^]-+~-/~-?;`_^[^_^]%!~-;-,;`_^[^_^]-"$-@~-@``_^[^_^]-{[-);-@:`_^[^_^]-/*,%`_^[^_^]`_^[^_^]`_^[^_^]`_^[^_^]%@$-@;-?;`_^[^_^]-/~-`&,#`_^[^_^]-`~-`{,*`_^[^_^]-@@-$!`_^[^_^]-:$,[,<`_^[^_^]-!|-.),!`_^[^_^]-@{-@`-/(`_^[^_^]`_^[^_^]`_^[^_^]`_^[^_^]-{!-{.,.`_^[^_^]-~/-/``_^[^_^]%""-}@$"`_^[^_^]%@@-!/,!`_^[^_^]-:*-=%`[[[[[[[[`^^^^^-%+)@@^^^!;@@_!,((,.((-$+)@*+@!!@-,!"(+@@,$-,!"($%&,&,&_&@"'%_&"',&$&!"-@*@$"`

- commented assembly source
- standalone encoder

.com后缀名文件可直接在16位操作系统执行，将txt里的数据保存，后缀名改为.com，传入MS-DOS虚拟机内，直接运行得到flag：

```
C:\>5.com
flag{https://utf-8.jp/public/sas/index.html}
C:\>_
```

64位系统只能运行64和32位的程序，理论上32位系统也能运行32位和16位的程序，一开始我试着装32位xp系统，但是装的有点慢还没试过。不知道有没有人是在32位系统下跑出来的flag，之后就看大佬们自己发挥了。
最终flag为：

```
flag{https://utf-8.jp/public/sas/index.html}
```