ctf misc 图片题知识点

原创

z.volcano 已于 2022-02-03 14:56:53 修改 8742 收藏 78

分类专栏： # 笔记 文章标签： 安全

于 2021-04-24 12:44:20 首次发布

补充了一点东西

图片题

- 常规思路
  - 图片名及图片内容
    - bpg图片
  - exif信息
  - 修改图片宽高
    - png
    - jpg
    - gif
    - bmp
    - 爆破脚本
  - 查看文件的字节
    - 分析文件头文件尾
    - 看看字节流中有没有隐藏信息
  - 使用stegsolve
    - lsb隐写
    - 有密码的lsb隐写
  - gif帧数间隔隐写
  - apng帧数间隔隐写

- 工具隐写
  - SilentEye
  - OurSecret
  - S-Tools
    - 加密
    - 解密
  - jpg
    - JAVA盲水印
    - Jphswin.exe
    - Free_File_Camouflage
    - f5-steganography（F5隐写，需要passwd）
    - outguess （可需要passwd）
    - steghide
  - png & bmp
    - 普通盲水印
    - 频域盲水印
    - zsteg（lsb隐写）
    - PNGDebugger.exe（检查IDAT块）
    - tweakpng.exe
- 二维码
  - 二进制转二维码
  - 二维码的修复
  - Aztec Code

常规思路

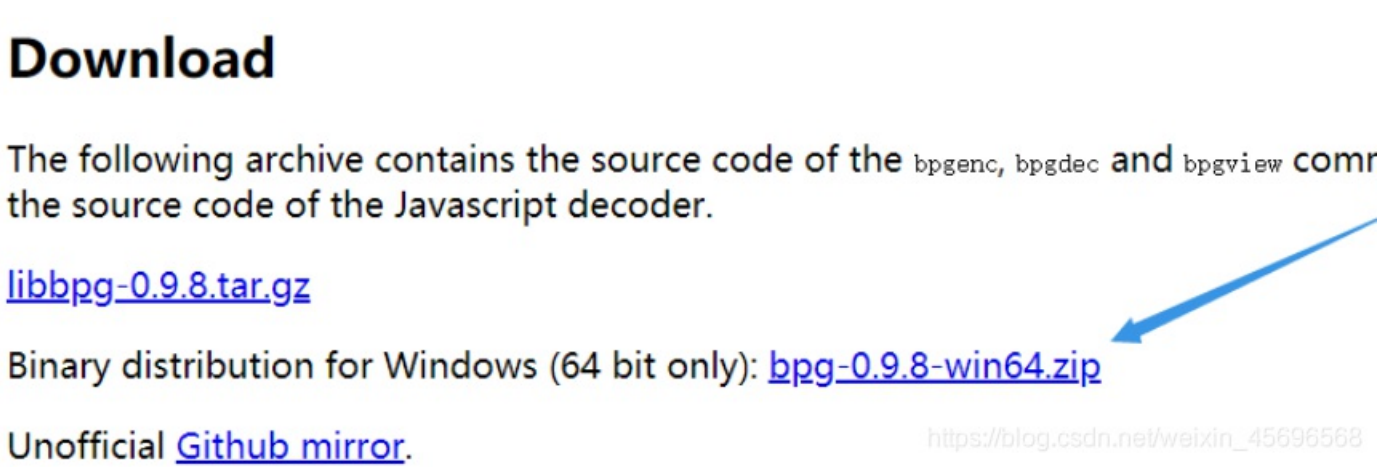
图片名及图片内容

除了题目描述和给的提示外，注意`图片名`和`图片里的内容`，很有可能是接下来要用到的`隐写工具或者加密方法的提示`。

例如`第四届蓝帽杯决赛`的一道misc：`MISC隐写`给了一张图片，里面是一条蛇，这个提示的是后面会用到的一种解密方法；然后从图片中提取出一个pdf文件，文件名是`no password.pdf`，也提示了需要用到工具，不需要输入密码。

bpg图片

bpg格式的图片在windows系统下是不能直接查看的，需要下载工具

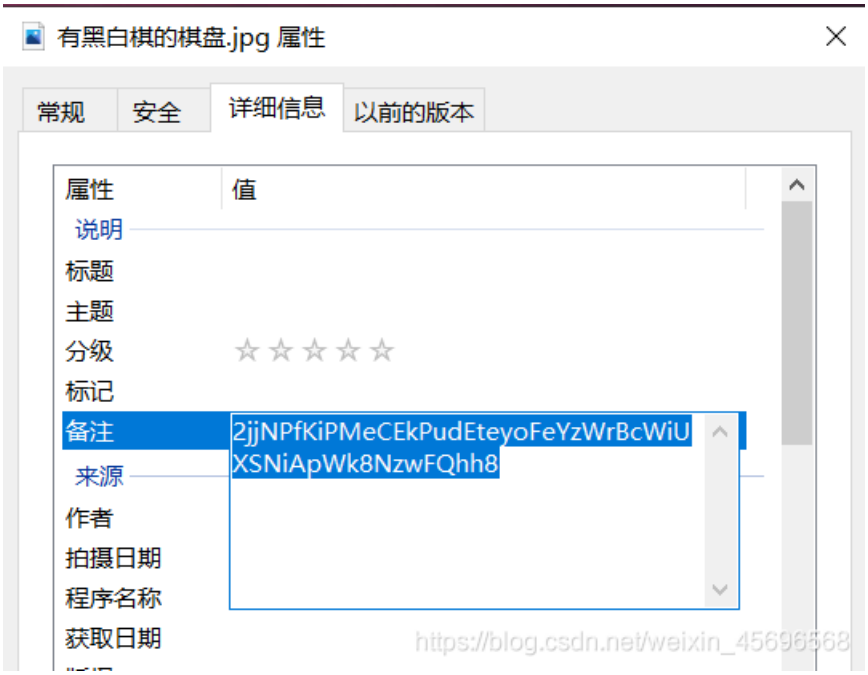


输入命令查看图片

exif信息

拿到一个图片时，建议查看一下它的exif信息，可能会有意想不到的收获

出题人经常在图片的exif信息中藏flag或者提示信息，这里以bugku的misc题目`有黑白棋的棋盘`为例



这里是直接右键查看属性，不过这样能获取的信息`十分有限`；

ctfshow中八神出的`misc入门`的`misc18-21`考察的都是这个知识点，具体可以看我另一篇博客。

拿其中的`misc20`为例，直接右键查看属性，翻看详细信息，得不到任何线索，推荐使用一个在线查看exif信息的网站，这里能得到的结果更详细，也可以使用exiftool

EXIF信息摘要

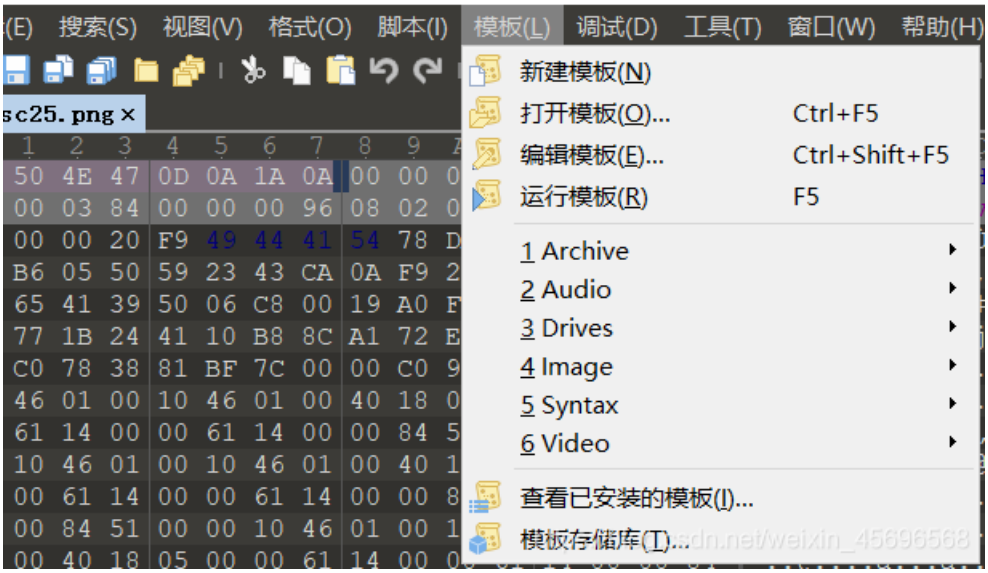
File

FileType	JPEG
FileTypeExtension	jpg
MIMEType	image/jpeg
ExifByteOrder	Big-endian (Motorola, MM)
Comment	这图片也太难看了。来自：西替爱抚秀大括号西九七九六四必一诶易西爱抚零六易一弟七九西二一弟弟诶弟五九三易四二大括号
ImageWidth	900
ImageHeight	150
EncodingProcess	Baseline DCT, Huffman coding
BitsPerSample	8
ColorComponents	3
YCbCrSubSampling	YCbCr4:2:0 (2 2)

修改图片宽高

很多时候，所给的图片的宽高，甚至crc32校验值都是被修改过的，需要我们去爆破得到正确的值。

这里推荐使用`010editor`，它的`模板功能`非常好用，我是010editor+winhex配合使用的。



png

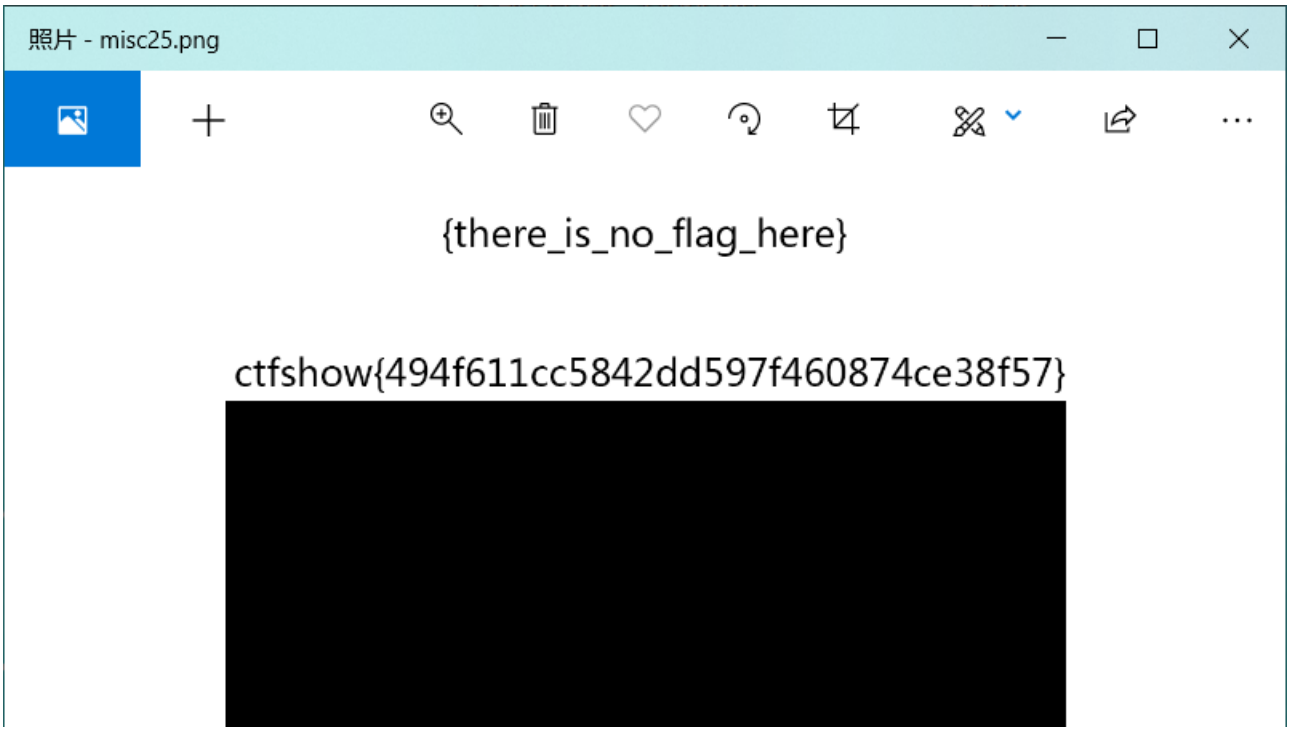
png图片修改宽高还是很容易的，这里是`png.bt`模板，框中的值分别是`宽高`和`crc值`，这里修改之后保存即可

```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
0000h: 89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52 ‰PNG........IHDR
0010h: 00 00 03 84 00 00 00 96 08 02 00 00 00 76 EC 1E ...„...–.....vì.
0020h: 40 00 00 20 F9 49 44 41 54 78 DA ED DD D9 95 E4 @.. ùIDATxÚíÝÙ•ä
0030h: 36 B6 05 50 59 23 43 CA 0A F9 20 1B 64 82 3C 90 6¶.PY#CÊ.ù .d‚<.
0040h: 05 65 41 39 50 06 C8 00 19 A0 FF FA CF 87 D5 5C .eA9P.È.. ÿúÏ‡Õ\
0050h: CD 77 1B 24 41 10 B8 8C A1 72 EF 8F 5E 6A 29 92 Íw.$A.¸Œ¡rï.^j)'
0060h: C1 C0 78 38 81 BF 7C 00 00 C0 93 FC A2 08 00 00 ÁÀx8.¿|..À"ü¢...
0070h: 10 46 01 00 10 46 01 00 40 18 05 00 40 18 05 00 .F...F..@...@...
0080h: 00 61 14 00 00 61 14 00 00 84 51 00 00 84 51 00 .a...a...„Q..„Q.
0090h: 00 10 46 01 00 10 46 01 00 40 18 05 00 40 18 05 ..F...F..@...@..
00A0h: 00 00 61 14 00 00 61 14 00 00 84 51 00 00 84 51 ..a...a...„Q..„Q
00B0h: 00 00 84 51 00 00 10 46 01 00 10 46 01 00 40 18 ..„Q...F...F..@.
00C0h: 05 00 40 18 05 00 00 61 14 00 00 61 14 00 00 84 ..@....a...a...„
00D0h: 51 00 00 84 51 00 00 10 46 01 00 10 46 01 00 40 Q..„Q...F...F..@
00E0h: 18 05 00 40 18 05 00 00 61 14 00 00 61 14 00 00 ...@....a...a...
00F0h: 84 51 00 00 84 51 00 00 84 51 00 00 10 46 01 00 „Q..„Q..„Q...F..
0100h: 10 46 01 00 40 18 05 00 40 18 05 00 00 61 14 00 .F..@...@....a..
0110h: 00 61 14 00 00 84 51 00 00 84 51 00 00 10 46 01 .a...„Q..„Q...F.
0120h: 00 10 46 01 00 40 18 05 00 40 18 05 00 00 61 14 ..F..@...@....a.
0130h: 00 00 61 14 00 00 61 14 00 00 84 51 00 00 84 51 ..a...a...„Q..„Q
0140h: 00 00 10 46 01 00 10 46 E1 D3 F8 F3 CF 3F 7F F9 ...F...FáÓøóÏ?.ù
0150h: AF BF FF FE 5B 81 F4 F8 E7 9F 7F BE 7C F9 52 4A ¯¿ÿþ[.ôøçŸ.¾|ùRJ
0160h: EC F7 DF 7F FF F1 E3 87 02 B9 CF F7 EF DF 63 13 ì÷ß.ÿñã‡.¹Ï÷ïßc.
```

模板结果 - PNG.bt

名称	值	开始	大小	颜色
struct PNG_SIGNATURE sig		0h	8h	Fg: Bg:
struct PNG_CHUNK chunk[0]	IHDR (Critical, Public, Unsafe to Copy)	8h	19h	Fg: Bg:
uint32 length	13	8h	4h	Fg: Bg:
union CTYPE type	IHDR	Ch	4h	Fg: Bg:
struct PNG_CHUNK_IHDR ihdr	900 x 150 (x8)	10h	Dh	Fg: Bg:
uint32 width	900	10h	4h	Fg: Bg:
uint32 height	150	14h	4h	Fg: Bg:
ubyte bits	8	18h	1h	Fg: Bg:
enum PNG_COLOR_SPACE_TY…	TrueColor (2)	19h	1h	Fg: Bg:
enum PNG_COMPR_METHOD c…	Deflate (0)	1Ah	1h	Fg: Bg:
enum PNG_FILTER_METHOD …	AdaptiveFiltering (0)	1Bh	1h	Fg: Bg:
enum PNG_INTERLACE_METH…	NoInterlace (0)	1Ch	1h	Fg: Bg:
uint32 crc	76EC1E40h	1Dh	4h	Fg: Bg:
struct PNG_CHUNK chunk[1]	IDAT (Critical, Public, Unsafe to Copy)	21h	2105h	Fg: Bg:
struct PNG_CHUNK chunk[2]	IEND (Critical, Public, Unsafe to Copy)	2126h	0h	Fg: Bg:

对于png，一般情况下，是把`高度改大`，看下面有没有内容



jpg

运用`jpg.bt`，可以很方便修改jpg的宽高

模板结果 - JPG.bt

名称	值	开始	大小	颜色	注释
struct JPGFILE jpgfile		0h	8991h	Fg: Bg:	
enum M_ID SOIMarker	M_SOI (FFD8h)	0h	2h	Fg: Bg:	
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	150	9Dh	2h	Fg: Bg:	
WORD X_image	900	9Fh	2h	Fg: Bg:	
ubyte nr_comp	3	A1h	1h	Fg: Bg:	
struct COMPS comp[3]		A2h	9h	Fg: Bg:	
struct DRI dri		ABh	6h	Fg: Bg:	
struct DHT dht		B1h	1A4h	Fg: Bg:	
struct SOS scanStart		255h	Eh	Fg: Bg:	
char scanData[34604]		263h	872Ch	Fg: Bg:	
enum M_ID EOIMarker	M_EOI (FFD9h)	898Fh	2h	Fg: Bg:	

对于jpg，一般情况下也是把`高度改大`，能看到图片下面的内容

gif

整体和前两种差不多，借助`gif.bt`，这里比较特殊的是，`每一帧都有独立的宽高`，因为不知道flag藏在哪一帧，一般`建议把所有帧的高度都改大`，然后用`Stegsolve`打开，翻看每一帧图片。

bmp

借助`bmp.bt`，整体和前面差不多，当宽度错误时，图片显示很乱

问题就是如何计算正确的宽度和高度，以ctfshow-misc入门的`misc24`为例，

目前是`900*150=135000`个像素大小，文件头占了53个字节，文件尾的位置在675053字节处(后面两个字节是windows的”补0”)，又因为每个像素点由3个字节（十六进制码6位）表示，每个字节负责控制一种颜色，分别为蓝（Blue）、绿（Green）、红（Red），所以文件真实的像素大小为：`(675053-53)/3=225000`

这题中给出了正确的宽度900，所以正确的高度就是`225000/900=250`

爆破脚本

png图片，`已知正确的IHDR块的CRC值`时，爆破宽度和高度

```
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'))
```

png图片，如果`IHDR块的CRC的值被修改过`，那就直接爆破，运行后会生成很多个图片，看一下哪个是正常的就行
这里是`已知高度的情况下爆破宽度`，根据自己需要修改脚本

```
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):  #界定宽度的范围
        name = str(i) + ".png"
        f1 = open(name,"wb")
        im = all_b[:16]+struct.pack('>i',i)+all_b[20:]
        f1.write(im)
        f1.close()
```

---

jpg图片，已知高度，`爆破宽度`
如果跑出来的图片看不到想要的东西，可能是狗出题人`把原本的高度调小了`，可以试一下把高度改大，再跑一遍试试

```
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): #界定宽度范围
        name = str(i) + ".jpg"
        f1 = open(name,"wb")
        im = all_b[:159]+struct.pack('>h',i)+all_b[161:]
        f1.write(im)
        f1.close()
```

---

gif图片，已知高度`爆破宽度`
如果跑出来没东西，试试把高度改高

```
import zlib
import struct
filename = "misc36.gif"
with open(filename, 'rb') as f:
    all_b = f.read()
    for i in range(920,951):
        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()
```

查看文件的字节

使用010editor和winhex对文件进行分析

分析文件头文件尾

如果图片打开错误，先看看文件头和文件尾出错没，如果有错误进行修改

JPEG (jpg)，
文件头：FFD8FF　　文件尾：FF D9

PNG (png)，
文件头：89504E47　文件尾：AE 42 60 82
GIF (gif)，
文件头：47494638　文件尾：00 3B
ZIP Archive (zip)，
文件头：504B0304　　文件尾：50 4B
TIFF (tif)，
文件头：49492A00
Windows Bitmap (bmp)，
文件头：424D

CAD (dwg)，
文件头：41433130

Adobe Photoshop (psd)，
文件头：38425053

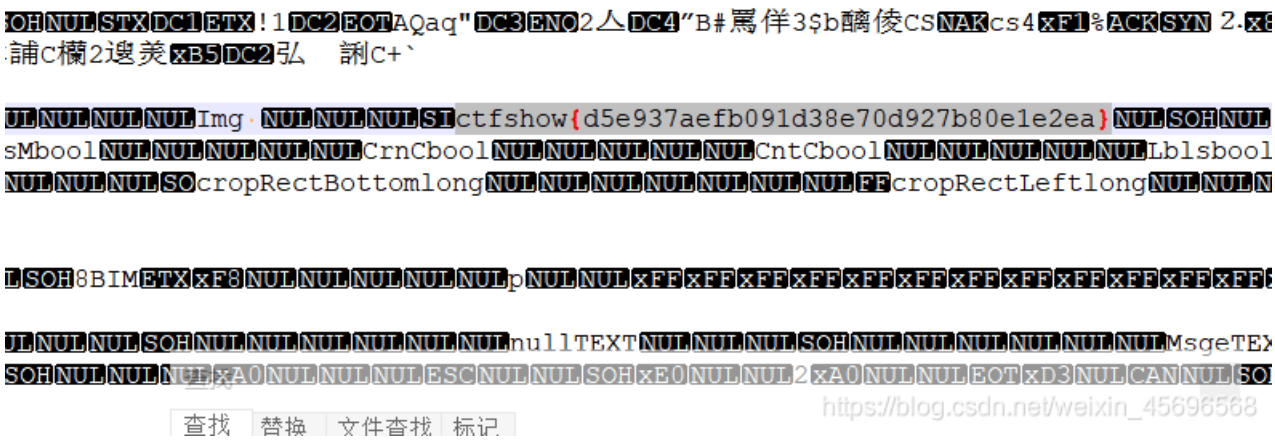
Rich Text Format (rtf)，
文件头：7B5C727466

XML (xml)，
文件头：3C3F786D6C

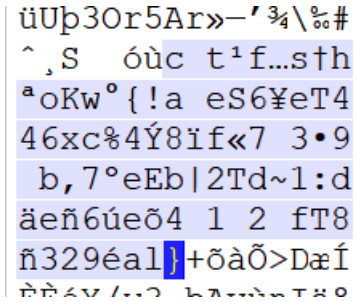
HTML (html)，
文件头：68746D6C3E
Email [thorough only] (eml)，
文件头：44656C69766572792D646174653A
Outlook Express (dbx)，
文件头：CFAD12FEC5FD746F
Outlook (pst)，
文件头：2142444E
MS Word/Excel (xls.or.doc)，
文件头：D0CF11E0
MS Access (mdb)，
文件头：5374616E64617264204A
WordPerfect (wpd)，
文件头：FF575043
Adobe Acrobat (pdf)，
文件头：255044462D312E
Quicken (qdf)，
文件头：AC9EBD8F
Windows Password (pwl)，
文件头：E3828596
RAR Archive (rar)，
文件头：52617221
Wave (wav)， 文件头：57415645
AVI (avi)， 文件头：41564920
Real Audio (ram)， 文件头：2E7261FD
Real Media (rm)， 文件头：2E524D46
MPEG (mpg)， 文件头：000001BA
MPEG (mpg)， 文件头：000001B3
Quicktime (mov)， 文件头：6D6F6F76
Windows Media (asf)， 文件头：3026B2758E66CF11
MIDI (mid)， 文件头：4D546864

看看字节流中有没有隐藏信息

推荐使用`winhex`或者`notepad`，打开后直接拉到底部，看有没有什么额外信息
然后直接ctrl+f搜索`key`、`flag`、`password`等字样，没准就拿到flag了



也有可能不是直接给到的，需要进行进一步的处理



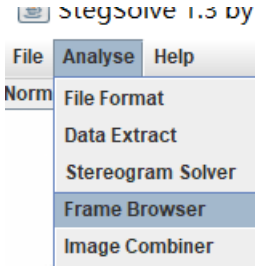
其实更常见的是，在尾部隐写另一个文件的数据，一般是一个压缩包，可以手动把数据提取出来，或者使用`foremost`或`binwalk`提取。

使用stegsolve

stegsolve是非常常用的工具，功能十分强大

用stegsolve打开图片后，建议`先把所有的图层都看一遍`，可能会有flag或者提示信息

如果是gif图片，可以使用`frame browser`查看所有帧

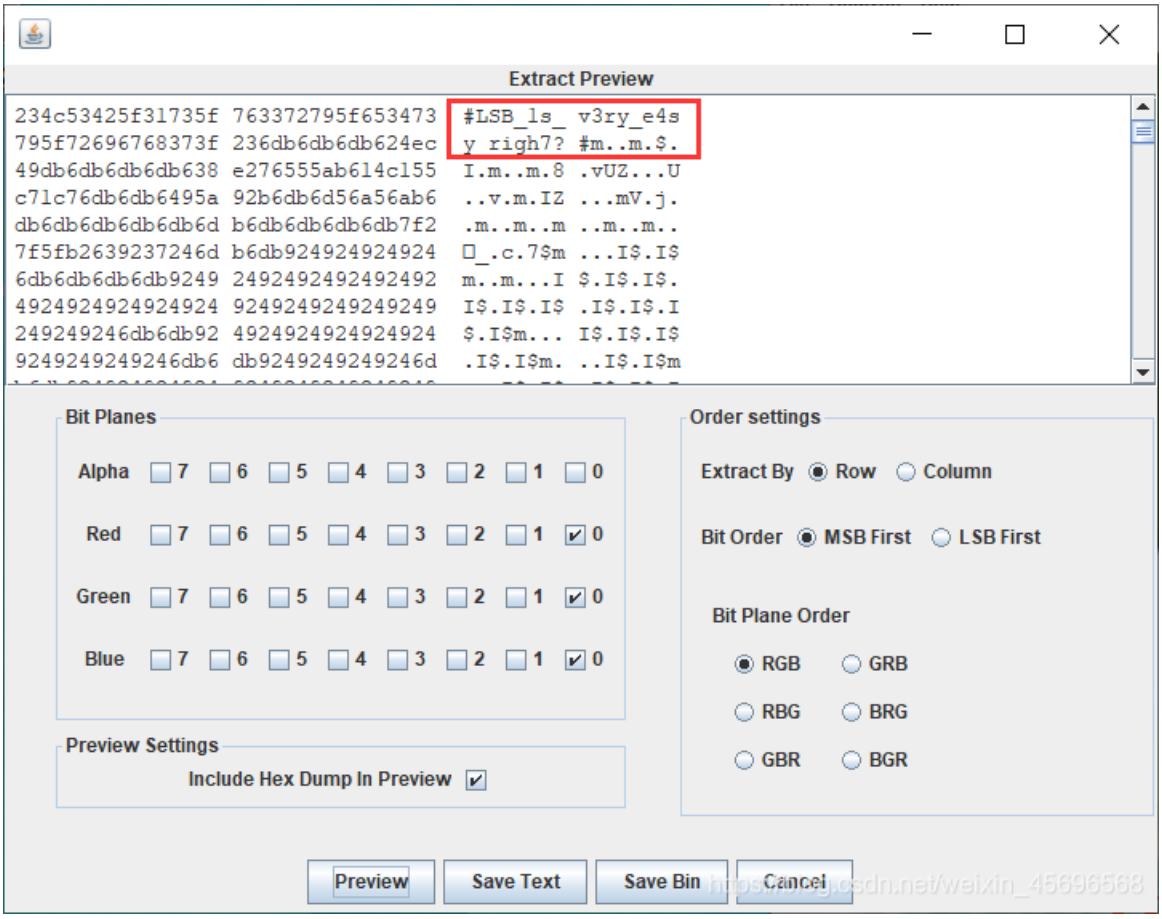


如果是两张很相似的图片，但是其中一张有很难以辨认的文字痕迹，可以用`image combiner`功能

lsb隐写

在翻看图层的时候，可能会发现lsb隐写的痕迹(不太好描述，做习惯了就知道)

使用stegsolve的`data extract`，选上对应的通道(比较常见的是如下配置)，就可以得到隐藏的信息



有密码的lsb隐写

项目地址：https://github.com/cyberinc/cloacked-pixel

这个项目是基于`python2`的，如果环境中同时有Py2和Py3需要注意下

```
加密
python2 lsb.py hide <img_file> <payload_file> <password>
解密
python2 lsb.py extract <stego_file> <out_file> <password>
```

gif帧数间隔隐写

使用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
```

提取之后会得到一系列数字，一般是由两种数字重复组成，其中一种转成1，另一种换成0，得到一长串二进制，后面一般是`转字符`或者`转二维码`

apng帧数间隔隐写

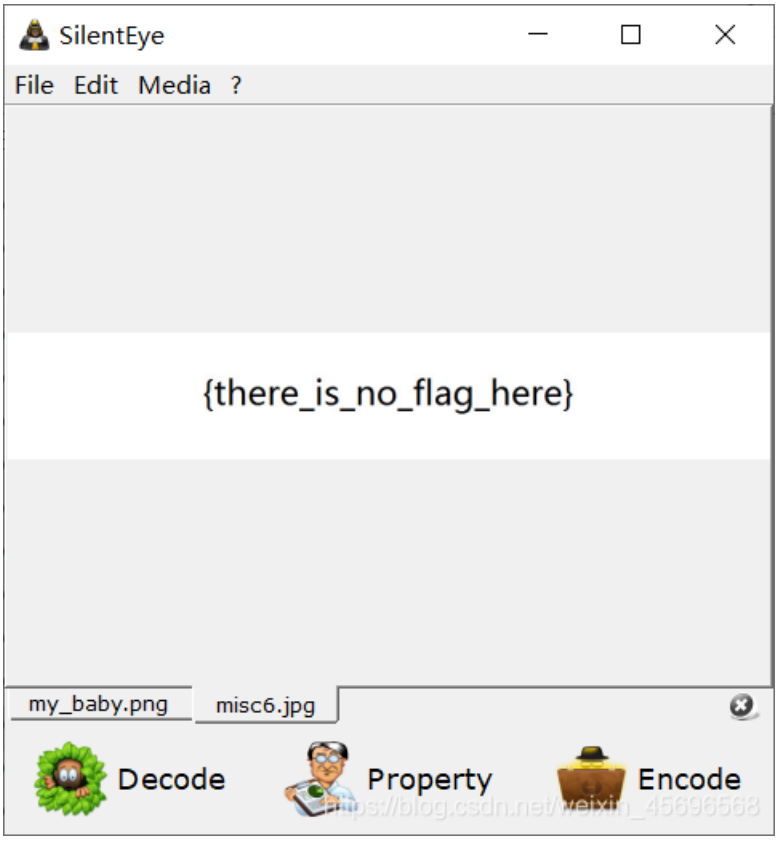
apng也是和gif一样会动的，使用工具`APNG Disassembler`提取出每一帧图片，和对应的详细信息，其中就包括`间隔时间`，注意使用这个工具的时候，提前把图片放在一个文件夹里...

后面步骤同上

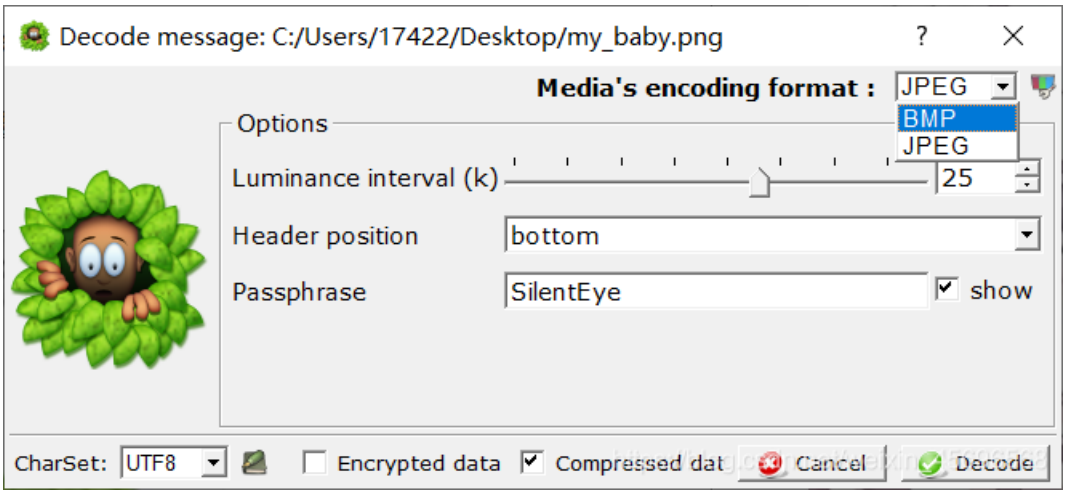
工具隐写

SilentEye

打开图片后点**decode**进行解密



可以注意到有bmp和jpg两种，如果有密码，勾选`encrypteddata`输入密码再解密即可



OurSecret

也是很常用的工具，如果解题需要用到这个工具，出题人一般会给提示，比如`我们的秘密是绿色的`，其中的**秘密**英文是**Secret**，就是暗示用这个工具。

也是windows下的软件，界面如下

使用也很简单，在右侧(**UNHIDE**)打开文件，输入密码，再点击UNHIDE就行

S-Tools

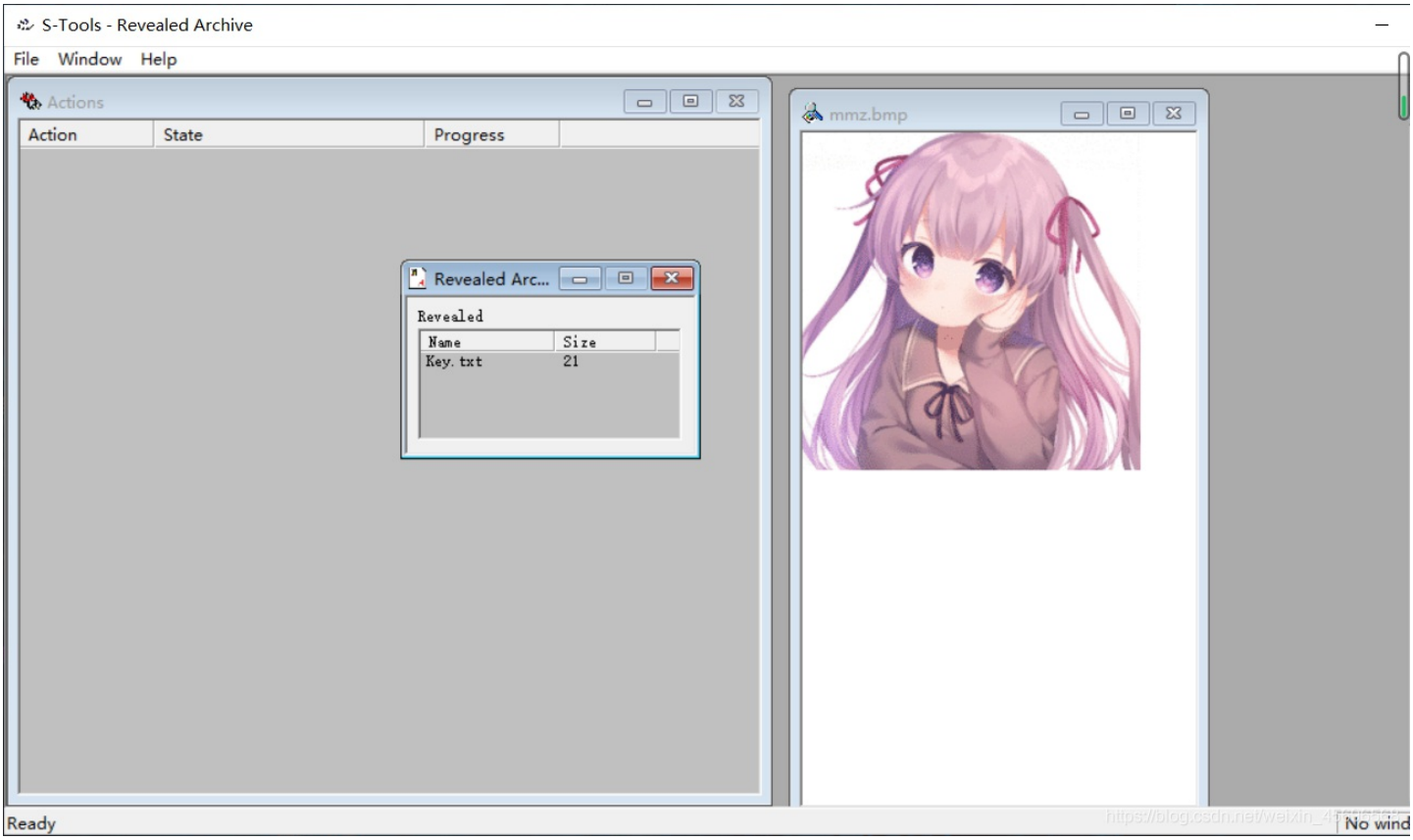
下载地址及详细内容

加密

打开之后，把要加密的图片拖进去，然后把要隐写的文件也拖进去，此时会提示让输入密码，设置密码后得到隐写后的文件

解密

打开之后，把要解密的图片拖进去，然后`右键→Reveal→输入密码`



就可以得到隐藏的文件了

jpg

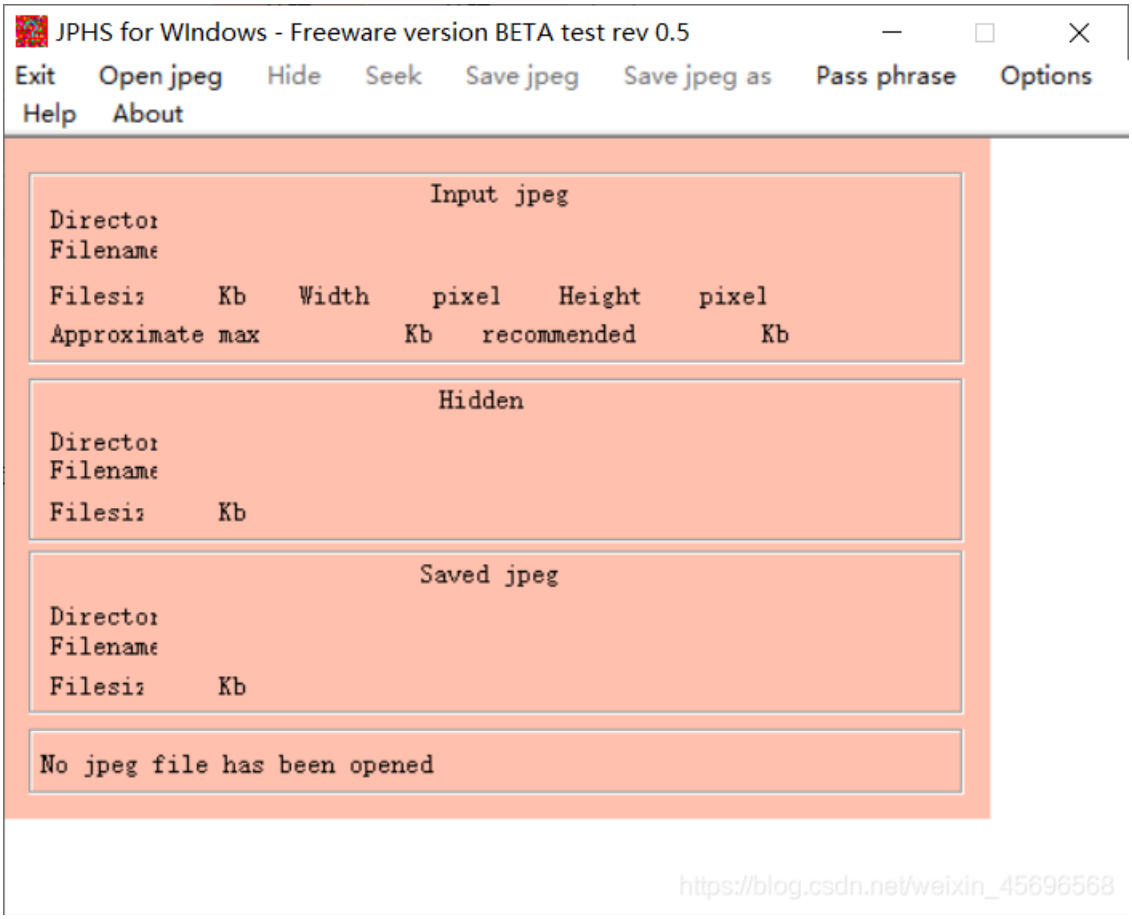
JAVA盲水印

项目地址：https://github.com/ww23/BlindWaterMark/releases

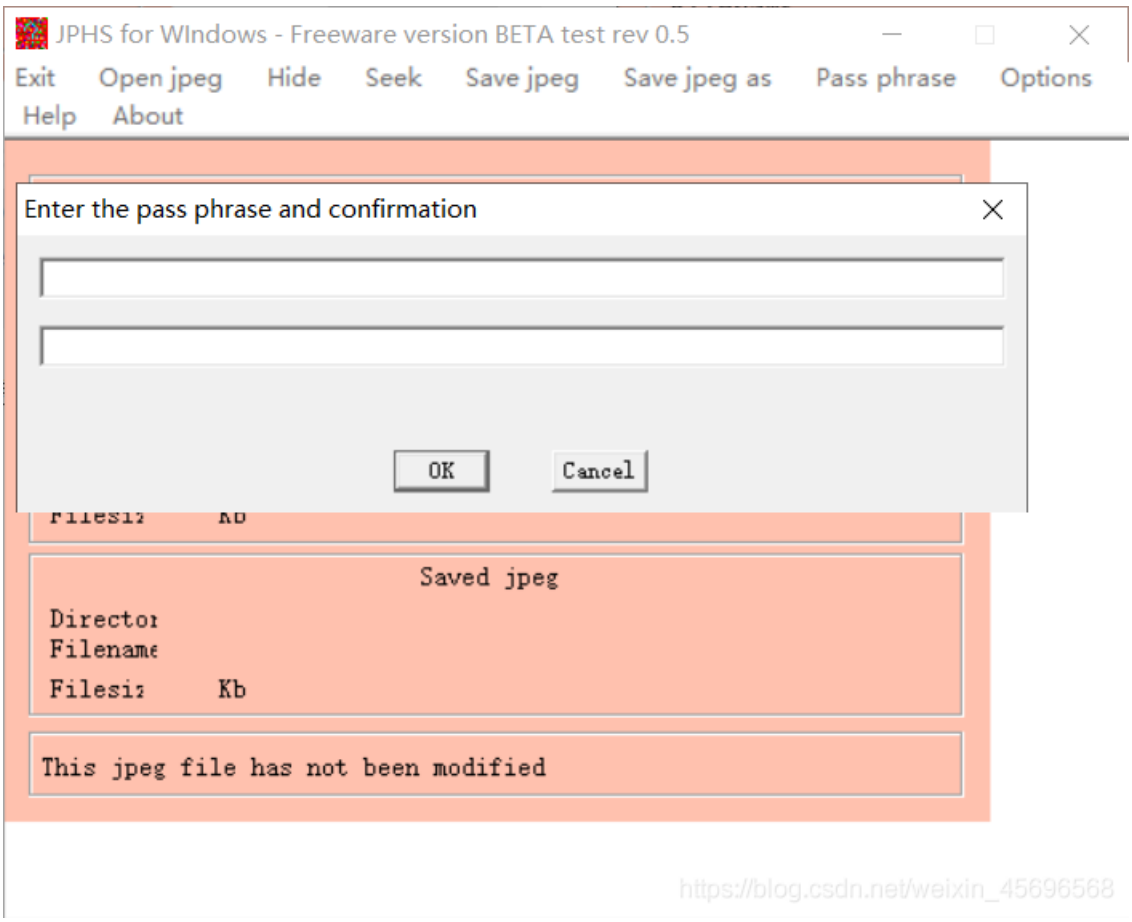
使用命令：`java -jar BlindWatermark.jar decode -c bingbing.jpg decode.jpg`

Jphswin.exe

我用的是windows版本的，界面如下



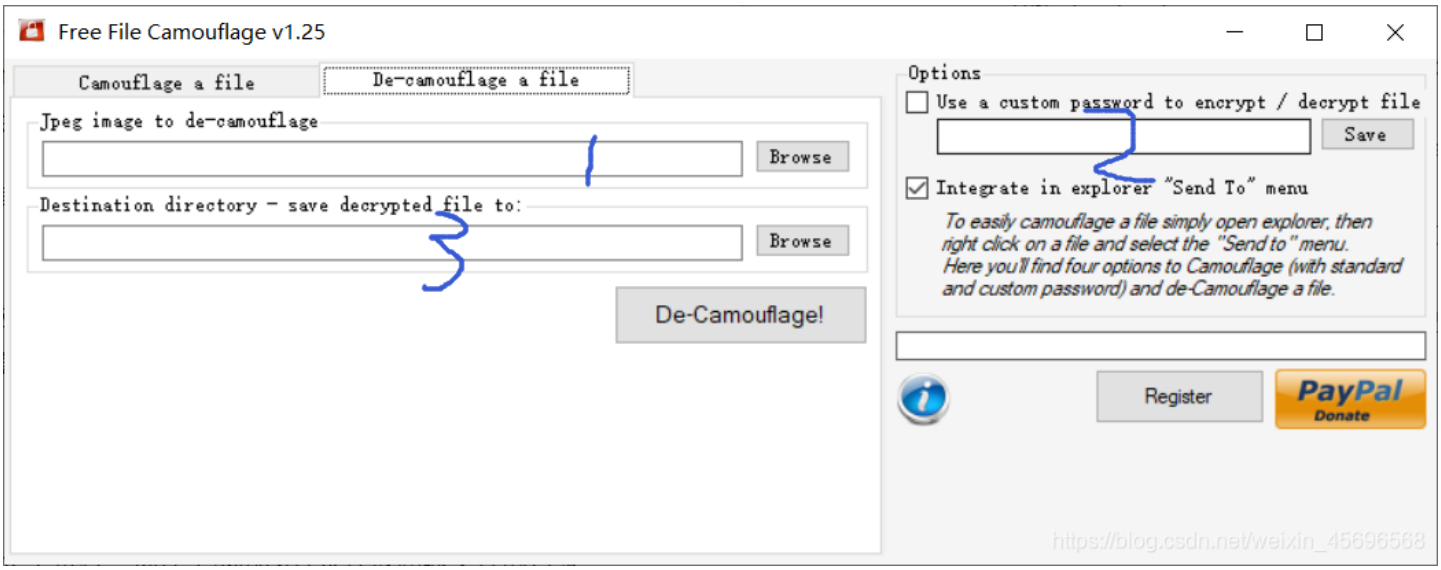
使用起来也很简单，点击`open jpeg`打开图片，然后点击`seek`，如果有密码就输入密码，没有就直接回车



Free_File_Camouflage

不算太常见的一个软件，123分别是

> 解密的图片的位置
> 如果有密码就勾选，然后输入密码，没有就直接跳过
> 解出文件放在什么位置



f5-steganography （F5隐写，需要passwd）

kali中安装，直接`git clone https://github.com/matthewgao/F5-steganography`

具体使用：

```
进入f5-steganography文件夹后，打开终端
# 解密
java Extract filename.jpg -p 密码
运行后会在文件夹里生成一个output.txt，打开即可

# 加密
java Embed 原图.jpg 生成图.jpg -e 隐藏的文件.txt -p 密码
```

outguess （可需要passwd）

kali中安装：

```
先下载
git clone https://github.com/crorvick/outguess
然后进入outguess文件夹，打开终端输入./configure && make && make install
成功之后即可
```

使用：

```
进入文件夹后打开终端

加密
outguess -k 密码 -d hidden.txt 1.jpg 2.jpg
#hidden.txt是要隐写的内容，运行后1.jpg会覆盖2.jpg

解密
outguess -k 密码 -r 2.jpg out.txt
outguess -r 2.jpg out.txt   #不用密码的情况
```

steghide

kali中的安装：`sudo apt-get install steghide`

使用：

```
加密
steghide embed -cf out.jpg -ef flag.txt [-p 密码]
把flag.txt隐写到out.jpg中，如果要添加密码，尾部接上 -p 密码

解密
steghide info out.jpg  #查看图片中嵌入的文件信息
#提取隐藏内容
steghide extract -sf out.jpg -p 密码
steghide extract -sf out.jpg
```

steghide本身不支持爆破密码，可以借助https://github.com/Va5c0/Steghide-Brute-Force-Tool

使用前还要安装一个库：`pip install progressbar2`

使用方法：

```
python steg_brute.py -b -d [字典] -f [jpg_file]
```

png & bmp

普通盲水印

项目地址：https://github.com/chishaxie/BlindWaterMark
具体安装过程自行百度

使用：

```
加密
python2 bwm.py encode 1.png water.png 2.png #PY2
python3 bwmforpy3.py encode 1.png water.png 2.png #PY3

解密
python2 bwm.py decode 1.png 2.png out.png  #PY2
python3 bwmforpy3.py decode 1.png 2.png out.png  #PY3
py2和py3的算法不一样，得到的out.png清晰度也不一样，建议都试一下

也可以加--alpha参数  可能会使图片更清晰
python3 bwmforpy3.py decode 1.png 2.png out.png --alpha 10 #PY3
```

频域盲水印

脚本：

```
# coding=utf-8
import cv2
import numpy as np
import random
import os
from argparse import ArgumentParser
ALPHA = 5
def build_parser():
    parser = ArgumentParser()
    parser.add_argument('--original', dest='ori', required=True)
    parser.add_argument('--image', dest='img', required=True)
    parser.add_argument('--result', dest='res', required=True)
    parser.add_argument('--alpha', dest='alpha', default=ALPHA)
    return parser
def main():
    parser = build_parser()
    options = parser.parse_args()
    ori = options.ori
    img = options.img
    res = options.res
    alpha = options.alpha
    if not os.path.isfile(ori):
        parser.error("original image %s does not exist." % ori)
    if not os.path.isfile(img):
        parser.error("image %s does not exist." % img)
    decode(ori, img, res, alpha)
def decode(ori_path, img_path, res_path, alpha):
    ori = cv2.imread(ori_path)
    img = cv2.imread(img_path)
    ori_f = np.fft.fft2(ori)
    img_f = np.fft.fft2(img)
    height, width = ori.shape[0], ori.shape[1]
    watermark = (ori_f - img_f) / alpha
    watermark = np.real(watermark)
    res = np.zeros(watermark.shape)
    random.seed(height + width)
    x = range(height / 2)
    y = range(width)
    random.shuffle(x)
    random.shuffle(y)
    for i in range(height / 2):
        for j in range(width):
            res[x[i]][y[j]] = watermark[i][j]
    cv2.imwrite(res_path, res, [int(cv2.IMWRITE_JPEG_QUALITY), 100])
if __name__ == '__main__':
    main()
```

使用命令：`python BlindWaterMarkplus.py --original 1.png --image 2.png --result res.png`

如果得到的res.png有问题，把1.png和2.png互换位置试一下

zsteg（lsb隐写）

kali安装：`gem install zsteg`

使用：

```
zsteg 1.png
zsteg 1.bmp
```

PNGDebugger.exe（检查IDAT块）

一般在windows系统下使用，在文件夹中打开cmd命令行

输入：`PNGDebugger.exe filename.png`

tweakpng.exe

windows系统下的一个软件，是很好用的一个PNG图像浏览工具

用来`增加或者删除png图片的IDAT块`很方便，之前做过八神的一个题，就是删除了一个png的图片中的部分IDAT块，保存得到一张不同的图片，其中就有flag，这里因为对png的结构了解的不是很透彻，原理说不清楚。

在使用这个软件打开图片之前，建议先用`PNGDebugger`看一下图片中**有多少IDAT块的crc32是出错**的，因为在使用这个软件的时候，每一个错误都会有一个弹窗。

比如misc入门的`misc44`就不建议用这个软件打开

二维码

二进制转二维码

这里以bugku中的`神奇宝贝`为例

拿到一串二进制，如果它的`长度被开方后正好是整数`，比如这题，满足`625=25*25`，就可以考虑转二维码。

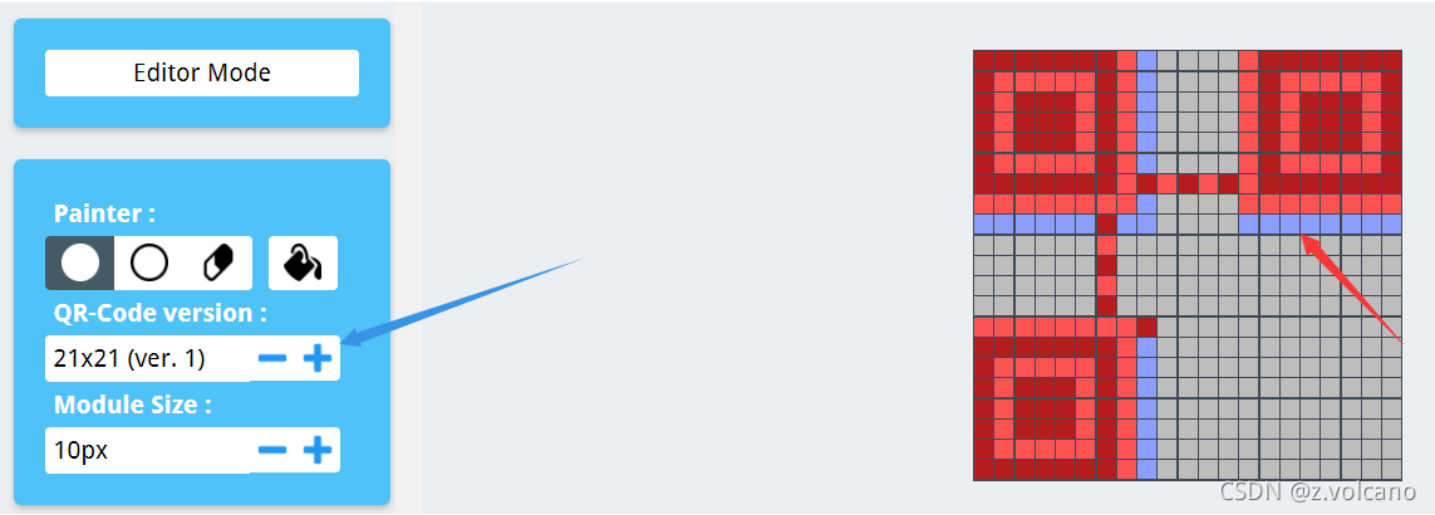
```
import PIL
from PIL import Image
MAX = 25  #图片边长
img = Image.new("RGB",(MAX,MAX))
str="1111111001110111001111111100000100001101010100000110111010011100101010111011011101010111010111000001010111011011101010111011000001001100110101000001111111101010101010111111100000000010010100000000001100011101101011000011000010000000101000010101111000010111101011001111100111001011010011001010100101110001010111001001011010011111100001101010111100110100000100100010111000011110000100111010101100011001011101010001110111111010100000000110001101000110001111111011001100101010101100000010111100111000110001011101001110000111111011101110100111011110110100111011101001110111010001001101001001011000001010001101001111001111111101101110010101000 1"
i = 0
for y in range (0,MAX):
    for x in range (0,MAX):
        if(str[i] == '1'):
            img.putpixel([x,y],(0, 0, 0))
        else:
            img.putpixel([x,y],(255,255,255))
        i = i+1
img.show()
img.save("flag.png")
```

二维码的修复

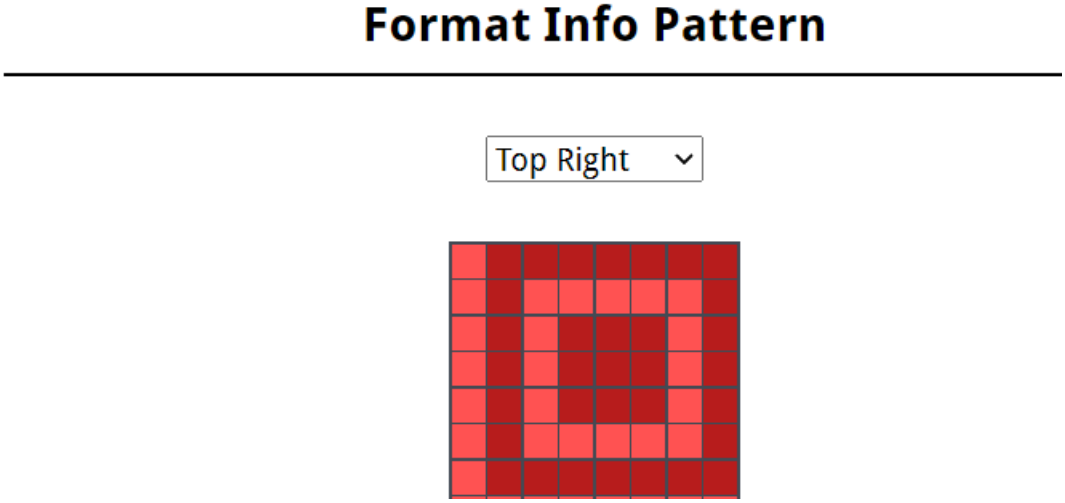
有的时候，得到的二维码是不完整的，如果缺失的部分比较少，是可以直接扫出来的。

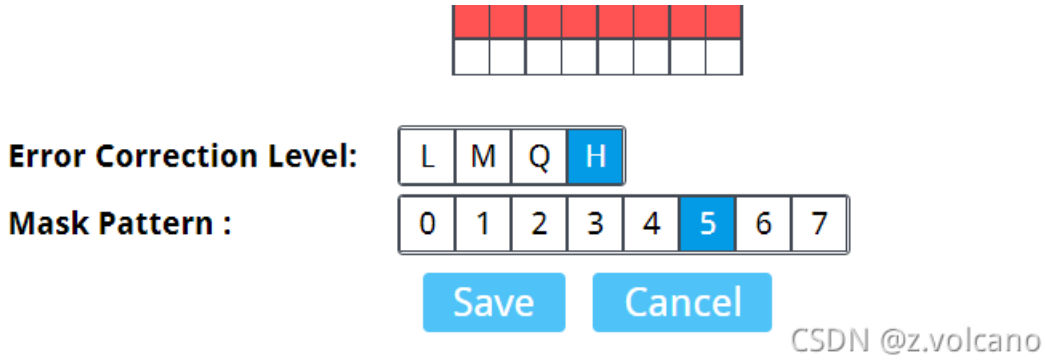
缺失部分比较多的时候就要考虑修复了，这里以bugku的`Improve yourself`为例，其中一步得到的二维码缺了三块定位符。

这里推荐一个在线网站，https://merricx.github.io/qrazybox/，进去的页面如下，蓝色箭头位置是二维码大小，根据实际情况进行修改。



红色箭头位置也可以点击，这里根据实际情况对定位符的一些设置进行修改。





有的时候，题目给了一个图片文件，打开发现是只有上半部分的二维码，一般把图片高度改高就能得到完整的二维码。

Aztec Code

大概长这样

使用在线网站进行解码：https://products.aspose.app/barcode/recognize/aztec#result

---

先写这么多，想起来再补