CTFSHOW-MISC入门

原创

Monica 于 2021-11-27 09:38:01 发布 542 收藏 1

分类专栏： CTFSHOW 文章标签： MISC

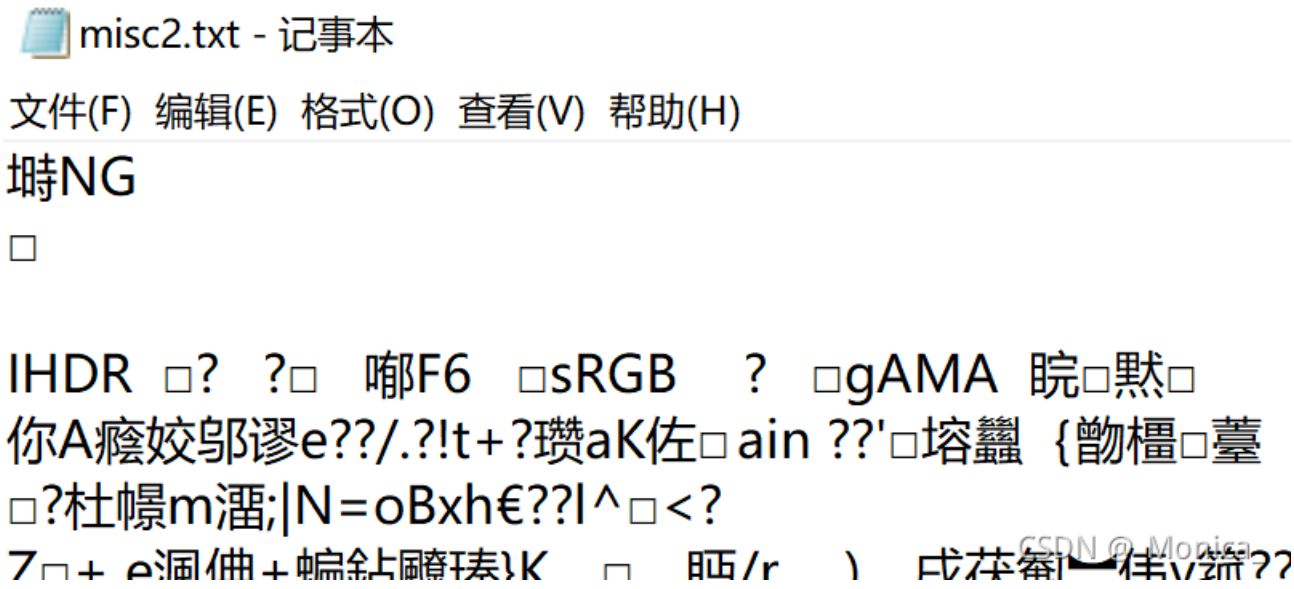
本文链接：https://blog.csdn.net/qq_46918279/article/details/121513585

图片篇(基础操作)

MISC 1

打开即得flag

MISC 2

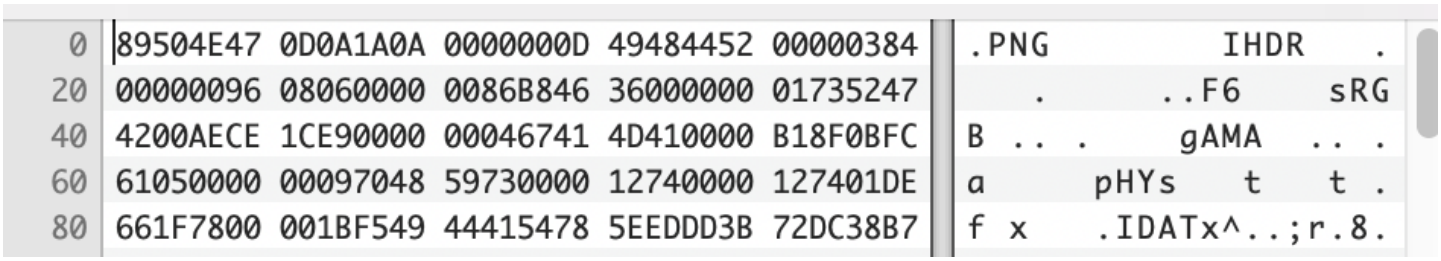


打开是一个

txt文件，使用win的记事本打开，这个塒NG开头就是png文件了。

改后缀为png，打开即得flag。

也可以winhex或者010直接打开，从而知道是png文件
mac上可以用hex fiend



```
0  89504E47 0D0A1A0A 0000000D 49484452 00000384 | .PNG         IHDR    .
20 00000096 08060000 0086B846 36000000 01735247 |    .      ..F6     sRG
40 4200AECE 1CE90000 00046741 4D410000 B18F0BFC | B  ..  .     gAMA   ..  .
60 61050000 00097048 59730000 12740000 127401DE | a      pHYs    t    t  .
80 661F7800 001BF549 44415478 5EEDDD3B 72DC38B7 | f  x    .IDATx^..;r.8.
```

python3脚本

```python
#用于获取图片中的文字
import pytesseract
from PIL import Image

pytesseract.pytesseract.tesseract_cmd = r'D:\应用\Tesseract-OCR\tesseract.exe'
tessdata_dir_config = r'--tessdata-dir "D:\应用\Tesseract-OCR\tessdata"'

image=Image.open(r"D:\this_is_feng\CTF\MISC\ctfshow_misc入门\misc2\misc2.png")
code = pytesseract.image_to_string(image, config=tessdata_dir_config)

print(code)
```

关于pytesseract，可以参考这个

MISC 3

是bpg图片，正常不能打开，需要使用能查看bpg图片的软件打开
bpg下载
win使用方法

```
bpgview.exe 图片名称
```

mac上的不知道咋用，有师傅会的话教教我hhh

MISC 4

给了6个txt文件，hex fiend打开 看文件头和文件尾判断是什么文件

知识点

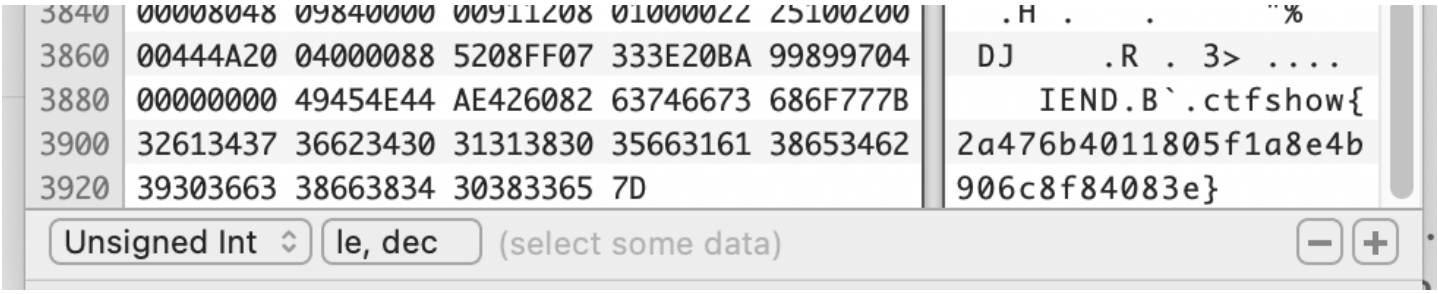
1. JPEG\JPG
   文件头：FF D8 FF
   文件尾：FF D9
   `开头四个点 结尾也是点`
2. TGA
   未压缩的前4字节 00 00 02 00
   RLE压缩的前5字节 00 00 10 00 00
3. PNG
   文件头：89 50 4E 47 0D 0A 1A 0A
   文件尾：AE 42 60 82
   `即开头为.PNG,结尾为IEND.B`
4. GIF
   文件头：47 49 46 38 39(37) 61
   文件尾：00 3B
   `开头为GIF89a`
5. BMP
   文件头：42 4D
   文件头标识(2 bytes) 42(B) 4D(M)
   `BM开头`
6. TIFF (tif)
   文件头：49 49 2A 00
   `II开头`
7. ico
   文件头：00 00 01 00
   Adobe Photoshop (psd)
   文件头：38 42 50 53

所以第一个为png、第二个是jpg、第三个是bmp、第四个是gif、第五个是tif、第六个是webp文件。`应该为RIFFL开头`
改后缀即可

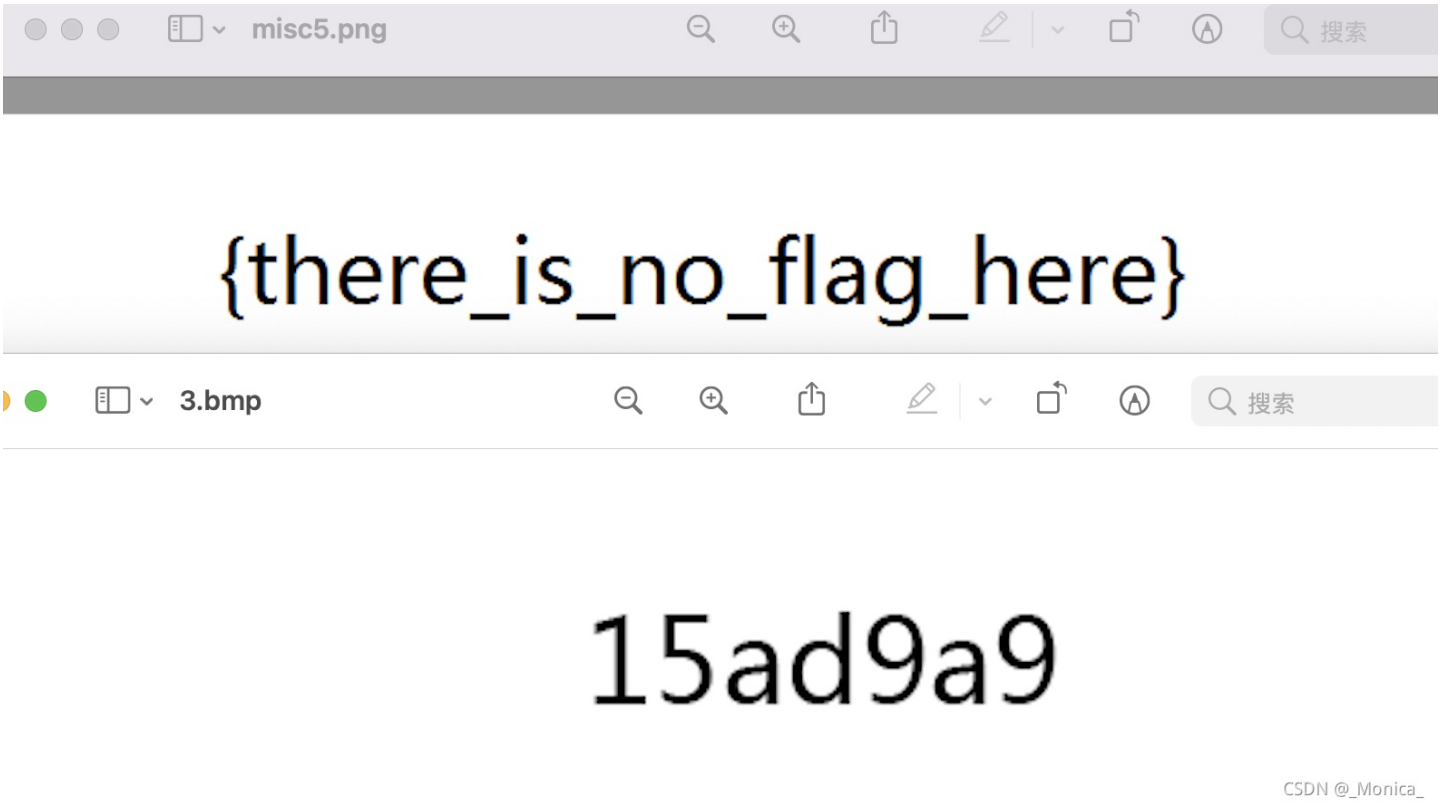
图片篇(信息附加)

MISC 5

用hex fiend或者win hex或者010打开，

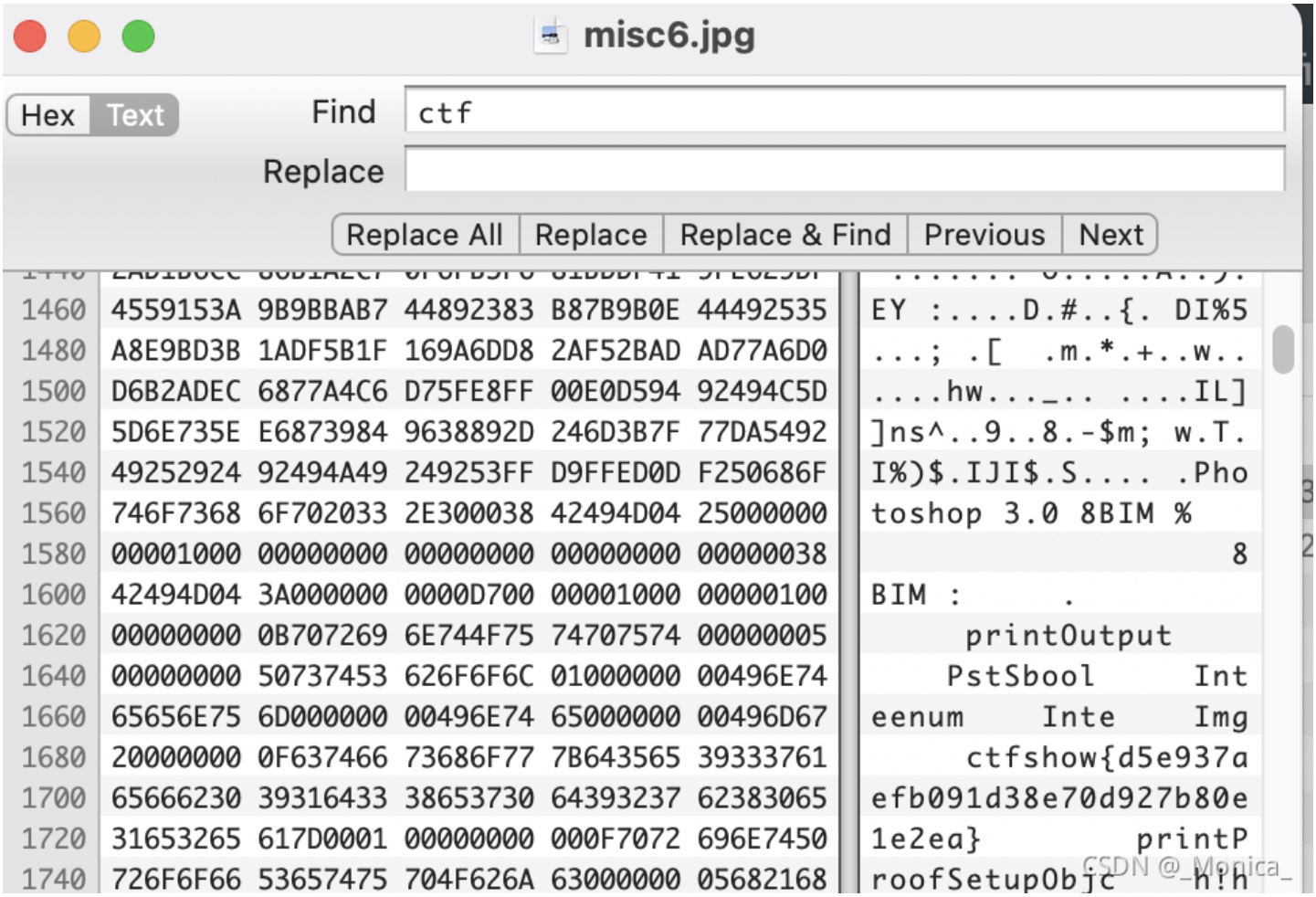


总是感觉经过修改过后的图片显示的内容有点奇怪：



MISC 6

同上，搜索即可



MISC 7

提示：flag在图片文件信息中。
同上hexfiend打开搜索ctf即可

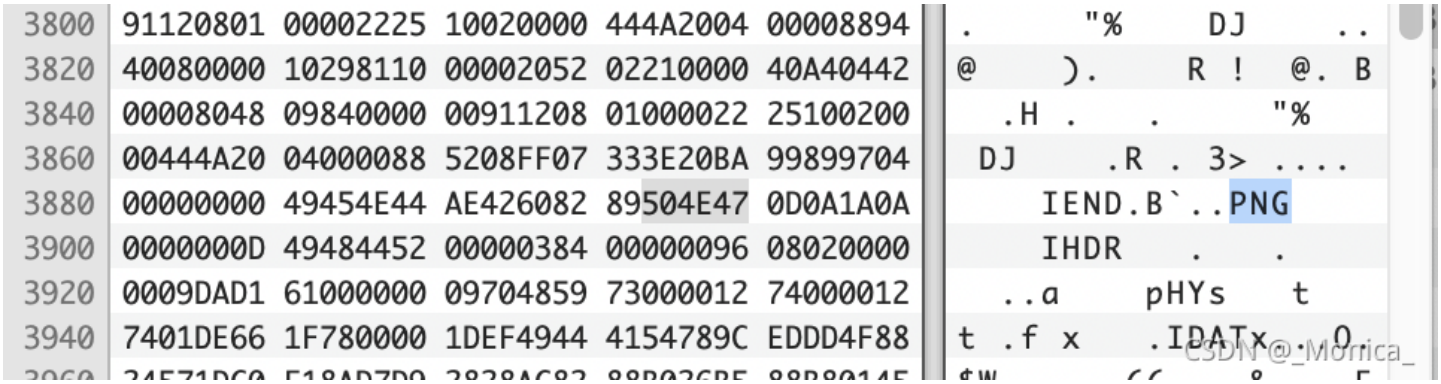
或者根据提示
右键查看属性、右键查看属性是常用的方法，不过获取不到图片的全部文件信息，也得不到这题的flag
听说可以使用在线网站查看详细的exif信息，但是还是没找到、显示不全

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

或者用notepad打开也能做

MISC 8

提示：flag在图片文件中图片文件中。
用hex fiend打开，可以发现图片的内容有PNG（即隐写了其他图片）手动或者binwalk或者foremost分离出图片。



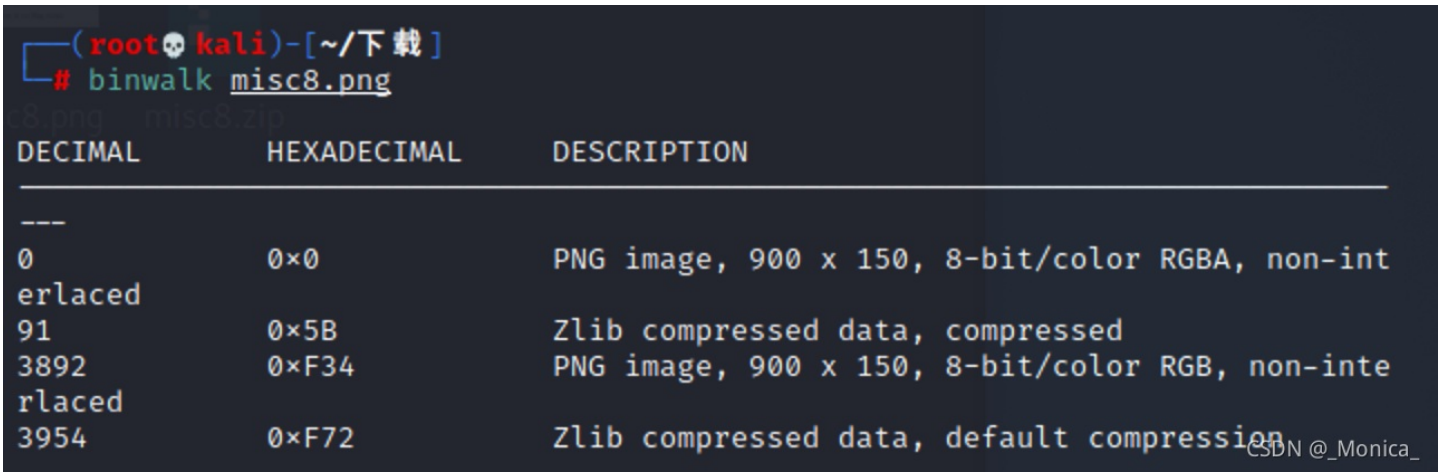
binwalk

> Binwalk是用于搜索给定二进制镜像文件以获取嵌入的文件和代码的工具。 具体来说,它被设计用于识别嵌入固件镜像内的文件和代码。 Binwalk使用libmagic库,因此它与Unix文件实用程序创建的魔数签名兼容。 Binwalk还包括一个自定义魔数签名文件,其中包含常见的诸如压缩/存档文件,固件头,Linux内核,引导加载程序,文件系统等的固件映像中常见文件的改进魔数签名。

使用方法参考文章

使用binwalk

```
输入命令 binwalk misc8.png
```



可以看见第三行还有一个png图片，起始地址是3892

```
输入命令：dd if=misc8.ong of=1.png skip=3892 bs=1
```

然后会在当前目录下生成一个1.png文件，打开即可
其中if=misc8.jpg是输入文件，of=1.png是输出文件，skip是指定从输入文件开头跳过3892个块后再开始复制，bs设置每次读写块的大小为1字节 。

使用命令foremost

> foremost 和 binwalk 类似，主要是用于CTF_杂项的隐写题，分离提取隐写的文件。

输入命令

```
输入命令：foremost misc8.png
```

可以直接将图片文件中包含的所有文件分离，输出到一个output文件夹中。打开这个文件夹，我们可以看到audit.txt为记录分离过程的文件，png为该图片文件中包含的所有.png文件，zip为该图片文件中包含的所有.zip文件。打开png文件夹即可找到flag

MISC 9

提示：flag在图片块里
用hex fiend打开一样找得到flag

MISC 10

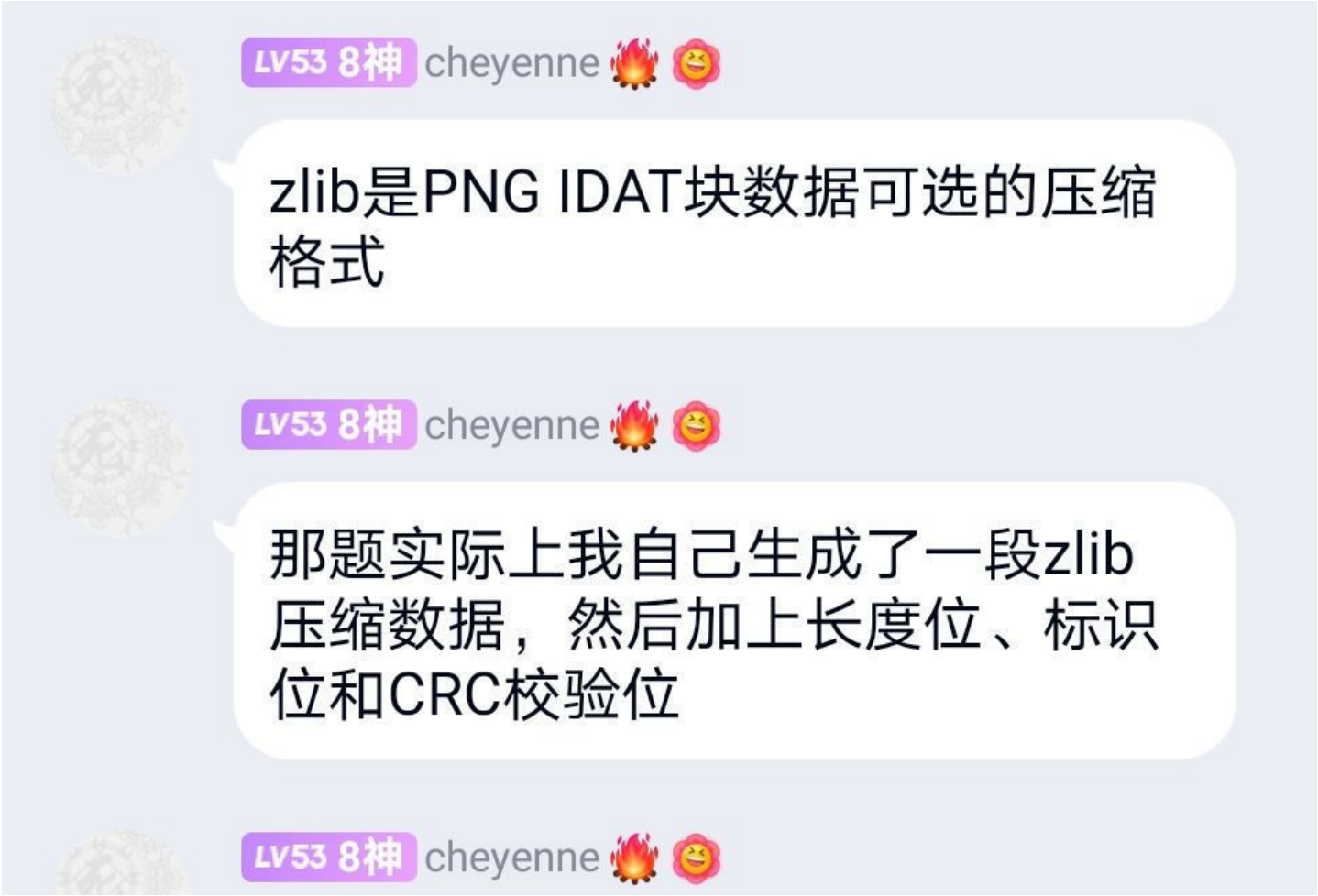
提示：flag在图片数据里。
binwalk -e 分离出数据后；打开第一个文件即得flag。

知识点

PNG定义了两种类型的数据块：一种是PNG文件必须包含、读写软件也都必须要支持的关键块（critical chunk）；另一种叫做辅助块（ancillary chunks），PNG允许软件忽略它不认识的附加块。这种基于数据块的设计，允许PNG格式在扩展时仍能保持与旧版本兼容。

关键数据块中有4个标准数据块：

- 文件头数据块IHDR（header chunk）：包含有图像基本信息，作为第一个数据块出现并只出现一次。
- 调色板数据块PLTE（palette chunk）：必须放在图像数据块之前。
- 图像数据块IDAT（image data chunk）：存储实际图像数据。PNG数据允许包含多个连续的图像数据块。
- 图像结束数据IEND（image trailer chunk）：放在文件尾部，表示PNG数据流结束。



伪装成了一个IDAT块，然后插入了图片文件里

LV53 8神 cheyenne

沐秋的清晨 下午6:24

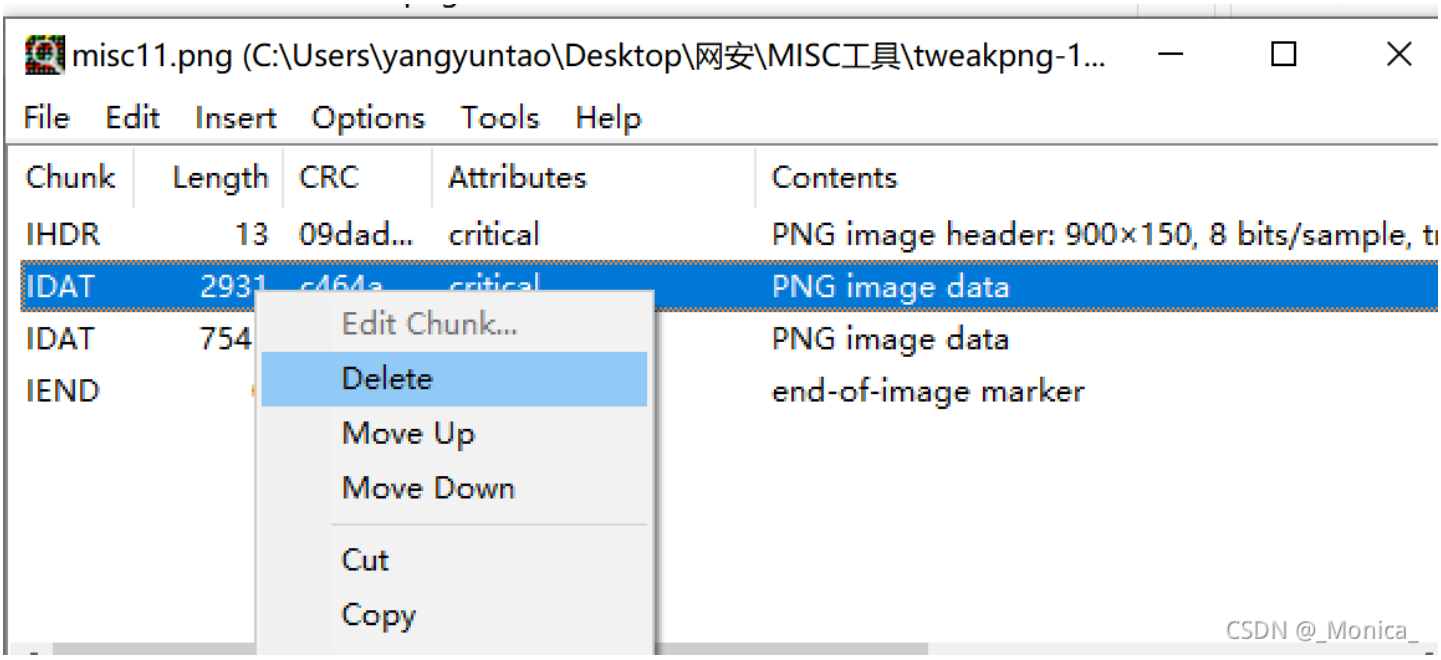
阿这

来自其他聊天

@沐秋的清晨 binwalk可以一把梭，是因为binwalk会找到zlib块的标记然后提取出来，同时因为这是个压缩数据，binwalk的-e参数会自动把提取到的压缩包尝试进行解压，所以最后的提取结果里就有原始的那段文本，就是flag了

MISC 11

提示：flag在另一张图里。
根据上一题的知识点，加上提示，大概猜测为多个IDAT数据，导致显示不出来flag，删除其他IDAT数据，只保留flag的即可得到flag。
把第一个IDAT块的数据删除，然后另存为一张新图片
使用工具：Tweakpng
右击第一个IDAT块，然后delete就行了。



MISC 12

提示：flag在另一张图里。

和上题一样，不过这题有30个IDAT块，用PNGDebugger跑了一下，但是发现没有出错的IDAT块...测试后发现需要删掉前8个IDAT块

知识点

PNG Debugger 可以进行读取 PNG 图片的数据，检测各数据块中的 CRC 是否正确，在 Windows 下使用。
CRC 码是 循环冗余校验码 的简称，是 png 图片中一种的数据， 是目前使用非常广泛的数据校验方式，不仅能校验传递过来的数据正确性,还能筛查出哪一位出现了错误。
比如可用于判断 png 图片的宽和高是否正确。

使用方法：
进入pngdebugger 目录下使用命令

```
pngdebugger misc12.png
```

```
C:\Users\yangyuntao\Desktop\网安\MISC工具\png-debugger-master\Debug>pngdebugger misc12.png
        ----
file-path=misc12.png
file-size=11206 bytes

0x00000000      png-signature=0x89504E470D0A1A0A

0x00000008      chunk-length=0x0000000D (13)
0x0000000C      chunk-type='IHDR'
0x0000001D      crc-code=0x09DAD161
>> (CRC CHECK)  crc-computed=0x09DAD161          =>      CRC OK!


0x00000021      chunk-length=0x000001EE (494)
0x00000025      chunk-type='IDAT'
0x00000217      crc-code=0xDAFE0102
>> (CRC CHECK)  crc-computed=0xDAFE0102          =>      CRC OK!


0x0000021B      chunk-length=0x000001B3 (435)
0x0000021F      chunk-type='IDAT'
0x000003D6      crc-code=0x4ACC299B
```


MISC 13

提示：flag位置在图片末尾。

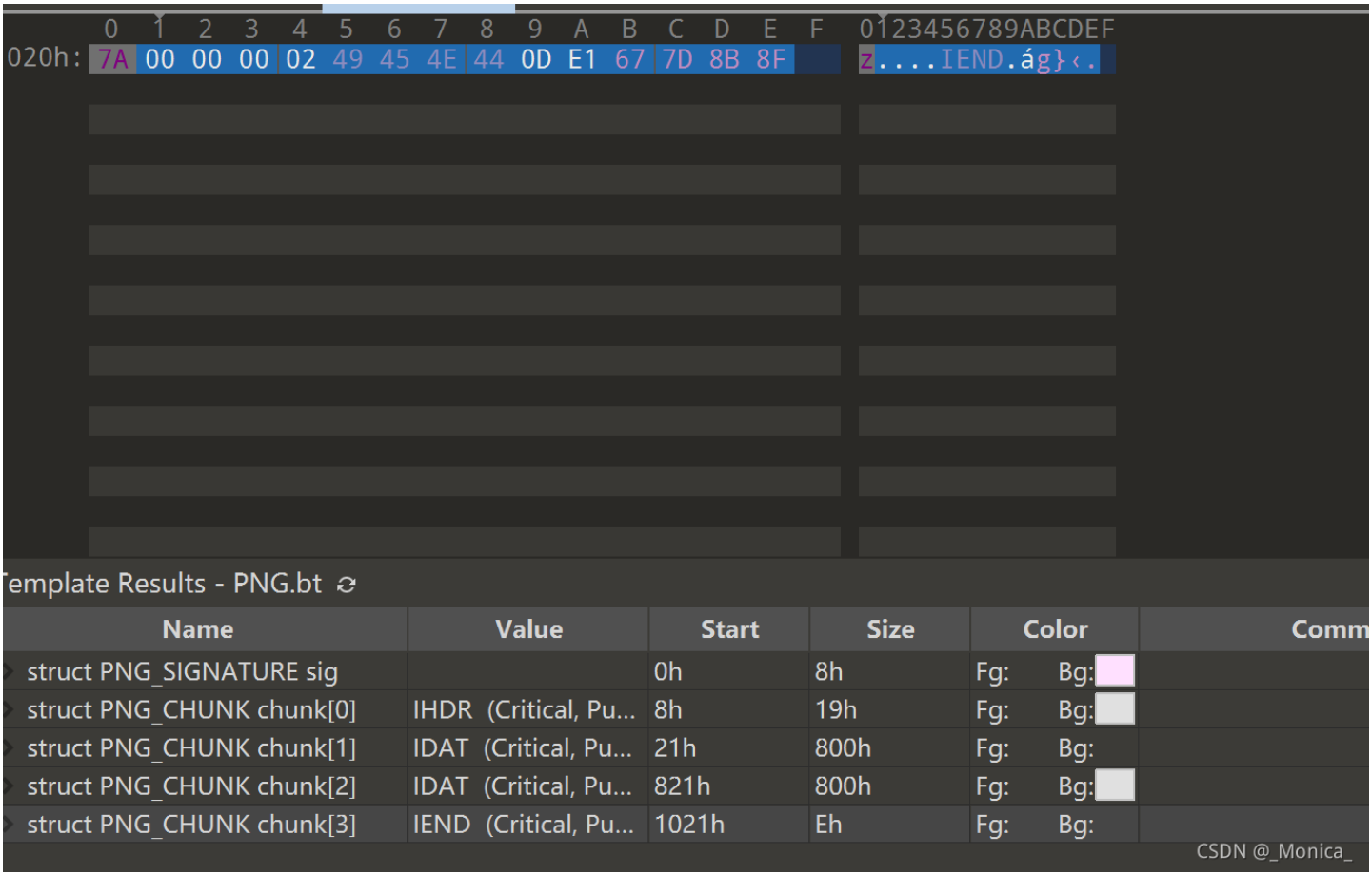
hex fiend打开可以看见四处ctfshow flag字样

```
3520  D545CEF5 E0DC883D 645C1725 3B7C9609 82F9B157  .E.....=d\ %;|. ...W
3540  49EF3340 09C80BC6 2BE4023A D4631A74 B9668573  I.3@ . .+. :.c t.f.s
3560  8668AA6F 4B77B07B 21611465 5336A565 54333465  .h.oKw.{!a eS6.eT34e
3580  78612534 DD38EF66 AB351031 95381F62 8237BA65  xa%4.8.f.5 1.8 b.7.e
3600  45347C32 54647E37 3A64E465 F136FA66 F5341E31  E4|2Td~7:d.e.6.f.4 1
3620  07321D66 5438F133 3239E961 6C7D9428 62E7A1CA   2 fT8.329.al}.(b...
3640  A7248E7E B82AAC1F A193E3FF 9F1300AF 30882A73  .$.~.*. ..... .0.*s
3660  79F69F49 20D18584 9313F735 D1852555 17069EEA  y..I .... .5..%U  ..
3680  B9599CC7 153F79B2 A64DC317 AA7C1231 2503FEFE  .Y.. ?y..M. .| 1% ..
3700  ABC8637C BECE1CDB 4ED47D35 D643BDB3 FF7C5C1A  ..c|.. .N.}5.C...|\
3720  781B7F02 6C795332 7A7CC43E 972E74B2 471754C1  x   lyS2z|.>..t.G T.
3740  A6E56FED 38C5C80F 49899339 04D5A7DF 2714589C  ..o.8.. I..9 ...' X.
3760  964C1F5B DF9C9292 39ABA43B D3CA3109 C059EAF3  .L [....9..;..1 .Y..
3780  0F5A23DC DC34C8DE 3A9C35A0 A7ABD556 45BC5D3F   Z#..4..:.5....VE.]?
3800  5450D240 DDB6147D FCDCFE33 D27235C0 72BB9792  TP.@.. }...3.r5.r...
3820  BE5C8923 88B8538D 17F3F963 1A74B966 85738668  .\.#..S. ..c t.f.s.h
3840  AA6F4B77 B07B2161 14655336 A5655434 34367863  .oKw.{!a eS6.eT446xc
3860  2534DD38 EF66AB37 10339539 1F628237 BA654562  %4.8.f.7 3.9 b.7.eEb
3880  7C325464 7E313A64 E465F136 FA65F534 1E310732  |2Td~1:d.e.6.e.4 1 2
3900  1D665438 F1333239 E9616C7D 2BF5E0D5 3E44E6CD   fT8.329.al}+...>D..
3920  C8C8F3A5 2F793396 FE4176F9 6E49E4BA BD00D892  ..../y3..Av.nI.....
```

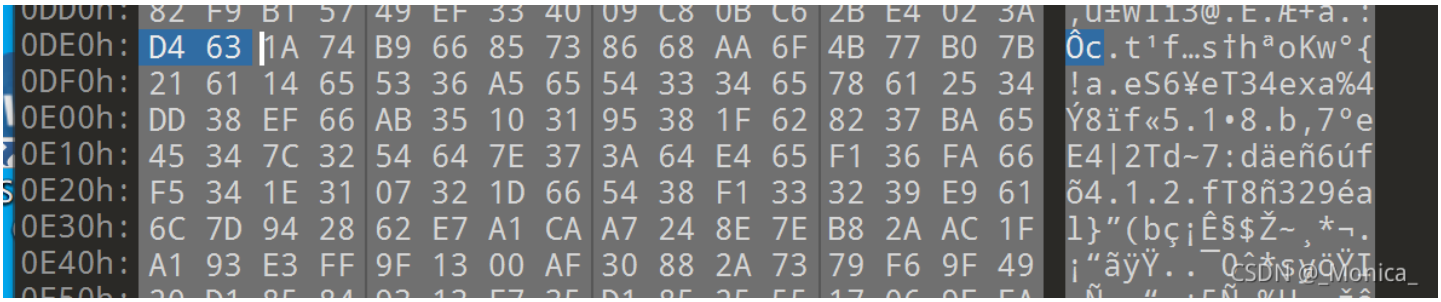
只有一处是对的
是这样解释的：

> 13是这样的，文件里一共塞了四个flag，需要判断哪一个是正确的。根据题目提示，“flag位置”在文件末尾，找到文件末尾的IEND块，会发现这个块是14字节而非通常的12字节，也就是说这个块除了4位长度、4位标识和4位CRC，还带了两位数据（通常IEND块是不带数据的，但它也可以带，这应该就是本题知识点）。这两位数据就是正确flag开头那个c字符的字节序号。换句话说其实先找到IEND块里这个序号，然后去找对应的字节，一眼就能看到flag了。强行硬找的话一般会找到第一个，那个是假的；但是不是很理解

png数据块结构
可以看到0DE1是隐藏的数据，



找到0DE1这一处即可



注意到{前面那一串字符，从第一位开始，每隔一位选取一个字符，连起来就是ctfshow
这里把这串十六进制数值复制下来，按照规律选取正确的数值

```
a="631A74B96685738668AA6F4B77B07B216114655336A56554333465786125340D38EF66AB35103195381F628237BA6545347C3254647E373A64E465F136FA66F5341E3107321D665438F1333239E9616C7D"

flag=""
for i in range(0,len(a),4):
    hexStr=a[i:i+2]
    flag+=chr(int("0x"+hexStr,16))
print(flag)
```

MISC 14

提示：flag在那张图里。
猜测应该是图片中还有图片

使用命令

```
binwalk -e misc14.jpg
```

结果：

```
┌──(root💀kali)-[~/下载]
└─# binwalk -e misc14.jpg                                                    3 ×

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, EXIF standard
12            0xC             TIFF image data, big-endian, offset of first im
age directory: 8
1681          0x691           TIFF image data, big-endian, offset of first im
age directory: 8
2103          0x837           JPEG image data, JFIF standard 1.01    CSDN @_Monica_
```

发现有4段、第四段还是jpeg文件

知识点：

> 标签图像文件格式（Tag Image File Format，TIFF）是一种灵活的位图格式，主要用来存储包括照片和艺术图在内的图像，最初由Aldus公司与微软公司一起为PostScript打印开发。TIFF与JPEG和PNG一起成为流行的高位彩色图像格式。TIFF格式在业界得到了广泛的支持，如Adobe公司的Photoshop、The GIMP Team的GIMP、Ulead PhotoImpact和Paint Shop Pro等图像处理应用、QuarkXPress和Adobe InDesign这样的桌面印刷和页面排版应用，扫描、传真、文字处理、光学字符识别和其它一些应用等都支持这种格式。从Aldus获得了PageMaker印刷应用程序的Adobe公司控制着TIFF规范。

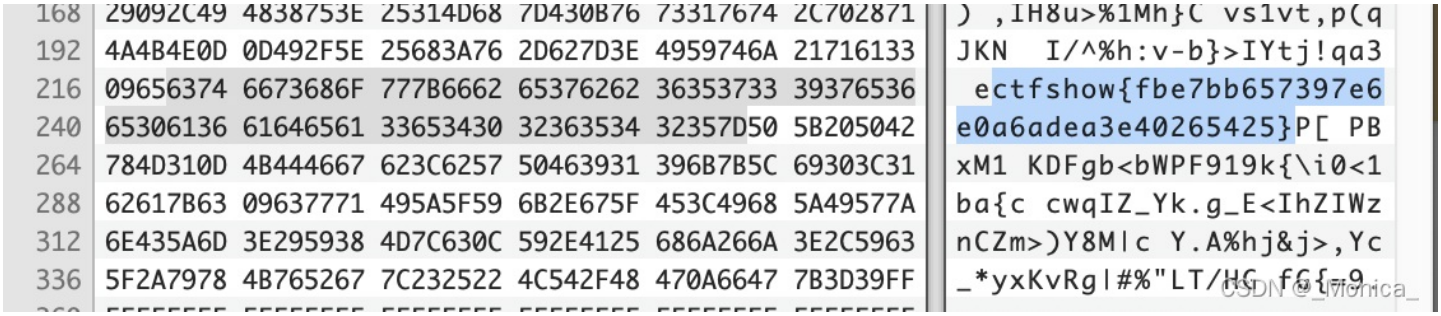
从binwalk分析出来的数据位置截取数据。可以看到其实地址为2103
使用命令

```
dd if=misc14.jpg of=flag.jpg skip=2103 bs=1
```

打开flag.jpg即可。关于dd命令可以看misc8

MISC15

hex fiend打开即可

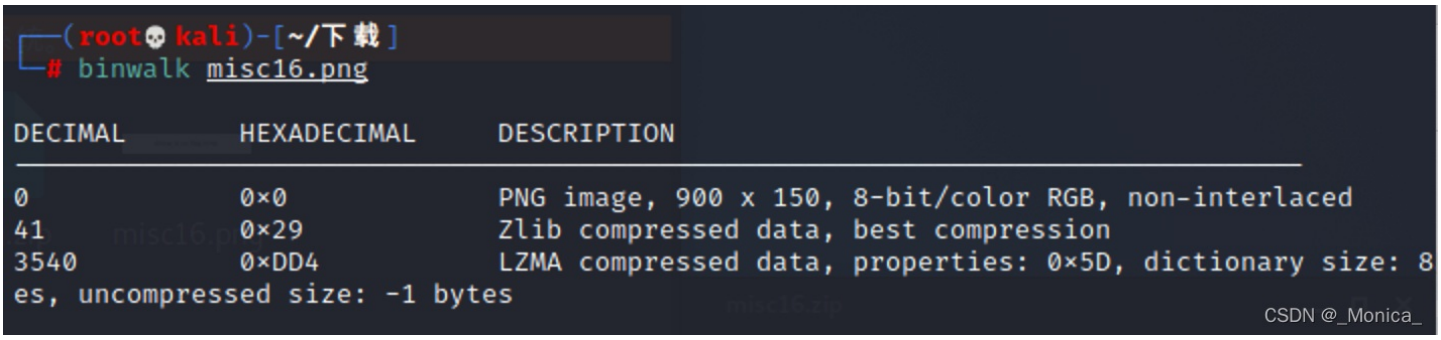


MISC16

提示：flag在图片数据里。
使用命令

```
binwalk misc16.png
```

结果：



可以看见有一个lzma算法的压缩文件，（这种文件的后缀名为.lzma，可以用7-zip打开）

> LZMA，（Lempel-Ziv-Markov chain-Algorithm的缩写），是一个Deflate和LZ77算法改良和优化后的压缩算法，开发者是Igor Pavlov，2001年被首次应用于7-Zip压缩工具中，是 2001年以来得到发展的一个数据压缩算法。它使用类似于 LZ77 的字典编码机制，在一般的情况下压缩率比 bzip2 为高，用于压缩的可变字典最大可达4GB。

方法：使用命令

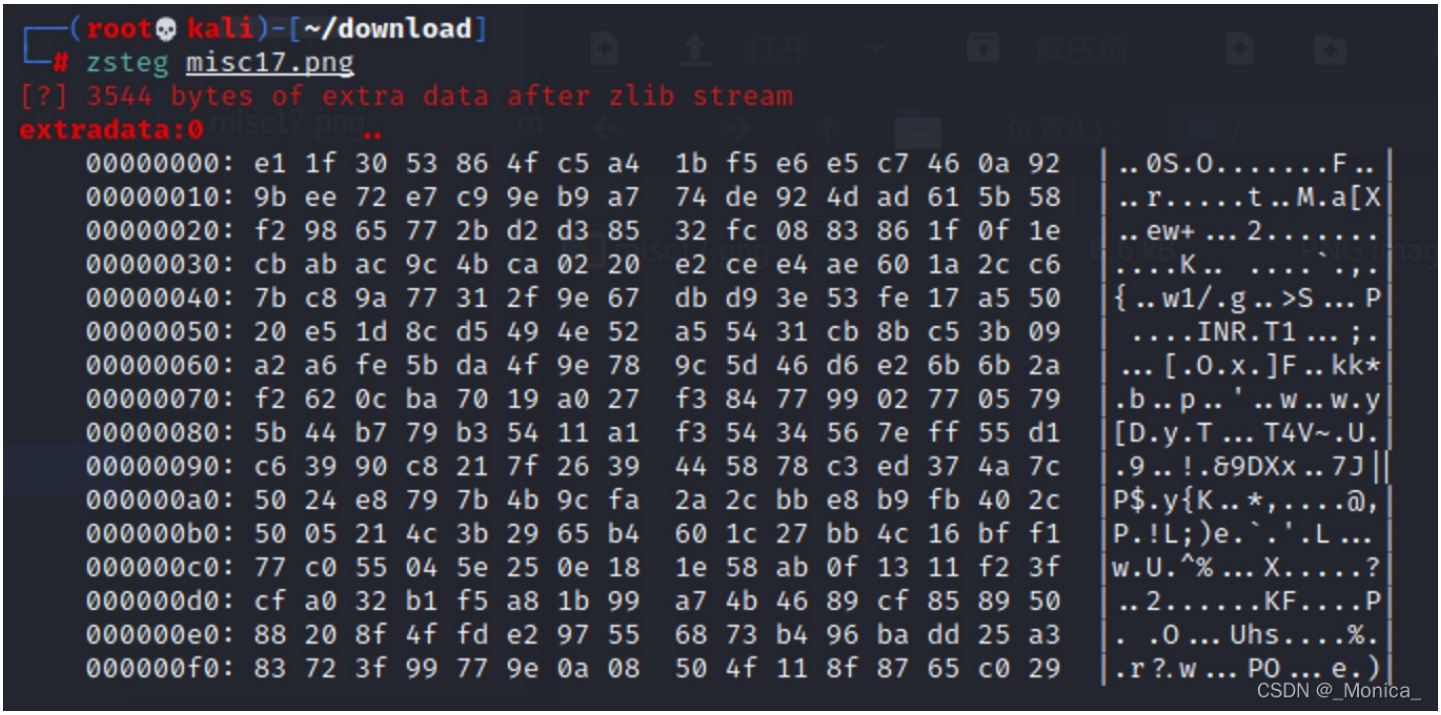
```
binwalk -e misc16.png
```

打开结果里面的DD4文件即可

MISC17

先binwalk分析，没啥问题，然后试试zsteg

> zsteg工具：用于检测被隐写在png，bmp图片里的数据。



发现隐藏的数据，位置处于extradata:0；有3544 bytes的隐写内容
将数据提取出来：

```
zsteg -E  "extradata:0" misc17.png > 1.txt
```

然后再binwalk -e把1.txt中的数据分离出来，拿到flag

```
binwalk -e 1.txt
```

MISC 18

提示：flag在标题、作者、照相机和镜头型号里。
使用在线网站查看详细的exif信息

IFD0

型号	28ac17e5f0
创作者	5d60c208f7
XP标题	ctfshow{32
XP作者	5d60c208f7
Padding	(Binary data 2072 bytes, use -b option to extract)

ExifIFD

Padding	(Binary data 2060 bytes, use -b option to extract)

XMP-rdf

About	uuid:faf5bdd5-ba3d-11da-ad31-d33d75182f1b

XMP-dc

标题	ctfshow{32
描述	ctfshow{32
Creator	5d60c208f7

XMP-microsoft

镜头型号	2d4cf5a839}

MISC 19

提示：flag在主机上的文档名里。
方法同上

MISC 20

提示：flag在评论里。
方法同上

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

ctfshow{c97964b1aecf06e1d79c21ddad593e42}

MISC 21

提示：flag在序号里。
方法同上

序列号	68657285826597329

16进制解码
得到：hex(X&Ys)

> hex() 函数用于将10进制整数转换成16进制，以字符串形式表示。

X分辨率	3902939465
Y分辨率	2371618619
PageName	https://ctf.show/
X定位	1082452817
Y定位	2980145261
目标Printer	ctfshow{} CSDN @_Monica_

将xy四段，每段分别转换为16进制，组合起来加上ctfshow{}即可
ctfshow{e8a221498d5c073b4084eb51b1a1686d}

MISC 22

提示：flag在图片里

知识点：

> ThumbnailImage 缩略图
> JPEG图片采用了有损压缩的方式，其过程比较复杂。过去的JPG图片是不含内嵌缩略图的，但现在为了能让大家快速查看其缩略图，研发PEG格式的专家组就制定了多一项标准在图片文件中记录了一些EXF信息。数码照相机拍出的图片带有相机的很多参数，这都属于EXIF信息，其中缩略图也是一部分。缩略图其实是一幅较小的JPEG图片，存储在EXIF信息段，因此这个缩略图是内嵌在图片文件里面的。支持EXF信息内嵌缩略图数据库的片格式的图片除了JPEG格式（.jpg、jpeg、jpe），还有PSD、PDD、EPS、TIF、TIFF等格式。

使用工具exiftool
使用命令：

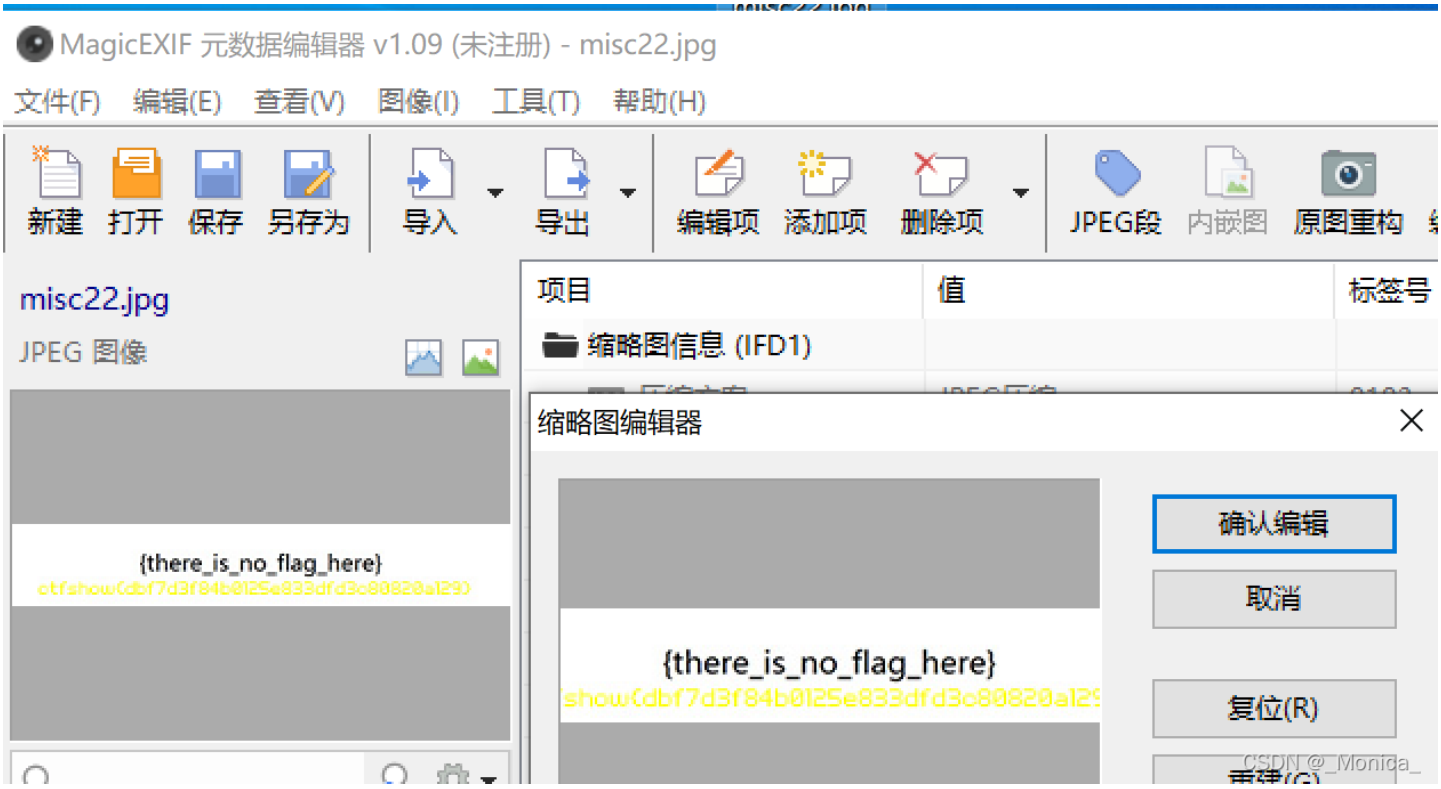
```
exiftool -ThumbnailImage -b misc22.jpg > 1.jpg
```

工具命令：

> 1：-b (-binary) 以二进制输出元数据
> 2：生成image.raw的缩略图thumbnail.jpg
> exiftool -b -ThumbnailImage image.raw > thumbnail.jpg
> 参考文章

打开1.jpg即可

或者使用magicexif直接打开



MISC 23

提示：flag在时间里。
时间应该是属于图片的属性，所苦可以用exiftool来查看

```
exiftool misc23.psd
```

```
History Action                  : ctfshow{}, UnixTimestamp, DECtoHEX, getflag
```

可以看到flag的形式， UnixTimestamp, DECtoHEX, getflag
Timestamp指的是时间戳，DECtoHEX是十进制转十六进制
（decimalism十进制，hexadecimal十六进制）
根据提示
History When : 1997:09:22 02:17:02+08:00, 2055:07:15 12:14:48+08:00, 2038:05:05 16:50:45+08:00, 1984:08:03 18:41:46+08:00

+08:00是转换为北京时间的意思

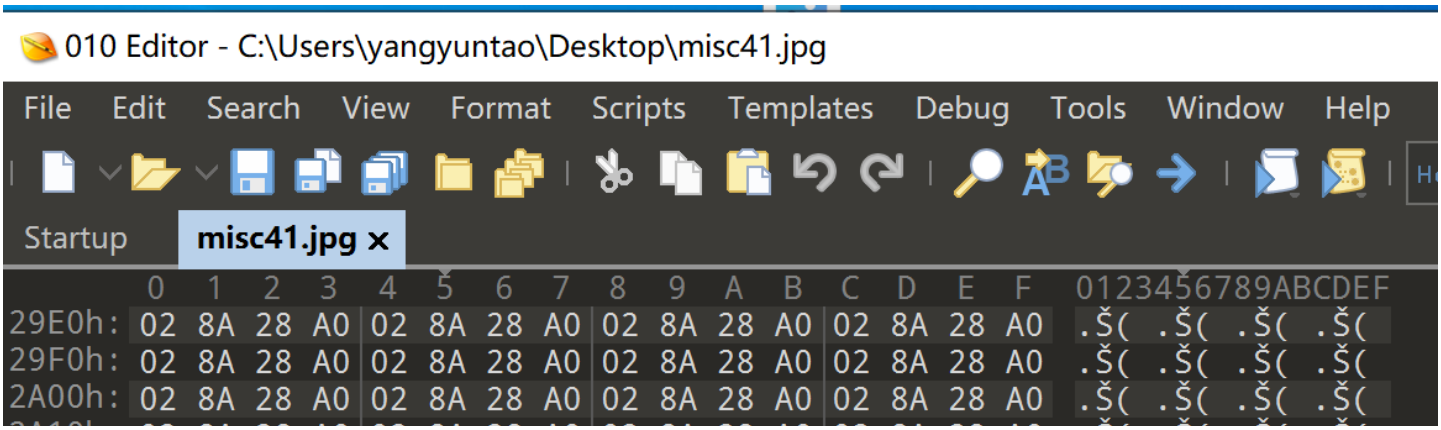
> UTC是国际时，UTC+8就是国际时加八小时，是东八区时间，是北京时间。
> GMT就是指格林尼治所在地的标准时间，＋8：00就是东八区的时间，即北京时间。

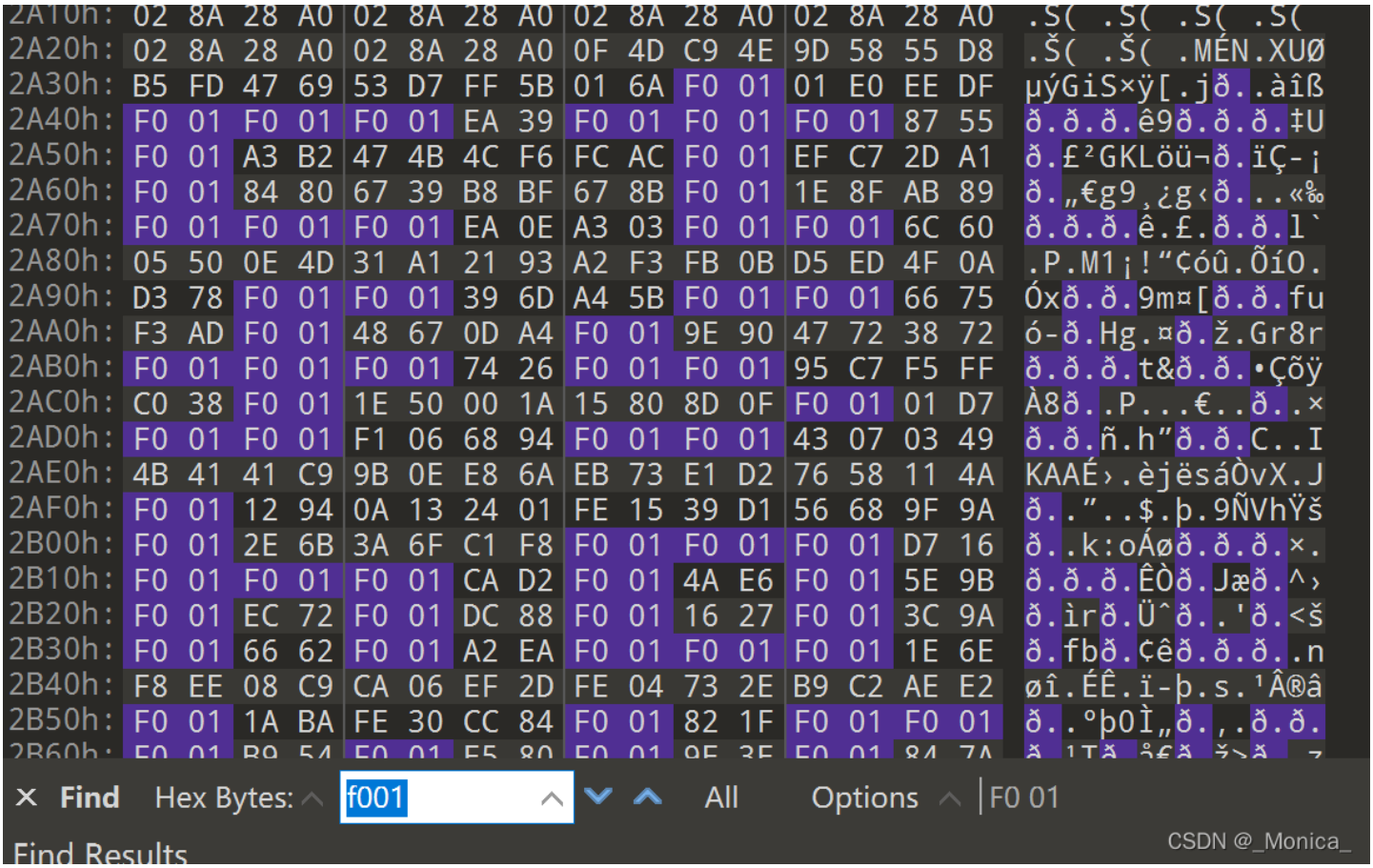
在线时间戳转换网站

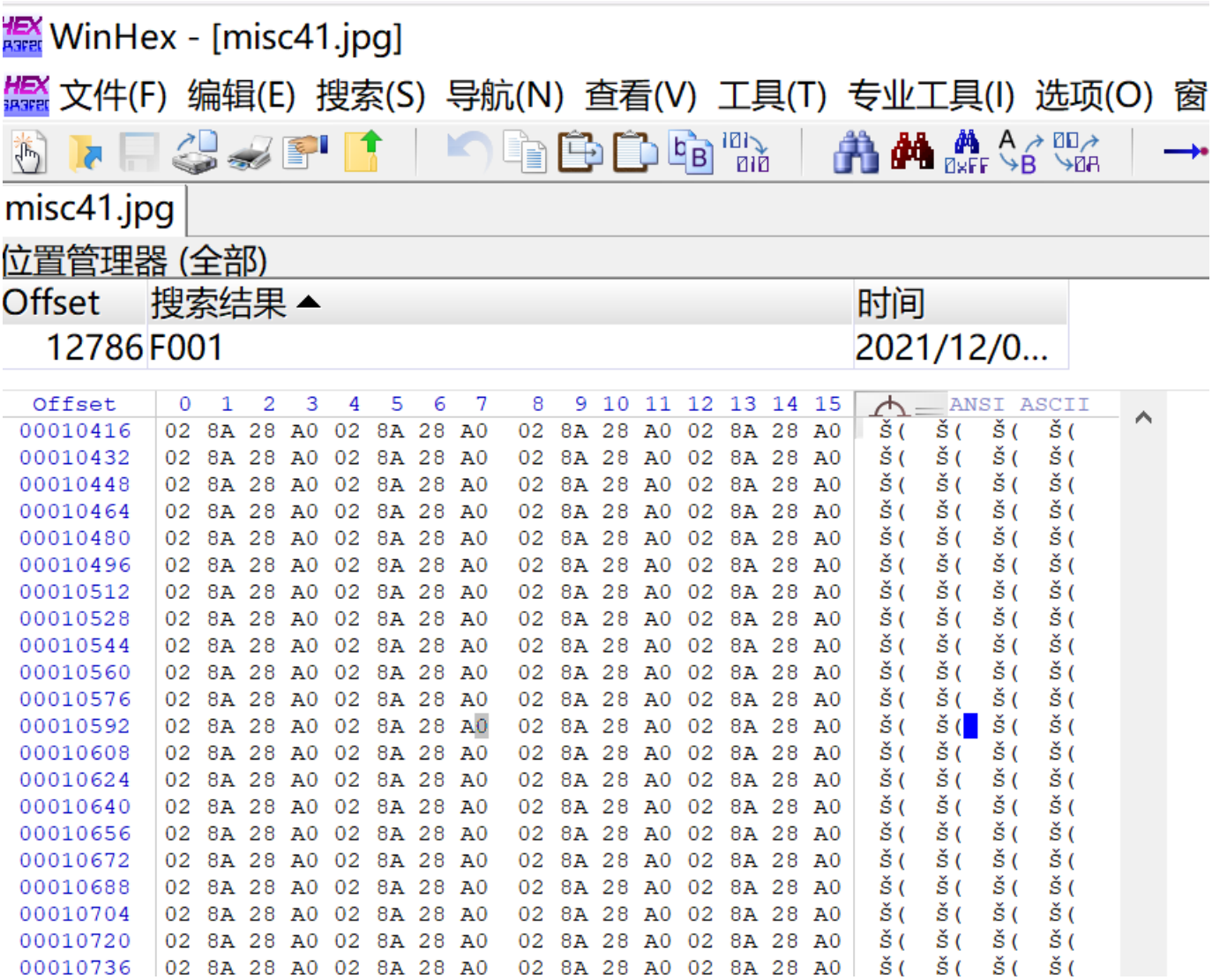
在将转换后的时间戳转为16进制，将四段都这样处理拼起来套上ctfshow{}即可。

MISC 41

提示：
（本题为Misc入门图片篇和愚人节比赛特别联动题）
H4ppy Apr1l F001's D4y!
愚人节到了，一群笨蛋往南飞，一会儿排成S字，一会儿排成B字。
用010editor或者windex搜索F001 全部高亮显示，得到flag
(mac上面的hexfiend好像不能全部高亮)




2A10h: 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 .Š(.Š(.Š(.Š(
2A20h: 02 8A 28 A0 02 8A 28 A0 0F 4D C9 4E 9D 58 55 D8 .Š(.Š(.MÉN.XUØ
2A30h: B5 FD 47 69 53 D7 FF 5B 01 6A F0 01 01 E0 EE DF µýGiS×ÿ[.jð..àîß
2A40h: F0 01 F0 01 F0 01 EA 39 F0 01 F0 01 F0 01 87 55 ð.ð.ð.ê9ð.ð.ð.‡U
2A50h: F0 01 A3 B2 47 4B 4C F6 FC AC F0 01 EF C7 2D A1 ð.£²GKLöü¬ð.ïÇ-¡
2A60h: F0 01 84 80 67 39 B8 BF 67 8B F0 01 1E 8F AB 89 ð.„€g9¸¿g‹ð...«‰
2A70h: F0 01 F0 01 F0 01 EA 0E A3 03 F0 01 F0 01 6C 60 ð.ð.ð.ê.£.ð.ð.l`
2A80h: 05 50 0E 4D 31 A1 21 93 A2 F3 FB 0B D5 ED 4F 0A .P.M1¡!"¢óû.ÕíO.
2A90h: D3 78 F0 01 F0 01 39 6D A4 5B F0 01 F0 01 66 75 Óxð.ð.9m¤[ð.ð.fu
2AA0h: F3 AD F0 01 48 67 0D A4 F0 01 9E 90 47 72 38 72 ó-ð.Hg.¤ð.ž.Gr8r
2AB0h: F0 01 F0 01 F0 01 74 26 F0 01 F0 01 95 C7 F5 FF ð.ð.ð.t&ð.ð.•Çõÿ
2AC0h: C0 38 F0 01 1E 50 00 1A 15 80 8D 0F F0 01 01 D7 À8ð..P...€..ð..×
2AD0h: F0 01 F0 01 F1 06 68 94 F0 01 F0 01 43 07 03 49 ð.ð.ñ.h"ð.ð.C..I
2AE0h: 4B 41 41 C9 9B 0E E8 6A EB 73 E1 D2 76 58 11 4A KAAÉ›.èjësáÒvX.J
2AF0h: F0 01 12 94 0A 13 24 01 FE 15 39 D1 56 68 9F 9A ð.."..$.þ.9ÑVhŸš
2B00h: F0 01 2E 6B 3A 6F C1 F8 F0 01 F0 01 F0 01 D7 16 ð..k:oÁøð.ð.ð.×.
2B10h: F0 01 F0 01 F0 01 CA D2 F0 01 4A E6 F0 01 5E 9B ð.ð.ð.ÊÒð.Jæð.^›
2B20h: F0 01 EC 72 F0 01 DC 88 F0 01 16 27 F0 01 3C 9A ð.ìrð.Üˆð..'ð.<š
2B30h: F0 01 66 62 F0 01 A2 EA F0 01 F0 01 F0 01 1E 6E ð.fbð.¢êð.ð.ð..n
2B40h: F8 EE 08 C9 CA 06 EF 2D FE 04 73 2E B9 C2 AE E2 øî.ÉÊ.ï-þ.s.¹Â®â
2B50h: F0 01 1A BA FE 30 CC 84 F0 01 82 1F F0 01 F0 01 ð..ºþ0Ì„ð.‚.ð.ð.
Find Hex Bytes: f001 All Options F0 01
Find Results
CSDN @_Monica_


WinHex - [misc41.jpg]
文件(F) 编辑(E) 搜索(S) 导航(N) 查看(V) 工具(T) 专业工具(I) 选项(O) 窗
misc41.jpg
位置管理器 (全部)
Offset 搜索结果 时间
12786 F001 2021/12/0...
Offset 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 ANSI ASCII
00010416 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 Š(Š(Š(Š(
00010432 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 Š(Š(Š(Š(
00010448 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 Š(Š(Š(Š(
00010464 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 Š(Š(Š(Š(
00010480 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 Š(Š(Š(Š(
00010496 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 Š(Š(Š(Š(
00010512 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 Š(Š(Š(Š(
00010528 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 Š(Š(Š(Š(
00010544 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 Š(Š(Š(Š(
00010560 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 Š(Š(Š(Š(
00010576 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 Š(Š(Š(Š(
00010592 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 Š(Š(Š(Š(
00010608 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 Š(Š(Š(Š(
00010624 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 Š(Š(Š(Š(
00010640 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 Š(Š(Š(Š(
00010656 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 Š(Š(Š(Š(
00010672 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 Š(Š(Š(Š(
00010688 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 Š(Š(Š(Š(
00010704 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 Š(Š(Š(Š(
00010720 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 Š(Š(Š(Š(
00010736 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 02 8A 28 A0 Š(Š(Š(Š(

```
00010752  02 8A 28 A0 02 8A 28 A0  02 8A 28 A0 02 8A 28 A0  Š( Š( Š( Š(
00010768  02 8A 28 A0 02 8A 28 A0  02 8A 28 A0 02 8A 28 A0  Š( Š( Š( Š(
00010784  02 8A 28 A0 02 8A 28 A0  0F 4D C9 4E 9D 58 55 D8  Š( Š( MÉN XUØ
00010800  B5 FD 47 69 53 D7 FF 5B  01 6A F0 01 01 E0 EE DF  µýGiS×ÿ[ jð  àîß
00010816  F0 01 F0 01 F0 01 EA 39  F0 01 F0 01 F0 01 87 55  ð ð ð ê9ð ð ð ‡U
00010832  F0 01 A3 B2 47 4B 4C F6  FC AC F0 01 EF C7 2D A1  ð £²GKLöü¬ð ïÇ-¡
00010848  F0 01 84 80 67 39 B8 BF  67 8B F0 01 1E 8F AB 89  ð „€g9¸¿g‹ð  «‰
00010864  F0 01 F0 01 F0 01 EA 0E  A3 03 F0 01 F0 01 6C 60  ð ð ð ê £ ð ð l`
00010880  05 50 0E 4D 31 A1 21 93  A2 F3 FB 0B D5 ED 4F 0A  P M1¡!"¢óû ÕíO
00010896  D3 78 F0 01 F0 01 39 6D  A4 5B F0 01 F0 01 66 75  Óxð ð 9m¤[ð ð fu
00010912  F3 AD F0 01 48 67 0D A4  F0 01 9E 90 47 72 38 72  ó-ð Hg ¤ð ž Gr8r
00010928  F0 01 F0 01 F0 01 74 26  F0 01 F0 01 95 C7 F5 FF  ð ð ð t&ð ð •Çõÿ
00010944  C0 38 F0 01 1E 50 00 1A  15 80 8D 0F F0 01 01 D7  À8ð  P   €  ð  ×
00010960  F0 01 F0 01 F1 06 68 94  F0 01 F0 01 43 07 03 49  ð ð ñ h"ð ð C  I
00010976  4B 41 41 C9 9B 0E E8 6A  EB 73 E1 D2 76 58 11 4A  KAAÉ› èjësáÒvX J
00010992  F0 01 12 94 0A 13 24 01  FE 15 39 D1 56 68 9F 9A  ð  "  $ þ 9ÑVhŸš
00011008  F0 01 2E 6B 3A 6F C1 F8  F0 01 F0 01 F0 01 D7 16  ð .k:oÁøð ð ð ×
00011024  F0 01 F0 01 F0 01 CA D2  F0 01 4A E6 F0 01 5E 9B  ð ð ð ÊÒð Jæð ^›
00011040  F0 01 EC 72 F0 01 DC 88  F0 01 16 27 F0 01 3C 9A  ð ìrð Üˆð  'ð <š
00011056  F0 01 66 62 F0 01 A2 EA  F0 01 F0 01 F0 01 1E 6E  ð fbð ¢êð ð ð  n
00011072  F8 EE 08 C9 CA 06 EF 2D  FE 04 73 2E B9 C2 AE E2  øî ÉÊ ï-þ s.¹Â®â
00011088  F0 01 1A BA FE 30 CC 84  F0 01 82 1F F0 01 F0 01  ð  °þ0Ì„ð , ð ð
00011104  F0 01 B9 54 F0 01 E5 80  F0 01 9E 3E F0 01 84 7A  ð ¹Tð å€ð ž>ð „z
00011120  F0 01 4B 45 F0 01 7D 15  F0 01 F0 01 F0 01 DC 10  ð KEð } ð ð ð Ü
00011136  F0 01 7D 6D F0 01 0A 8C  F0 01 49 9A F0 01 EE 88  ð }mð  Œð Išð îˆ
00011152  D8 B4 F0 01 B4 C8 F0 01  5B 12 D4 61 F0 01 F0 01  Ø´ð ´Èð [ Ôað ð
```


图片篇(文件结构)

MISC 24

提示：flag在图片上面。