

CTF misc图片类总结（深育杯brige、西湖论剑YUSA的小秘密、湖湘杯leaker、wear_a_mask）

原创

[Hardworking666](#) 于 2021-11-24 13:26:38 发布 630 收藏 5

分类专栏: [CTF](#) 文章标签: [安全](#) [unctf](#) [python](#)

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/Hardworking666/article/details/121512575>

版权



[CTF 专栏收录该内容](#)

21 篇文章 2 订阅

订阅专栏

文章目录

一、[zlib_rar+exif+分析像素_zip](#)

二、[YCrCb通道隐写](#)

[2021“西湖论剑”网络安全大赛YUSA的小秘密](#)

[2020ByteCTF Hardcore Watermark 01](#)

三、[gif二维码](#)

四、[水印_矩阵代表一个字符的ASCII码](#)

五、[python 0 1转化为二维码](#)

六、[stegpy隐写爆破](#)

一、[zlib_rar+exif+分析像素_zip](#)

第一届深育杯misc, brige.png

使用binwalk分析出有zlib数据，但是无法使用binwalk -e或foremost分离出有效文件，在010editor中查看图片。

```
L-$ binwalk -e brige.png
DECIMAL      HEXADECIMAL  DESCRIPTION
-----
0            0x0         PNG image, 4000 x 1862, 8-bit/color RGB, non-interlaced
WARNING: Extractor.execute failed to run external extractor 'unstuff '%e%': [Errno 2] No such file or directory: 'unstuff', 'unstuff '%e%' might not be installed correctly
6847947     0x687DCB    StuffIt Deluxe Segment (data): fr
7966272     0x798E40    MySQL MISAM compressed data file Version 4
12676230    0xC16C86    Zlib compressed data, default compression
CSDN @Hardworking666
```

(1) 运行命令 pngcheck.exe -v xx.png，可以详细查看每个数据块的情况

```
D:\CTF\pngcheck-3.0.3-win32>D:\CTF\pngcheck-3.0.3-win32\pngcheck.win64.exe -v D:\桌面文件\其他\深育杯\misc1\brige.png
File: D:\桌面文件\其他\深育杯\misc1\brige.png (12676416 bytes)
```

```
chunk IDAT at offset 0xb107cb, length 8192
chunk IDAT at offset 0xbf27d2, length 8192
chunk IDAT at offset 0xbf47de, length 8192
chunk IDAT at offset 0xbf67ea, length 8192
chunk IDAT at offset 0xbf87f6, length 8192
chunk IDAT at offset 0xbfa802, length 8192
chunk IDAT at offset 0xbfc80e, length 7332
chunk IDAT at offset 0xbfe4be, length 100257
chunk zTXt at offset 0xc16c6b, length 193, keyword: Raw profile type APP1
chunk IEND at offset 0xc16d38, length 0
No errors detected in D:\桌面文件\其他\深育杯\misc1\brige.png (1538 chunks, 49.3% compression).
CSDN @Hardworking666
```

或者 (2) 010 editor中看到最后一个IDAT数据块长度异常，导出这段zlib数据。（编辑->复制为十六进制文本，粘贴到txt中）观察IDAT标识后面的87 9C两个字节，恢复成zlib数据头标识78 9C，写python3脚本将此段zlib数据解压缩，可得到一个rar压缩包。

注意解压缩的zlib数据应该是去掉IDAT-length、IDAT-type、IDAT-crc的数据部分，即只有（78 9C ... 60 A5 85 A2）。

| | | |
|-----------|---|--------------------|
| BF:E490h: | 28 0A 63 D4 F2 CA E2 95 2B 57 26 C3 C1 CF 7E F6 | (.c0dEa+W&AAI~8 |
| BF:E4A0h: | B3 ED DD 1E A1 62 10 66 50 4A A5 D6 B4 F2 A9 EC | 3iY.jb.fPJYÖ'ò@i |
| BF:E4B0h: | FF 03 68 00 C5 14 DB 05 8D 63 00 01 87 A1 49 44 | ÿ.h.Ä.Ü..c..#;ID |
| BF:E4C0h: | 41 54 87 9C DC BC 77 50 14 DF D3 37 AA 24 51 10 | AT+ceÛ4wP.ßÓ7ª\$Q. |
| BF:E4D0h: | 04 25 27 25 0A C2 12 95 20 C1 40 50 49 92 24 8B | .%'.%.Ä.· Á@PI'Š< |
| BF:E4E0h: | 48 56 96 25 48 0E FB 05 25 49 06 C9 49 41 32 82 | HV-#H.ú.%I.ÉIA2, |
| BF:E4F0h: | A4 DD 25 AE 64 90 1C 04 64 49 4B CE 39 2C F9 9D | =Y%@d...dIKÍ9,ù. |
| BF:E500h: | 59 9E E7 F9 FE FE 7B EF AD 7A EF AD 5B B7 CA 9A | Yžçùbb{i-zì-[·Êš |
| BF:E510h: | 62 BA FB 74 7F BA 4F CF 99 33 A7 7B D5 34 75 BC | b^ùt.°OÏ™3S{Ö4u¼ |
| BF:E520h: | C3 78 E5 F2 A5 17 F1 FC DE E4 97 89 49 2F 5D B9 | Àxáò¥.núPa-#I/]¹ |
| BF:E530h: | 7C 39 2E 25 F0 9F 4B 07 CB DE BA 37 09 09 6A 2F | 9.%òYK.ËP°7..j/ |
| BF:E540h: | 11 D5 5E BA 14 29 D5 37 F6 CF A5 4B 84 4F 55 B5 | .Ö^°.)õ78Ï¥K,,OUµ |
| BF:E550h: | E7 B2 FD E6 D3 52 66 6B BF CE A2 C2 16 D2 B3 96 | ç²yæÖRfk;îçÄ.ò³- |
| BF:E560h: | 7C 63 16 D1 89 F3 09 C5 B3 D5 FF CC 56 06 CD A2 | c.N#ó.Ä³ÖÿiV.íç |
| BF:E570h: | 52 16 B3 D0 1B 35 A1 73 11 00 25 6B 2E BC 7A 21 | R.³D.5;js..%k.¼z! |
| BF:E580h: | 25 04 F8 67 69 6B 6A 25 7C E7 D2 7C 6E 0C 2B 07 | %.øgikj% çÖ n.+. |
| BF:E590h: | 01 21 59 50 F2 15 A2 D0 1C 81 DB 4C EA 4C 82 FF | !YPò.çÐ.ÜLêL,ÿ |
| BF:E5A0h: | B0 5C 22 06 D9 22 D7 08 09 EA CA BE 55 B3 B6 0D | °\".ù"×..êËU³q. |

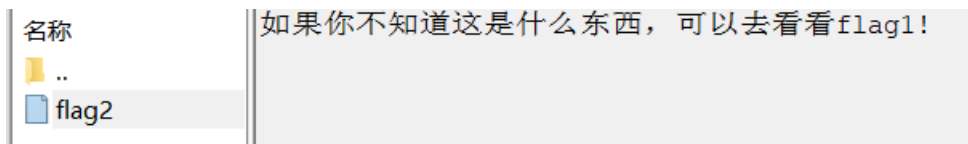
| 类型 | 值 | 名称 | 值 | 开始 | 大小 | 颜色 | 注释 |
|-----------|-----------------------|------------------------------|----------------|---------|--------|---------|----|
| 带符号字节 | 0 | struct PNG_CHUNK chunk[1518] | IDAT (Criti... | BDC74Ah | 200Ch | Fg: Bg: | |
| 无符号字节 | 0 | struct PNG_CHUNK chunk[1519] | IDAT (Criti... | BDE750h | 200Ch | Fg: Bg: | |
| 带符号短型 | 1 | struct PNG_CHUNK chunk[1520] | IDAT (Criti... | BE0762h | 200Ch | Fg: Bg: | |
| 无符号短型 | 1 | struct PNG_CHUNK chunk[1521] | IDAT (Criti... | BE276Eh | 200Ch | Fg: Bg: | |
| 带符号整型 | 100257 | struct PNG_CHUNK chunk[1522] | IDAT (Criti... | BE477Ah | 200Ch | Fg: Bg: | |
| 无符号整型 | 100257 | struct PNG_CHUNK chunk[1523] | IDAT (Criti... | BE6786h | 200Ch | Fg: Bg: | |
| 带符号 Int64 | 430601765405012 | struct PNG_CHUNK chunk[1524] | IDAT (Criti... | BE8792h | 200Ch | Fg: Bg: | |
| 无符号 Int64 | 430601765405012 | struct PNG_CHUNK chunk[1525] | IDAT (Criti... | BEA79Eh | 200Ch | Fg: Bg: | |
| 浮点 | 1.4049e-40 | struct PNG_CHUNK chunk[1526] | IDAT (Criti... | BEC7AAh | 200Ch | Fg: Bg: | |
| 双精度 | 2.12745539325208e-309 | struct PNG_CHUNK chunk[1527] | IDAT (Criti... | BEE7B6h | 200Ch | Fg: Bg: | |
| 半浮点 | 5.960464e-08 | struct PNG_CHUNK chunk[1528] | IDAT (Criti... | BF07C2h | 200Ch | Fg: Bg: | |
| 字符串 | | struct PNG_CHUNK chunk[1529] | IDAT (Criti... | BF27C8h | 200Ch | Fg: Bg: | |
| DOSDATE | | struct PNG_CHUNK chunk[1530] | IDAT (Criti... | BF47DAh | 200Ch | Fg: Bg: | |
| DOSTIME | 00:00:02 | struct PNG_CHUNK chunk[1531] | IDAT (Criti... | BF67E6h | 200Ch | Fg: Bg: | |
| FILETIME | 05/14/1602 09:09:36 | struct PNG_CHUNK chunk[1532] | IDAT (Criti... | BF87F2h | 200Ch | Fg: Bg: | |
| OLETIME | | struct PNG_CHUNK chunk[1533] | IDAT (Criti... | BFA7FEh | 200Ch | Fg: Bg: | |
| time_t | 01/02/1970 03:50:57 | struct PNG_CHUNK chunk[1534] | IDAT (Criti... | BFC80Ah | 1C80h | Fg: Bg: | |
| time64_t | | struct PNG_CHUNK chunk[1535] | IDAT (Criti... | BFE48Ah | 187ADh | Fg: Bg: | |
| | | struct PNG_CHUNK chunk[1536] | TEXT (Ancil... | C16C67h | CDh | Fg: Bg: | |
| | | struct PNG_CHUNK chunk[1537] | IEND (Criti... | C16D34h | Ch | Fg: Bg: | |

选定: 100269 [187ADh] 个字节的范围: 12575930 [BFE48Ah] 到 12676198 [C16C66h]

```
import zlib

data = open("zlib_hex_data.txt", 'r').read().replace(" ", "").replace("\n", "").strip()
data_dec = zlib.decompress(bytes.fromhex(data))
print(data_dec[:100])
with open("zlib_data.rar", 'wb') as wf:
    wf.write(data_dec)
# 结果: b'Rare!\x1a\x07\x01\x00J\x97,}\x0c\x01\x05\x08\x00\x07\x01\x01\x96\x9c\x87\x80\x00\xf7\xea}W\x13\x03\x02\xbd\x00\x04\xbd\x00\x00\x90:\xd1\xdc\x80\x00\x00\x03CMT\xe5\xa6\x82\xe6\x9e\x9c\xe4\xbd\xa0\xe4\xb8\x8d\xe7\x9f\xa5\xe9\x81\x93\xe8\xbf\x99\xe6\x98\xaf\xe4\xbb\x80\xe4\xb9\x88\xe4\xb8\x9c\xe8\xa5\xbf\xef\xbc\x8c\xe5\x8f\xaf\x8c\xe4\xbb\xa5\xe5\x8e\xbb\xe7\x9c\x8b'
```

解压压缩包可得flag2，注意压缩包中有提示请先获取flag1。



继续找flag1，分析最开始的那张图片，实际使用zsteg和zsteg可以发现其他可以信息。

ExifTool可以提取照片exif信息。可交换图像文件格式（英语：Exchangeable image file format，官方简称Exif），是专门为数码相机照片设定的，可以记录数码相机照片的属性信息和拍摄数据。

exiftool安装:

```
sudo apt install libimage-exiftool-perl
```

exiftool看到Copyright（版权）是十六进制，转换成文本：dynamical-geometry（动态几何），这个是后面的密钥。

```
bytes.fromhex('64796e616d6963616c2d6765666d65747279')  
# b'dynamical-geometry'
```

```
└─$ exiftool brige.png  
ExifTool Version Number      : 12.32  
File Name                    : brige.png  
Directory                   : .  
File Size                   : 12 MiB  
File Modification Date/Time  : 2021:10:24 00:41:16+08:00  
File Access Date/Time       : 2021:10:26 23:32:55+08:00  
File Inode Change Date/Time  : 2021:10:26 23:32:37+08:00  
File Permissions             : -rwxrwx-rw-  
File Type                   : PNG  
File Type Extension         : png  
MIME Type                   : image/png  
Image Width                 : 4000  
Image Height                : 1862  
Bit Depth                   : 8  
Color Type                  : RGB  
Compression                 : Deflate/Inflate  
Filter                      : Adaptive  
Interlace                   : Noninterlaced  
Warning                     : [minor] Text/EXIF chunk(s) found after PNG IDAT (may be ignored by some readers)  
Exif Byte Order              : Big-endian (Motorola, MM)  
X Resolution                 : 72  
Y Resolution                 : 72  
Resolution Unit             : inches  
Artist                      : w4nde3  
Y Cb Cr Positioning         : Centered  
Copyright                   : ©2021 by 64796e616d6963616c2d6765666d65747279  
Image Size                  : 4000x1862  
Megapixels                  : 7.4
```

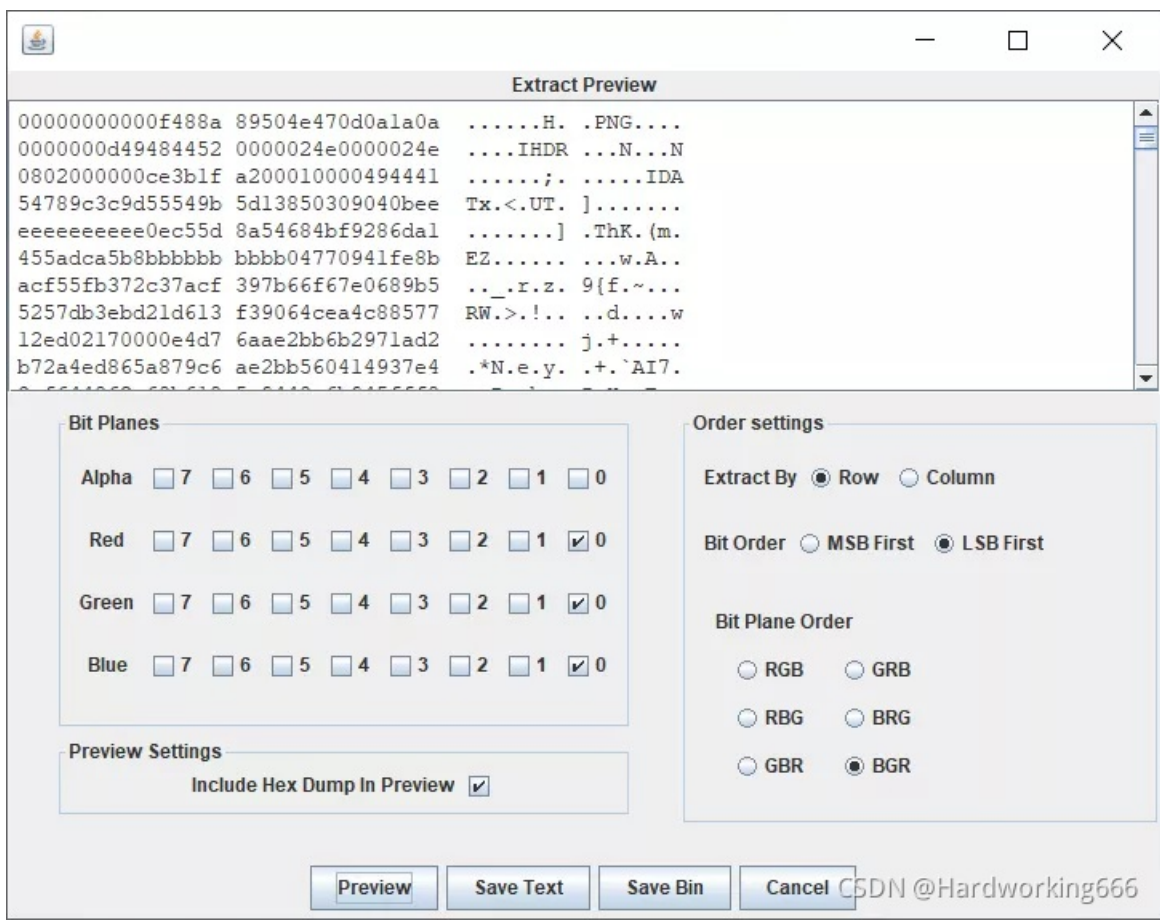
CSDN @Hardworking666

zsteg发现这张图片除了存在extradata外，在中也有脏数据。

```
└─$ zsteg brige.png  
[7] 100257 bytes of extra data after zlib stream  
extradata:0 ..  
00000000: 87 9c dc bc 77 50 14 df d3 37 aa 24 51 10 04 25 |...wP...7.$Q.%.|  
00000010: 27 25 0a c2 12 95 20 c1 40 50 49 92 24 8b 48 56 |'%.... .@PI.$.HV|  
00000020: 96 25 48 0e fb 05 25 49 06 c9 49 41 32 82 a4 dd |.%H...%I..IA2...|  
00000030: 25 ae 64 90 1c 04 64 49 4b ce 39 2c f9 9d 59 9e |%.d...dIK.9,..Y.|  
00000040: e7 f9 fe fe 7b ef ad 7a ef ad 5b b7 ca 9a 62 ba |...{..z..[...b.|  
00000050: fb 74 7f ba 4f cf 99 33 a7 7b d5 34 75 bc c3 78 |.t..0..3.{.4u..x|  
00000060: e5 f2 a5 17 f1 fc de e4 97 89 49 2f 5d b9 7c 39 |.....I/|.9|  
00000070: 2e 25 f0 9f 4b 07 cb de ba 37 09 09 6a 2f 11 d5 |%.K...7..j/..|  
00000080: 5e ba 14 29 d5 37 f6 cf a5 4b 84 4f 55 b5 e7 b2 |^..).7...K.OU...|  
00000090: fd e6 d3 52 66 6b bf ce a2 c2 16 d2 b3 96 7c 63 |...Rfk.....|c|  
000000a0: 16 d1 89 f3 09 c5 b3 d5 ff cc 56 06 cd a2 52 16 |.....V...R.|  
000000b0: b3 d0 1b 35 a1 73 11 00 25 6b 2e bc 7a 21 25 04 |...5.s...%k..z!%.|  
000000c0: f8 67 69 6b 6a 25 7c e7 d2 7c 6e 0c 2b 07 01 21 |.gikj%|..|n.+..!|  
000000d0: 59 50 f2 15 a2 d0 1c 81 db 4c ea 4c 82 ff b0 5c |YP.....L.L.L...|  
000000e0: 22 06 d9 22 d7 08 09 ea ca be 55 b3 b6 0d 5f fe |"...".....U..._|  
000000f0: 5c ff 44 eb a5 85 bc bc 26 17 fc 96 cb 1b 15 eb |\.D.....&.....|  
meta Artist .. text: "w4nde3"  
meta Raw profile type APP1.. text: "\ngeneric profile\n 164\nn457869660004d4d002a000000000006011a00050000  
000660213000300000001\n00010000829800020000002f000000e0000000000004800000010000004800000001\n77346e646533  
436353734373237390000\n"  
imagedata .. text: "*/->=>\n\r\n"  
b1,r,lsb,xy .. text: "|t$ljccV3"  
b1,g,lsb,xy .. file: raw G3 (Group 3) FAX, byte-padded  
b1,b,lsb,xy .. file: raw G3 (Group 3) FAX, byte-padded  
b1,bgr,lsb,xy .. zlib: data="\x01(\xB0Pnc\x9D\xB7\x93m\v\xE9\x17\xD5W\xA9h0\xD0\b\xF0\x10\x00\x00\x00\x00\x00  
\xF0\x1F\xE1#\xB8E\x17\xD2.8\x8Fq\xC8p\x90\xFB\n\xF6\x1E\xC4<\xF4\x19\xE7\xCAk\x95)\xAFQ\xDAK\xB51\x9Fa\x13\x  
b2,g,msb,xy .. file: 0420 Alliant virtual executable not stripped  
b2,b,lsb,xy .. text: "SUKWRDL"  
b2,rgb,msb,xy .. file: Apple DiskCopy 4.2 image @\001U\024E\021AD\004a\354\0143Es6\031\014I\220\356D\27
```

```
I:\210\002\026\025, 1628439617 bytes, 0x12550115 tag size, 0x37 encoding, 0x47 format
b2,bgr,msb,xy flag2-zlib. text: "z;{>/ko{n;;kE;K"
b3,r,msb,xy .. text: "]I1lK$d7"
b4,r,lsb,xy .. file: GeoSwath RDF
b4,r,msb,xy .. text: "b&\"&b.*\\"fb"
```

使用StegSolve检查隐写。



导出十六进制，这里不能直接打开图片，可使用foremost将PNG快速分离出来，最后得到一张590x590，大小为979KB的图片。

注意如果仅去掉PNG字符前数据并改后缀为PNG也能正常查看图片，但会阻塞下一步分析像素操作。到这里只有一张色彩值杂乱的PNG图片，分析其像素。

```
from PIL import Image
image = Image.open(r'bri.png')
allpixels = []
for x in range(image.width):
for y in range(image.height):
    allpixels.append(image.getpixel((x, y)))
print(len(allpixels)) # 348100
print(allpixels[:4])
# [(40, 176, 80), (37, 181, 75), (1, 253, 3), (2, 252, 4)]
#          0x50          0x4B          0x03          0x04
```

取前四个字节即可看出，像素第三列隐藏着压缩包十六进制，批量提取并保存成zip压缩包，使用前面得到的密码：dynamical-geometry解密，得到flag1文件。

```

from PIL import Image
image = Image.open(r'bri.png')
allpixels = []
for x in range(image.width):
for y in range(image.height):
if image.getpixel((x, y)) == (0, 0, 0):
continue
        allpixels.append(image.getpixel((x, y))[2])
hex_datalist = [str(hex(i))[2:].zfill(2) for i in allpixels]
print("".join(hex_datalist)[:100])
# 504b0304140009006300caa05753d904fdb22a4b0500dce856000f000b00666c6167312d61736369692e73746c0199070001

with open("outpur.txt", 'w') as wf:
    wf.write("".join(hex_datalist))

```

记事本打开文件后，是3D打印模型中的STL格式文件，STL格式分为ascii、binary格式，使用在线工具查看模型即可。使用在线网站查看stl，根据flag1的STL格式，将flag2也尝试用STL预览器查看：

<https://www.viewstl.com/#/>

📷 保存封面

SangFor{czzPnz-vyChYzOb

CSDN @Hardworking666

📷 保存封面

15cLjux00062qb2qT}

CSDN @Hardworking666

2021深育杯线上初赛官方WriteUp

深育杯misc

二、YCrCb通道隐写

2021“西湖论剑”网络安全大赛YUSA的小秘密

这题采用的YCrCb通道。通过 `cv2.cvtColor(img, cv2.COLOR_BGR2YCrCb)`对 `img` 图片数据进行色彩空间转换，即可得到三个通道的数据，然后对三个通道的数据分别根据奇偶做二值化处理并保存为图片。

```
from cv2 import *
import cv2 as cv
img=cv2.imread('C:\\Users\\XXX\\Desktop\\yusa\\yusa.png')
src_value=cv2.cvtColor(img, cv2.COLOR_BGR2YCrCb)
a, b, c = cv.split(src_value) #使用cv.split分离通道
cv.imwrite('a.png', (a % 2) * 255) #对三个通道中的数据分别根据奇偶做二值化处理，并分别保存为图片
cv.imwrite('b.png', (b % 2) * 255)
cv.imwrite('c.png', (c % 2) * 255)
```

运行后会得到三个通道的图片，在其中a.png即可清晰看到flag

2021“西湖论剑”其他wp

2020ByteCTF Hardcore Watermark 01

图片中每个像素可以通过三个值(通道)来表示，常见的是 R(red)G(green)B(blue) 模式。而本题用到的通道是 YCrCb。通过 `cv2.cvtColor(img, cv2.COLOR_BGR2YCrCb)`对 `img` 图片数据进行色彩空间转换，即可得到三个通道的数据：



对三个通道中的数据根据奇偶做二值化处理，也即判断数据的最低位：

```
dst_value = (src_value % 2) * 255
```

二值化后的数据分别代表二维码的黑和白，并且每个通道可得到部分二维码图片。最后只需将三个通道数据结合到一幅图中即可复现二维码。

如果不对三通道进行拆分的话，直接提取RGB通道会造成非常多的噪点，无法识别。但是可以搜到原图进行了diff，这样能减少大部分的噪点。再结合QRazyBox工具对二维码进行还原，也可以拿到flag的内容。

QRazyBox二维码拼接网站

Linux diff 命令

ByteCTF 2020 部分题目 官方Writeup

三、gif二维码

安恒月赛 DASCTF 7月赛QrJoker

题目是一个gif，都是只有一点点的二维码



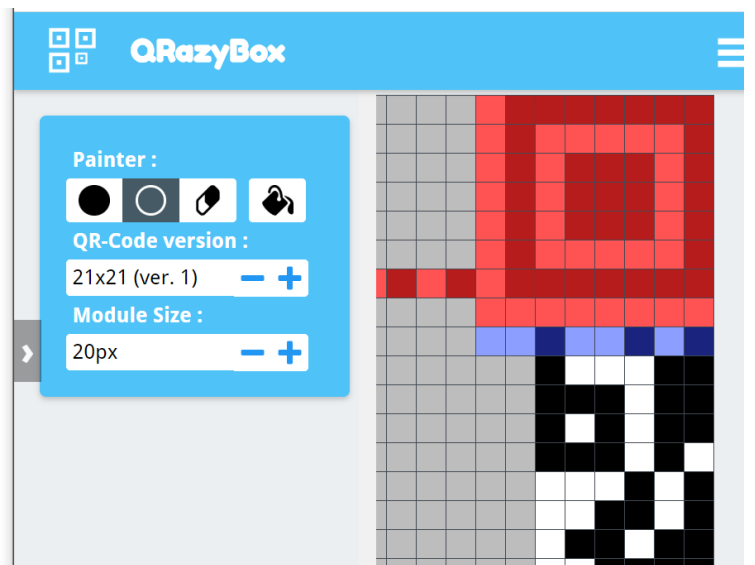
使用脚本切割一下：

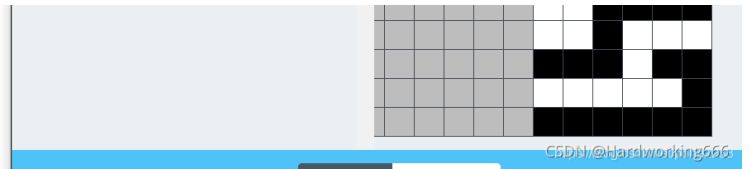
```
from PIL import Image
import os
gifFileName = 'QrJoker.gif'
#使用Image模块的open()方法打开gif动态图像时，默认是第一帧
im = Image.open(gifFileName)
pngDir = gifFileName[:-4]
#创建存放每帧图片的文件夹
os.mkdir(pngDir)
try:
    while True:
        #保存当前帧图片
        current = im.tell()
        im.save(pngDir+'/' +str(current)+'.png')
        #获取下一帧图片
        im.seek(current+1)
except EOFError:
    pass
```

<https://merricx.github.io/qrazybox/>

这个高是420 一个格子20 $420/20=21$

得出是21*21格子然后开始操作





CSDN/@Hjardworking666

QRazyBox

新 加載 保存 工具類 救命 關於

編輯器模式

畫家：

● ○ 🖌️ 🔗

QR碼版本：

21x21 (ver. 1) – +

模組尺寸

20px – +

工具清單

提取QR信息

強制解碼並儘可能獲取有關當前QR碼的信息

里德-所羅門解碼器

通過解碼Reed-Solomon塊進行糾錯和糾刪

蟹力格式信息模式

解碼時嘗試所有格式信息模式的可能性

數據屏蔽

使用Mask模式模擬數據屏蔽 (XOR)

填充位恢復

通過放置終止符和填充位來恢復丟失的位

數據序列分析 (實驗)

分析QR碼的數據序列

關

原始樣本：

加載樣本

歷史：

| |
|----|
| 畫家 |
| 畫家 |
| 畫家 |
| 畫家 |

CSDN/@Hjardworking666

在当中选择这个模式，就可以填充了

Original

Format Info Pattern

Top Right ▼

Error Correction Level: L M Q H

Mask Pattern : 0 1 2 3 4 5 6 7

Save
Cancel

list

pda

pda

look

QRazyBox

新 加載 保存 工具類 救命 關於

編輯器模式

畫家：

QR碼版本：
 21x21 (ver. 1) - +
 模組尺寸
 20px - +

工具清單

- 提取QR信息**
強制解碼並儘可能獲取有關當前QR碼的信息
- 里德-所羅門解碼器**
通過解碼Reed-Solomon塊進行糾錯和糾刪
- 蠻力格式信息模式**
解碼時嘗試所有格式信息模式的可能性
- 數據屏蔽**
使用Mask模式模擬數據屏蔽 (XOR)
- 填充位恢復**
通過放置終止符和填充位來恢復丟失的位
- 數據序列分析 (實驗)**
分析QR碼的數據序列

關

原始樣本：

加載樣本

歷史：
 更新格式信息模式
 洪水填充
 洪水填充
 更新格式信息模式
 更新格式信息模式

CSDN/@Hardworking566

錯誤糾正日誌：
 表演
 解碼錯誤：
 表演
 返回編輯

-----區塊1 -----
 Reed-Solomon Block:
 [32,54,179,38,132,96,0,236,17,51,51,51,204,204,204,195,51,204,195,51,51,51,204,51,204,51]
 綜合徵: [169,191,159,35,91,114,232,227,29,11]
 錯誤數量: 4
 錯誤位置多項式的係數: [252,5,20,183]
 錯誤位置: [9]
 誤差大小: 10101001

最終數據位:
 00100000001101101011001100100110100001000110000000000001110110000010001001100110011001100

[0010] [000000110] [110101100110010011010000100011000]
 模式指示器: 字母數字模式 (0010)
 字符數指示器: 0
 解碼數據: %56%6A
 最終解碼的字符串: %56%6A

CSDN/@Hardworking566

```
%56%6A%49%77%65%45%35%48%52%6B%64%69%4D%33%42%72%55%6A%4E%53%55%46%6C%58%4D%54%52%6A%4D%57%52%79%57%6B%5A%61%54%
31%5A%55%52%6C%5A%5A%56%57%51%30%56%32%31%57%63%6B%31%45%52%6C%56%4E%56%6B%70%78%56%47%78%56%4E%56%4A%58%53%6B%6
8%68%52%54%6C%58%54%56%5A%56%64%31%59%79%4D%58%64%69%4D%6B%5A%79%54%6C%52%61%55%6C%64%49%51%6D%46%57%61%32%52%50
%54%6C%5A%52%65%46%6F%7A%5A%44%46%56%56%44%41%35
解码一下
VjIweE5HRkdiM3BrUjNSUF1XMTRjMWRyWkZaT1ZUR1ZZVWQ0V21Wck1ER1VNVkpxVGxVNVJXSkhhRT1XTVZVd1YyMXdiMkZyT1RaUldIQmFwa2RP
T1ZReFozZFFVDA5
base64
V20xNGFGb3pkR3RPYW14c1drZFZOVTfVYud4WmVrMDFUMVJqTlU5RWJHaE9WMVUwV21wb2FrNTZRWHBaVkdONVQxZ3dQUT09
base64
Wm14aFozdGtOamxsWkdVNU1UaGxZek01T1RjNU9EbGhOV1U0Wmpoak56QXpZVGN5T1gwPQ==
base64
ZmxhZ3tkNjllZGU5MThlYzM5OTc5ODlhNWU4ZjhjNzAzYTcyOX0=
base64
flag{d69ede918ec3997989a5e8f8c703a729}
```

一个神仙的脚本

```

from PIL import Image
from Crypto.Util import number
import base64

dic = {0:'0',1:'1',2:'2',3:'3',4:'4',5:'5',6:'6',7:'7',8:'8',9:'9',10:'A',11:'B',12:'C',13:'D',14:'E',15:'F',38:
'%'}
res = ''
for num in range(64):
    p = Image.open('frame'+str(num+1)+'.bmp')
    a,b = p.size
    # print(a)
    data = []
    for y in range(100,b-10,10):
        d = []
        for x in range(410,470,10):
            if (y//10)%2 == 0:
                if p.getpixel((x,y)) >= 200:
                    d.append('1')
                else:
                    d.append('0')
            else:
                if p.getpixel((x,y)) >= 200:
                    d.append('0')
                else:
                    d.append('1')
        data.append(d)
    mode = data[11][5]+data[11][4]+data[10][5]+data[10][4]
    # print(mode)
    length = data[9][5]+data[9][4]+data[8][5]+data[8][4]+data[7][5]+data[7][4]+data[6][5]+data[6][4]+data[5][5]
    # print(length)
    d1 = data[5][4]+data[4][5]+data[4][4]+data[3][5]+data[3][4]+data[2][5]+data[2][4]
    d2 = data[1][5]+data[1][4]+data[0][5]+data[0][4]+data[0][3]+data[0][2]+data[1][3]+data[1][2]
    d3 = data[2][3]+data[2][2]+data[3][3]+data[3][2]+data[4][3]+data[4][2]+data[5][3]+data[5][2]
    d4 = data[6][3]+data[6][2]+data[7][3]+data[7][2]+data[8][3]+data[8][2]+data[9][3]+data[9][2]
    d5 = data[10][3]+data[10][2]+data[11][3]+data[11][2]+data[11][1]+data[11][0]+data[10][1]+data[10][0]
    d6 = data[9][1]+data[9][0]+data[8][1]+data[8][0]+data[7][1]+data[7][0]+data[6][1]+data[6][0]+data[5][1]
    d = d1+d2+d3+d4+d5+d6
    fin = d[:33]
    for i in range(0,len(fin),11):
        y = int(fin[i:i+11],2)%45
        x = int(fin[i:i+11],2)//45
        res += dic[x] + dic[y]
b64_data = number.long_to_bytes(int(res.replace('%',''),16))
while 1:
    b64_data = base64.b64decode(b64_data)
    if b'flag' in b64_data:
        print(b64_data)
        break

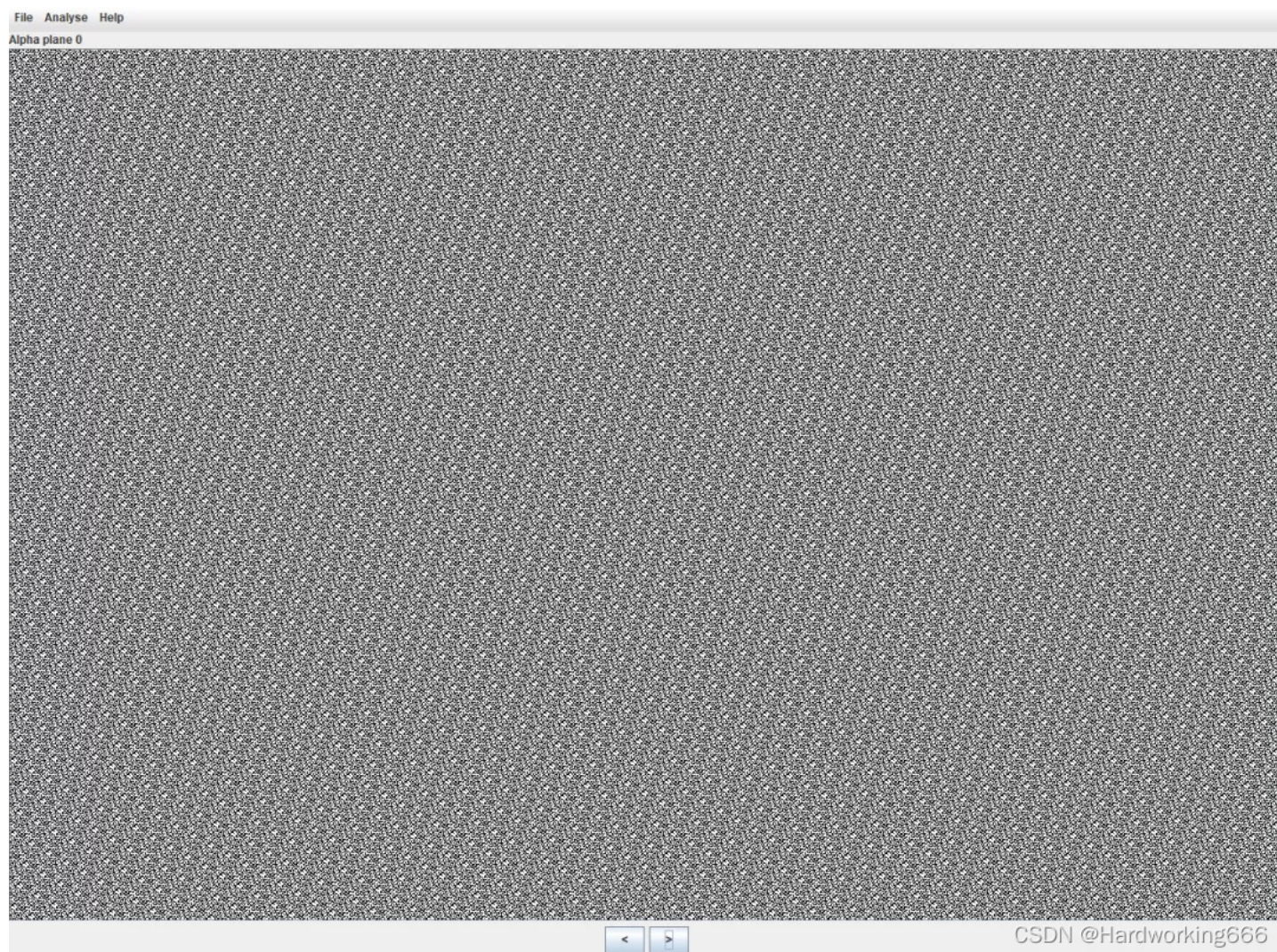
```

四、水印_矩阵代表一个字符的ASCII码

第七届“湖湘杯” leaker

解压题目附件，得到一张截图，用 Stegsolve 读取图片，发现图片是 RGBA 模式，而 Alpha 通道只有 255 和 254 两种取值，说明最低位有问题。

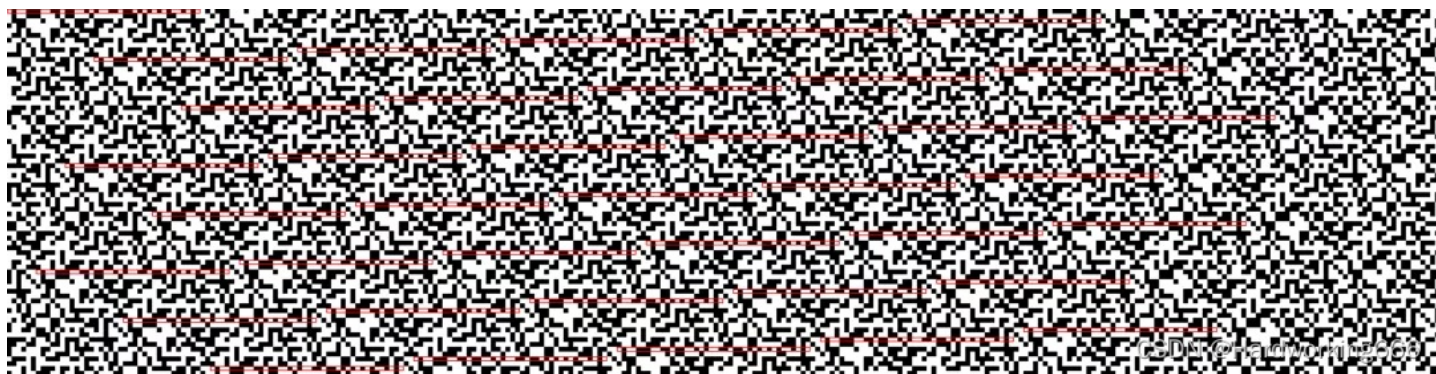
注：阿尔法通道（ α Channel或Alpha Channel）是指一张图片的透明和半透明度。可以表示256级的半透明度，因为阿尔法通道有8个比特可以有256种不同的数据表示可能性。当Alpha值为0时，该像素是完全透明的，而当Alpha值为255时，则该像素是完全不透明。



容易发现像素分布存在一定规律，先单独提取出来得到 01 矩阵。（把上面这个图保存后在MATLAB中打开看01值）

对于水印题，我们要做的事情就是找规律。一般来说，为了做到随机截取图片任意一块还能完整读取水印包含的信息，水印会将想要隐藏的信息重复填写，达到抗修改的效果，所以可以尝试先寻找数据的规律。

首先可以发现，行存在重复出现的规律，周期是 76 行，也就是说第 1 行的数据和第 77 行的数据是一样的。我们取出第 1 行所代表的 01 序列的前缀，大概取前 40 个 01 数据就好，然后在图中查找，发现这串 01 序列前缀同样出现在了第 3,5,7... 行中。通过分析我们可以发现数据隔两行就会重复一次，因此我们可以将分析数据的范围缩小成两行。



然后我们再尝试查找这两行有没有什么重复的模式，发现这两行内出现了很多次 2x4 的 0 矩阵，而每两个 0 矩阵之间的信息都是一样的，这让我们又可以再将范围缩小。到这里我们就得到了完整的一份信息经过加密后的结果。

接着我们发现第一行每隔 4 位都为 0，可以猜测是 ASCII 码的最高位，于是猜测每 2x4 个矩阵代表一个字符的 ASCII 码。通过分析 ASCII 码的 01 分布规律，我们发现第二行第二列的格子上的 01 分布是不均匀的，因此我们可以猜测该地方为 ASCII 的第 4 位，再根据第二行第一列以及第一行第二列的 01 分布情况，结合 [0-9a-zA-Z] 的 ASCII 码在各个二进制位上的 01 分布情况，进行一一对应，最后推测出解读方法：

1 3 5 7

2 4 6 8

解读数据，得到一串 Base64 编码 ZmxhZ3tlZjNkZTlzYS02MTRhLTQ4NjYtYjZlYi0yNDk2MjBiYTk1ZWRR9，解码得到 flag。

注：Zmxh Base64解码为fla

```
from PIL import Image
import numpy as np
import random
import base64

im = Image.open('leaker.png')
im = np.asarray(im)
x, y, z = im.shape

print(im.shape)

c = []
for j in range(y):
    c.append(1 - im[0, j, 3] // 255)
    c.append(1 - im[1, j, 3] // 255)

c = ''.join([str(x) for x in c])
c = c[:c.find('00000000')]
flag = ''.join([chr(int(c[i:i+8], 2)) for i in range(0, len(c), 8)])

print(flag)
print(base64.b64decode(flag.encode()))
```

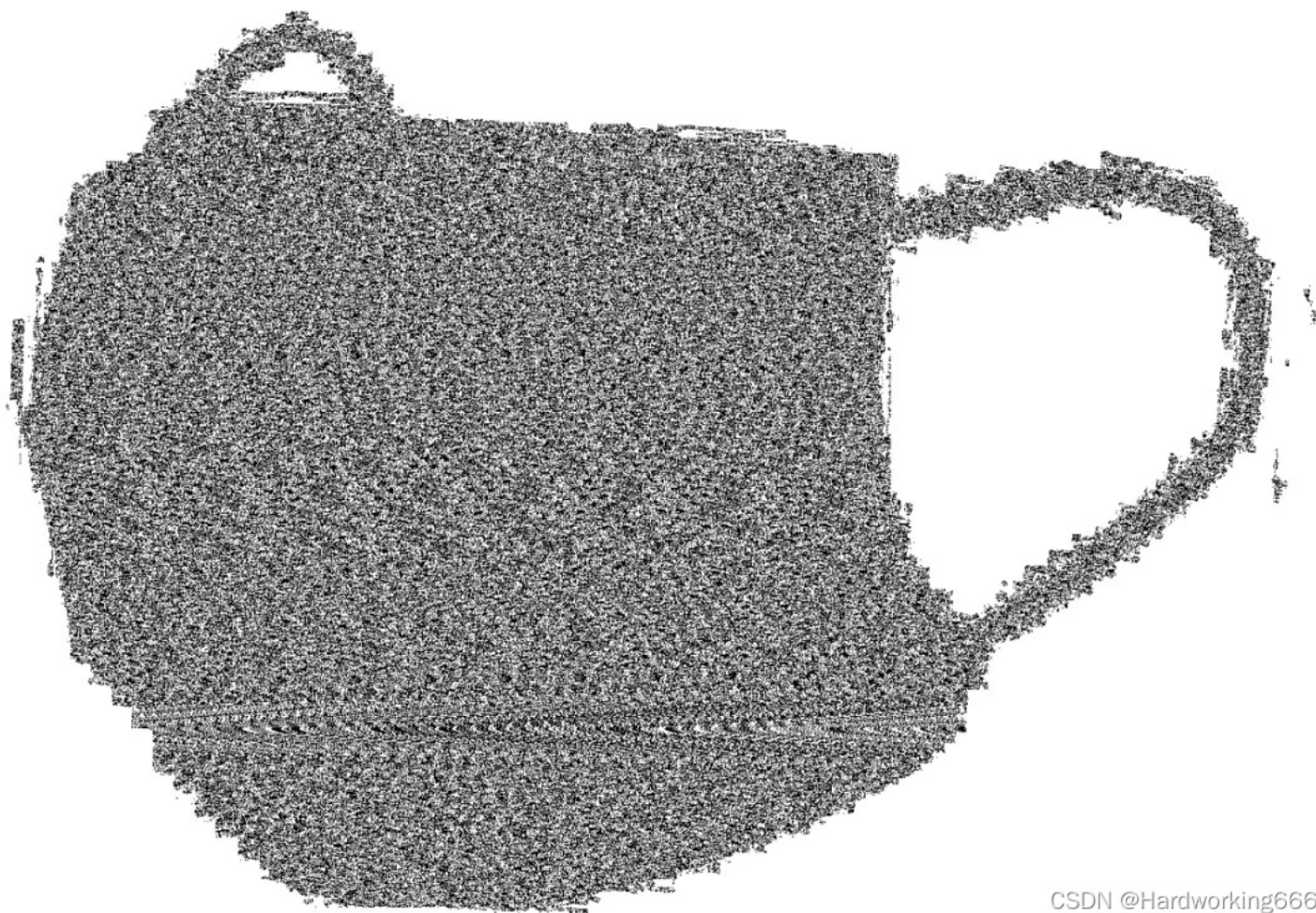
本题附件：

https://github.com/chunqiugame/cqb_writeups/raw/master/2021hxb/leaker_4a03eb590cb2db880820e52b475e3def.zip

五、python 0 1转化为二维码



首先通过 Stegsolve 查看各个颜色通道各个二进制位上的情况，发现 Blue 通道的最低位有问题。



CSDN @Hardworking666

推测数据隐藏在非全白的像素中，并被写入到了 Blue 通道的最低位。

通过观察可以发现口罩的下面有一段奇怪的花纹，呈现一个有规律的变动。



推测数据存在重复模式，通过研究花纹的重复规律可以发现，如果从上往下从左往右阅读数据，每 841 位就会重复一次。

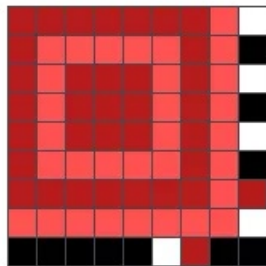
接着问题就是如何解读重复数据。将 841 质因数分解可以得到 29×29 ，所以尝试将其转成正方形的形状，可以得到一个二维码。



但是这个二维码并不能扫，因为此时的二维码是一个没有与模板（Mask）异或的二维码，所以扫描不出来。结合题目名称「wear_a_mask」，加上双关语「mask」的提示，根据 QRCode 格式所标出来的 Mask Mode 进行 XOR 变换。

Format Info Pattern

Top Left ▾



Error Correction Level:

L M Q H

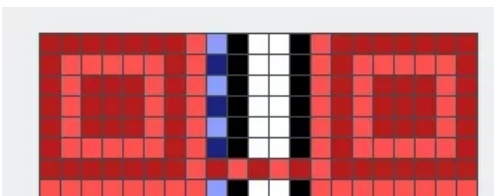
Mask Pattern :

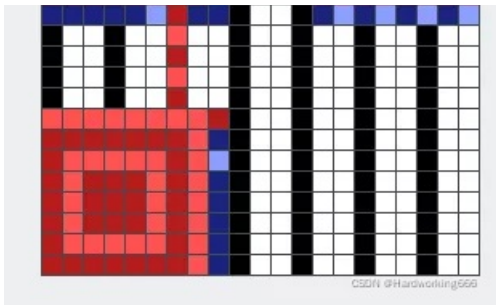
0 1 2 3 4 5 6 7

Save

Cancel

CSDN @Hardworking666

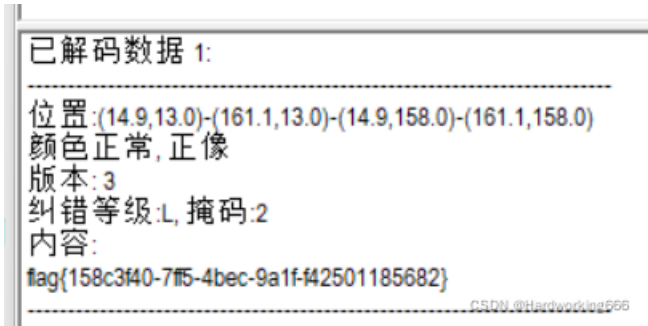




得到最后的 QRCode。





扫描得到 Flag。



本题附件: https://github.com/chunqiugame/cqb_writeups/raw/master/2021hxb/mask_a2c5c7556dc66ca474a412e4f90af3b9.zip

CTF python 0 1 转化为二维码:

通过stegsolve可以确定是颜色通道最低两位的隐写，而stegpy则是这种隐写方式。

A screenshot of a software interface showing a header bar labeled "Red plane 0". The rest of the image content is obscured by a black redaction box.A screenshot of a software interface showing a header bar labeled "Red plane 1". The rest of the image content is obscured by a black redaction box.

解密需要密码，可以写脚本交互爆破，同时根据图片名boom，也联想到爆破。

```
#!/usr/bin/env python3
# Module for processing images, audios and the Least significant bits.

import numpy
from PIL import Image
from . import crypt

MAGIC_NUMBER = b'stegv3'

class HostElement:
    """ This class holds information about a host element. """
    def __init__(self, filename):
        self.filename = filename
        self.format = filename[-3:]
        self.header, self.data = get_file(filename)

    def save(self):
        self.filename = '_' + self.filename
        if self.format.lower() == 'wav':
            sound = numpy.concatenate((self.header, self.data))
            sound.tofile(self.filename)
        elif self.format.lower() == 'gif':
            gif = []
            for frame, palette in zip(self.data, self.header[0]):
                image = Image.fromarray(frame)
                image.putpalette(palette)
                gif.append(image)
            gif[0].save(self.filename, save_all=True, append_images = gif[1:], loop=0, duration=self.header[1])
        else:
            if not self.filename.lower().endswith(('png', 'bmp', 'webp')):
                print("Host has a lossy format and will be converted to PNG.")
                self.filename = self.filename[:-3] + 'png'
            image = Image.fromarray(self.data)
            image.save(self.filename, lossless=True, minimize_size=True, optimize=True)
        print("Information encoded in {}.".format(self.filename))

    def insert_message(self, message, bits=2, parasite_filename=None, password=None):
        raw_message_len = len(message).to_bytes(4, 'big')
        formatted_message = format_message(message, raw_message_len, parasite_filename)
        if password:
            formatted_message = crypt.encrypt_info(password, formatted_message)
        self.data = encode_message(self.data, formatted_message, bits)

    def read_message(self, password=None):
        msg = decode_message(self.data)
```

```

if password:
    try:
        salt = bytes(msg[:16])
        msg = crypt.decrypt_info(password, bytes(msg[16:]), salt)
    except:
        return("Wrong password.")

check_magic_number(msg)
msg_len = int.from_bytes(bytes(msg[6:10]), 'big')
filename_len = int.from_bytes(bytes(msg[10:11]), 'big')

start = filename_len + 11
end = start + msg_len
end_filename = filename_len + 11
if(filename_len > 0):
    filename = '_' + bytes(msg[11:end_filename]).decode('utf-8')

else:
    text = bytes(msg[start:end]).decode('utf-8')
    print(text)
    return

with open(filename, 'wb') as f:
    f.write(bytes(msg[start:end]))

print('File {} succesfully extracted from {}'.format(filename, self.filename))

def free_space(self, bits=2):
    shape = self.data.shape
    self.data.shape = -1
    free = self.data.size * bits // 8
    self.data.shape = shape
    self.free = free
    return free

def print_free_space(self, bits=2):
    free = self.free_space(bits)
    print('File: {}, free: (bytes) {:,}, encoding: 4 bit'.format(self.filename, free, bits))

def get_file(filename):
    ''' Returns data from file in a list with the header and raw data. '''
    if filename.lower().endswith('wav'):
        content = numpy.fromfile(filename, dtype=numpy.uint8)
        content = content[:10000], content[10000:]
    elif filename.lower().endswith('gif'):
        image = Image.open(filename)
        frames = []
        palettes = []
        try:
            while True:
                frames.append(numpy.array(image))
                palettes.append(image.getpalette())
                image.seek(image.tell()+1)
        except EOFError:
            pass
        content = [palettes, image.info['duration']], numpy.asarray(frames)
    else:
        image = Image.open(filename)
        if image.mode != 'RGB':
            image = image.convert('RGB')

```

```

        image = image.convert('RGB')
        content = None, numpy.array(image)
    return content

def format_message(message, msg_len, filename=None):
    if not filename: # text
        message = MAGIC_NUMBER + msg_len + (0).to_bytes(1, 'big') + message
    else:
        filename = filename.encode('utf-8')
        filename_len = len(filename).to_bytes(1, 'big')
        message = MAGIC_NUMBER + msg_len + filename_len + filename + message
    return message;

def encode_message(host_data, message, bits):
    ''' Encodes the byte array in the image numpy array. '''
    shape = host_data.shape
    host_data.shape = -1, # convert to 1D
    uneven = 0
    divisor = 8 // bits

    print("Host dimension: {:,} bytes".format(host_data.size))
    print("Message size: {:,} bytes".format(len(message)))
    print("Maximum size: {:,} bytes".format(host_data.size // divisor))

    check_message_space(host_data.size // divisor, len(message))

    if(host_data.size % divisor != 0): # Hacky way to deal with pixel arrays that cannot be divided evenly
        uneven = 1
        original_size = host_data.size
        host_data = numpy.resize(host_data, host_data.size + (divisor - host_data.size % divisor))

    msg = numpy.zeros(len(host_data) // divisor, dtype=numpy.uint8)

    msg[:len(message)] = list(message)

    host_data[:divisor*len(message)] &= 256 - 2 ** bits # clear last bit(s)
    for i in range(divisor):
        host_data[i::divisor] |= msg >> bits*i & (2 ** bits - 1) # copy bits to host_data

    operand = (0 if (bits == 1) else (16 if (bits == 2) else 32))
    host_data[0] = (host_data[0] & 207) | operand # 5th and 6th bits = log_2(bits)

    if uneven:
        host_data = numpy.resize(host_data, original_size)

    host_data.shape = shape # restore the 3D shape

    return host_data

def check_message_space(max_message_len, message_len):
    ''' Checks if there's enough space to write the message. '''
    if(max_message_len < message_len):
        print('You have too few colors to store that message. Aborting.')
        exit(-1)
    else:
        print('Ok.')

def decode_message(host_data):
    ''' Decodes the image numpy array into a byte array. '''
    host_data.shape = -1, # convert to 1D

```

```

bits = 2 ** ((host_data[0] & 48) >> 4) # bits = 2 ^ (5th and 6th bits)
divisor = 8 // bits

if(host_data.size % divisor != 0):
    host_data = numpy.resize(host_data, host_data.size + (divisor - host_data.size % divisor))

msg = numpy.zeros(len(host_data) // divisor, dtype=numpy.uint8)

for i in range(divisor):
    msg |= (host_data[i::divisor] & (2 ** bits - 1)) << bits*i

return msg

def check_magic_number(msg):
    if bytes(msg[0:6]) != MAGIC_NUMBER:
        print(bytes(msg[:6]))
        print('ERROR! No encoded info found!')
        exit(-1)

if __name__ == '__main__':
    message = 'hello'.encode('utf-8')
    host = HostElement('gif.gif')
    host.insert_message(message, bits=4)
    host.save()

```

此脚本为原python库 stegpy脚本修改，主要改动读取隐写信息函数read_message一段，命名为lsb2.py，放在stegpy库目录下/usr/local/lib/python3.6/dist-packages/stegpy
网上找常见的弱口令字典进行爆破

```

#coding=utf-8
from stegpy import lsb2

host = lsb2.HostElement('flag.png')
dic = open('password1.txt').readlines()

for i in range(len(dic)):
    tmp = host.read_message(dic[i][: -1])
    if tmp != "Wrong password.":
        break
    else:
        print(i, dic[i][: -1])

```

爆破得到密码是123123@@@

隐写内容是783d793c313030

解开压缩包得到一张图片，无法正常观看。很明显是crc校验错误，爆破图片的正确宽高。

第四届2021美团网络安全高校挑战赛MT-CTF线上初赛官方wp