




ctfshow中Misc入门WP（超级全）

原创

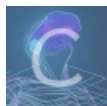
罡罡同学  于 2021-07-26 21:08:19 发布  3093  收藏 10

分类专栏: [ctfshow-misc](#) 文章标签: [ctfshow](#) [Misc](#) [CTF](#) [杂项](#) [信息安全](#)

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/m0_46625346/article/details/119083860

版权



[ctfshow-misc](#) 专栏收录该内容

18 篇文章 2 订阅

订阅专栏

misc17

提示: flag在图片数据里。

先binwalk分析, 没啥问题, 然后试试zsteg

如何安装zsteg呢?

```
git clone https://github.com/zed-0xff/zsteg
```

```
cd zsteg/
```

```
gem install zsteg
```

```

(root@kali2021)-[~/桌面]
# zsteg misc17.png
[?] 3544 bytes of extra data after zlib stream
extradata:0
00000000: e1 1f 30 53 86 4f c5 a4 1b f5 e6 e5 c7 46 0a 92 |..0S.O.....F..
00000010: 9b ee 72 e7 c9 9e b9 a7 74 de 92 4d ad 61 5b 58 |..r.....t..M.a[X
00000020: f2 98 65 77 2b d2 d3 85 32 fc 08 83 86 1f 0f 1e |..ew+ ...2.....
00000030: cb ab ac 9c 4b ca 02 20 e2 ce e4 ae 60 1a 2c c6 |....K.. ..`.,.
00000040: 7b c8 9a 77 31 2f 9e 67 db d9 3e 53 fe 17 a5 50 |{..w1/.g..>S...P
00000050: 20 e5 1d 8c d5 49 4e 52 a5 54 31 cb 8b c5 3b 09 |...INR.T1...;.
00000060: a2 a6 fe 5b da 4f 9e 78 9c 5d 46 d6 e2 6b 6b 2a |... [.0.x.]F..kk*
00000070: f2 62 0c ba 70 19 a0 27 f3 84 77 99 02 77 05 79 |.b..p.. ' ..w..w.y
00000080: 5b 44 b7 79 b3 54 11 a1 f3 54 34 56 7e ff 55 d1 |[D.y.T...T4V~.U.
00000090: c6 39 90 c8 21 7f 26 39 44 58 78 c3 ed 37 4a 7c |.9..!.69DXx..7j|
000000a0: 50 24 e8 79 7b 4b 9c fa 2a 2c bb e8 b9 fb 40 2c |P$.y{K..*,...@,
000000b0: 50 05 21 4c 3b 29 65 b4 60 1c 27 bb 4c 16 bf f1 |P.!L;)e.`.'L...
000000c0: 77 c0 55 04 5e 25 0e 18 1e 58 ab 0f 13 11 f2 3f |w.U.^%...X.....?
000000d0: cf a0 32 b1 f5 a8 1b 99 a7 4b 46 89 cf 85 89 50 |..2.....KF....P
000000e0: 88 20 8f 4f fd e2 97 55 68 73 b4 96 ba dd 25 a3 |. .0 ... Uhs....%.
000000f0: 83 72 3f 99 77 9e 0a 08 50 4f 11 8f 47 65 09 46 |

```

发现隐藏的数据，位置处于 extradata:0

将数据提取出来：zsteg -E "extradata:0" misc17.png > 1.txt

然后再 binwalk 1.txt -e 把1.txt中的数据分离出来，拿到flag

ctfshow{0fe61fc42e8bbe55b9257d251749ae45}

misc18

提示：flag在标题、作者、照相机和镜头型号里。

描述	1
说明	
标题	ctfshow{32
主题	
分级	☆☆☆☆☆
标记	
备注	
来源	
作者	5d60c208f7
拍摄日期	
程序名称	
获取日期	
URL	https://blog.csdn.net/m0_46625346

描述	1
照相机制造商	
照相机型号	28ac17e5f0
光圈值	
曝光时间	
ISO 速度	
曝光补偿	
焦距	
最大光圈	
测光模式	
目标距离	
闪光灯模式	
闪光灯能量	
35mm 焦距	
高级照片	
镜头制造商	
镜头型号	2d4cf5a839}
闪光灯制造商	

ctfshow{325d60c208f728ac17e5f02d4cf5a839}

misc19

提示：flag在主机上的文档名里。

用matlab查看文件信息

```
info = imfinfo('misc19.tif')
```

```
>> info = imfinfo('misc19.tif')
```

```
info =
```

```
Filename: 'D:\img\misc19.tif'  
FileModDate: '25-3月-2021 03:12:27'  
FileSize: 26172  
Format: 'tif'
```

```
FormatVersion: []
      Width: 900
      Height: 150
      BitDepth: 24
      ColorType: 'truecolor'
FormatSignature: [73 73 42 0]
      ByteOrder: 'little-endian'
NewSubFileType: 0
      BitsPerSample: [8 8 8]
      Compression: 'LZW'
PhotometricInterpretation: 'RGB'
      StripOffsets: [21688 25422]
SamplesPerPixel: 3
      RowsPerStrip: 97
StripByteCounts: [3733 749]
      XResolution: 72
      YResolution: 72
ResolutionUnit: 'Inch'
      Colormap: []
PlanarConfiguration: 'Chunky'
      TileWidth: []
```

fx

https://blog.csdn.net/m0_46625346

```
ColorMap: []
PlanarConfiguration: 'Chunky'
      TileWidth: []
      TileLength: []
      TileOffsets: []
      TileByteCounts: []
Orientation: 1
      FillOrder: 1
GrayResponseUnit: 0.0100
      MaxSampleValue: [255 255 255]
      MinSampleValue: [0 0 0]
      Thresholding: 1
      Offset: 8
      DocumentName: 'ctfshow{dfdcf08038cd446a5}'
      Software: 'Adobe Photoshop CC 2019 (Windows)'
      DateTime: '2021:03:25 10:35:18'
      HostComputer: 'eb50782f8d3605d}'
      Predictor: 'Horizontal differencing'
      XMP: '<?xpacket begin="          " id="W5M0MpCehiHzreSzNTczkc9d"?>...'
      Photoshop: [1x3464 double]
      DigitalCamera: [1x1 struct]
      ICCProfileOffset: 18502
```

https://blog.csdn.net/m0_46625346

DocumentName和HostComputer连起来就是flag啦!

ctfshow{dfdcf08038cd446a5eb50782f8d3605d}

misc20

提示: flag在评论里。

exif信息查看在线网站, 上传图片, 看到信息(谐音可还行)

图虫EXIF查看器 - 查询结果



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)

https://blog.csdn.net/m0_46625346

ctfshow{c97964b1aecf06e1d79c21ddad593e42}

misc21

提示: flag在序号里。

ExifFD

Exif版本	0232
ComponentsConfiguration	Y, Cb, Cr, -
SecurityClassification	Top Secret
Flashpix版本	0100
色彩空间	Uncalibrated
序列号	686578285826597329

https://blog.csdn.net/m0_46625346

686578285826597329

转字符得到hex(X&Ys)

```
import binascii
str='686578285826597329'
print(binascii.a2b_hex(str))
```

b'hex(X&Ys)'

Process finished with exit code 0

发现上面有两组与XY有关的数据，中间还有https://ctf.show/和ctfshow{}

根据提示hex(X&Ys)，应该是要把这里的十进制数值转为十六进制

把四段拼起来得到3902939465237161861910824528172980145261，然后转十六进制，再套上ctfshow{}，如果不是整体直接转换的话，每段分别转hex，然后拼起来

最终得到：ctfshow{e8a221498d5c073b4084eb51b1a1686d}

填写所需检测的密码：

3902939465

结果：

e8a22149

米斯特安全团队 CTFCrakTools pr

解码方式 进制转换 插件 妹子

填写所需检测的密码：(已输入字符数统

2371618619

结果：

8d5c073b

后面一样的道理。。。

ctfshow{e8a221498d5c073b4084eb51b1a1686d}

misc22

提示: flag在图片里。

MagicExif下载网址

用magicexif打开, 直接发现flag

这里有个bug, 工具放到移动硬盘里打不开。。。必须放到本地磁盘。。。不知道为啥



```
ctfshow{dbf7d3f84b0125e833dfd3c80820a129}
```

misc23

没有exiftool的, 先在kali中装一下吧

```
apt-get install exiftool
```

命令 `exiftool misc23.psd`

Timestamp指的是时间戳, DECtoHEX是十进制转十六进制

这里利用在线网站获取时间戳

```
root@kali2021: ~/桌面
文件 动作 编辑 查看 帮助
Image Width : 900
Bit Depth : 8
IPTC Digest : 00000000000000000000000000000000
XMP Toolkit : Image::ExifTool 11.98
Format : application/vnd.adobe.photoshop
Color Mode : RGB
Text Layer Name : {there is no flag here}
Text Layer Text : {there is no flag here}
Create Date : 2021:03:25 15:45:24+08:00
Creator Tool : Adobe Photoshop CC 2019 (Windows)
Metadata Date : 2021:03:25 16:02:50+08:00
Modify Date : 2021:03:25 16:02:50+08:00
Document ID : xmp.did:49520599-6932-e144-8f4b-dfd5873be5bc
History Action : ctfshow{, UnixTimestamp, DECtoHEX, getflag
History Instance ID : xmp.iid:1, xmp.iid:2, xmp.iid:3, xmp.iid:4
History Software Agent : Adobe Photoshop CC 2019 (Windows), Adobe Photoshop CC
2019 (Windows), Adobe Photoshop CC 2019 (Windows), Adobe Photoshop CC 2019 (Windows)
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
History Changed : /
Instance ID : xmp.iid:06e30d4e-08bd-0246-815c-0c8c684a0c81
Original Document ID : xmp.did:49520599-6932-e144-8f4b-dfd5873be5bc
X Resolution : 72
Displayed Units X : inches
Y Resolution : 72
Displayed Units Y : inches
https://blog.csdn.net/m0_46625346
```

现在: 1627477659

控制: ■ 停止

时间戳 1627477072 秒(s) 转换 >> 北京时间

时间 1997-09-22 02:17:02 北京时间 转换 >> 874865822 秒(s)

最后得到4段

874865822 2699237688 2156662245 460377706

分别hex后拼在一起

得到: ctfshow{3425649ea0e31938808c0de51b70ce6a}

misc41

提示:

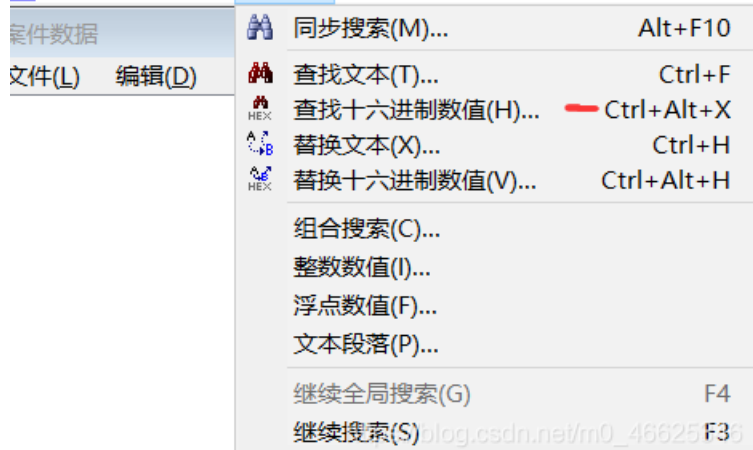
H4ppy Apr1l F001's D4y!

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。

第一句提示的F001是突破点，这个位置有大量F001，看起来组成了某种形状
Winhex打开图片，搜索F001，全部高亮！

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(P)



```
00010800 B5 FD 47 69 53 D7 FF 5B 01 6A F0 01 01 E0 EE DF
00010816 F0 01 F0 01 F0 01 EA 39 F0 01 F0 01 F0 01 87 55
00010832 F0 01 A3 B2 47 4B 4C F6 FC AC F0 01 EF C7 2D A1
00010848 F0 01 84 80 67 39 B8 BF 67 8B F0 01 1E 8F AB 89
00010864 F0 01 F0 01 F0 01 EA 0E A3 03 F0 01 F0 01 6C 60
00010880 05 50 0E 4D 31 A1 21 93 A2 F3 FB 0B D5 ED 4F 0A
00010896 D3 78 F0 01 F0 01 39 6D A4 5B F0 01 F0 01 66 75
00010912 F3 AD F0 01 48 67 0D A4 F0 01 9E 90 47 72 38 72
00010928 F0 01 F0 01 F0 01 74 26 F0 01 F0 01 95 C7 F5 FF
00010944 C0 38 F0 01 1E 50 00 1A 15 80 8D 0F F0 01 01 D7
00010960 F0 01 F0 01 F1 06 68 94 F0 01 F0 01 43 07 03 49
00010976 4B 41 41 C9 9B 0E E8 6A EB 73 E1 D2 76 58 11 4A
00010992 F0 01 12 94 0A 13 24 01 FE 15 39 D1 56 68 9F 9A
00011008 F0 01 2E 6B 3A 6F C1 F8 F0 01 F0 01 F0 01 D7 16
00011024 F0 01 F0 01 F0 01 CA D2 F0 01 4A E6 F0 01 5E 9B
00011040 F0 01 EC 72 F0 01 DC 88 F0 01 16 27 F0 01 3C 9A
00011056 F0 01 66 62 F0 01 A2 EA F0 01 F0 01 F0 01 1E 6E
00011072 F8 EE 08 C9 CA 06 EF 2D FE 04 73 2E B9 C2 AE E2
00011088 F0 01 1A BA FE 30 CC 84 F0 01 82 1F F0 01 F0 01
00011104 F0 01 B9 54 F0 01 E5 80 F0 01 9E 3E F0 01 84 7A
00011120 F0 01 4B 45 F0 01 7D 15 F0 01 F0 01 F0 01 DC 10
00011136 F0 01 7D 6D F0 01 0A 8C F0 01 49 9A F0 01 EE 88
00011152 D8 B4 F0 01 B4 C8 F0 01 5B 12 D4 61 F0 01 F0 01
00011168 AF 4E 61 3D 98 01 B4 A9 8E 16 5B 91 67 9E 5B A6
00011184 64 BB F0 01 F0 01 21 EA BE 99 3B FD 31 C5 02 42
00011200 B9 F3 F0 01 19 CB 06 4B F0 01 F0 01 F0 01 6C 06
00011216 F0 01 F0 01 F0 01 26 C5 F0 01 12 2B 8B BE C5 33
00011232 96 5F F0 01 FA 47 F8 F6 F0 01 C0 76 B2 E7 14 1D
00011248 F0 01 F0 01 E3 B6 CF FE F0 01 F0 01 F0 01 0E 7C
00011264 1C 62 85 58 6F AE 32 A0 03 B9 97 B9 75 AF F3 3C
00011280 F0 01 FA CF 43 5C 11 D1 B8 FC FA 3B F0 01 37 F8
00011296 F0 01 D3 16 C9 14 AC BB DF BA B2 68 F0 01 7D 87
00011312 F0 01 F0 01 F0 01 7D 38 F0 01 F0 01 F0 01 20 27
00011328 F0 01 1F 1C F0 01 09 4F F0 01 87 3D F0 01 CD 0E
00011344 F0 01 F0 01 F0 01 5E CF F0 01 F0 01 F0 01 A7 B1
00011360 4C C3 86 FF 49 8A 4D 05 40 07 EF 21 BA 38 29 5B
00011376 F0 01 48 B1 F0 01 AF 8E F0 01 F0 01 F0 01 1F 96
00011392 F0 01 FC E2 F0 01 DE 79 84 EC 5E A4 F0 01 0A 49
00011408 F0 01 F0 01 F0 01 C7 B4 F0 01 F0 01 F0 01 BA 7B
```

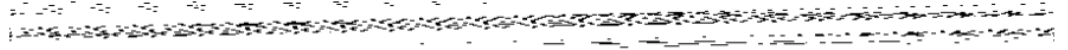
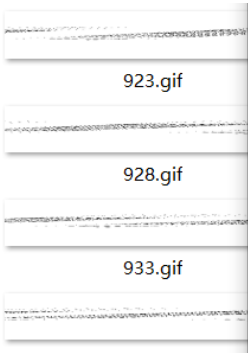
后面还有不少。。。不在放图了。。。

ctfshow{fcbd427ca4a52f1147ab44346cd1cdd}

misc36

提示：出题人坦白从宽，正确的宽度在920-950之间

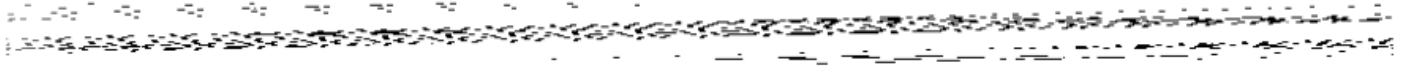
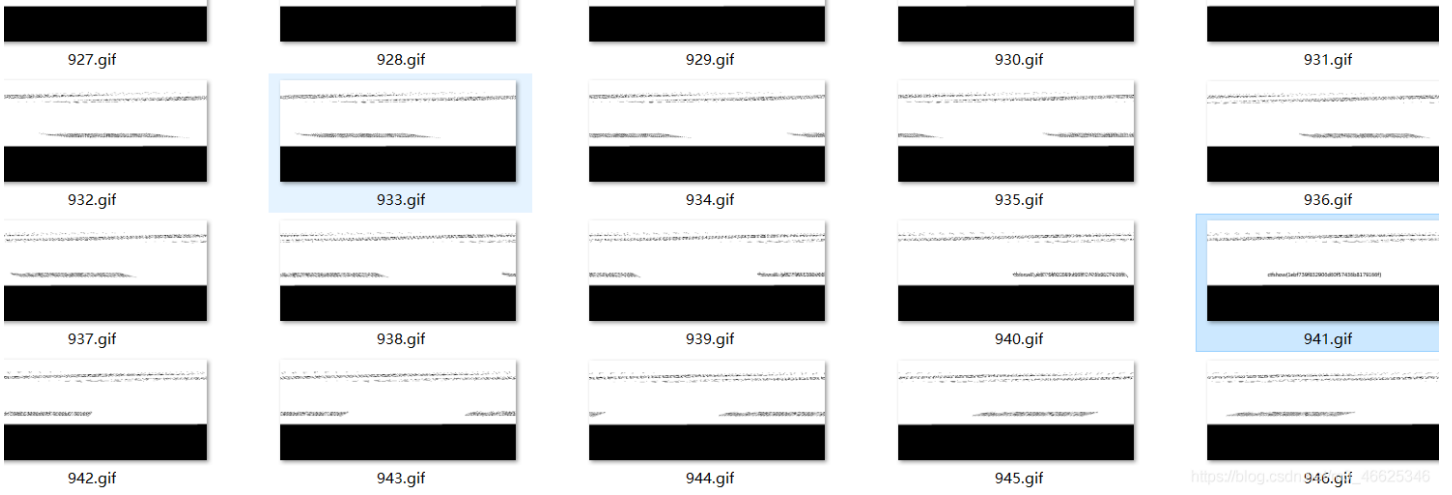
脚本爆破宽度，先把gif的高度拉高，否则会这样子。。。 (啥都没有)



https://blog.csdn.net/m0_46625346

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
00000000	47	49	46	38	39	61	84	03	90	01	91	00	00	00	00	00
00000016	FF	FF	FF	FF	FF	FF	00	00	00	21	F9	04	01	00	00	02
00000032	00	2C	00	00	00	00	84	03	90	01	00	02	FF	8C	8F	A9
00000048	CB	ED	0F	A3	9C	B4	DA	8B	B3	DE	BC	FB	0F	86	E2	48
00000064	96	E6	89	A6	EA	CA	B6	EE	0B	C7	F2	4C	D7	F6	8D	E7
00000080	FA	CE	F7	FE	0F	0C	0A	87	C4	A2	F1	88	4C	2A	97	CC
00000096	A6	F3	09	8D	4A	A7	D4	AA	F5	8A	CD	6A	B7	DC	AE	F7
00000112	0B	0E	8B	C7	E4	B2	F9	8C	4E	AB	D7	EC	B6	FB	0D	8F
00000128	CB	E7	F4	BA	FD	8E	CF	EB	F7	FC	BE	FF	0F	18	28	38
00000144	48	58	68	78	88	98	A8	B8	C8	D8	E8	F8	08	19	29	39
00000160	49	59	69	79	89	99	A9	B9	C9	D9	E9	F9	09	1A	2A	3A
00000176	4A	5A	6A	7A	8A	9A	AA	BA	CA	DA	EA	FA	0A	1B	2B	3B
00000192	4B	5B	6B	7B	8B	9B	AB	BB	CB	DB	EB	FB	0B	1C	2C	3C
00000208	4C	5C	6C	7C	8C	9C	AC	BC	CC	DC	EC	FC	0C	1D	2D	3D
00000224	4D	5D	6D	7D	8D	9D	AD	BD	CD	DD	ED	FD	0D	1E	2E	3E
00000240	4E	5E	6E	7E	8E	9E	AE	BE	CE	DE	EE	FE	0E	1F	2F	3F

```
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()
```



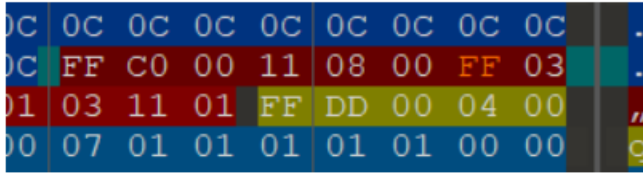
ctfshow{1ebf739f832906d60f57436b8179166f}

https://blog.csdn.net/m0_46625346

ctfshow{1ebf739f832906d60f57436b8179166f}

misc27

图片是jpg, 修改图片高度, 把150的十六进制0096改成00FF得到flag



{there_is_no_flag_here}

ctfshow{5cc4f19eb01705b99bf41492430a1a14}

CSDN @墨墨同学

misc28

gif的每一帧都有宽高所以修改的地方不止一处

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	ANSI ASCII	
00000000	47	49	46	38	39	61	84	03	FF	00	C4	00	00	00	00	00	GIF89a,, y Ä	
00000016	FF	FF	FF	F4	F4	F4	E9	E9	E9	DD	DD	DD	D1	D1	D1	C5	yÿÿôôôéééÝÝÝŃŃŃ	
00000032	C5	C5	B8	B8	B8	AA	AA	AA	9C	9C	9C	8D	8D	8D	7D	7D	ÄÄ, ,, a a a cecece }	
00000048	7D	6B	6B	6B	58	58	58	42	42	42	26	26	26	FF	FF	FF	}kkkXXXBBB&&&yÿÿ	
00000064	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000096	00	00	00	00	00	00	00	00	00	00	00	00	00	00	21	F9	04	!ù
00000112	01	00	00	10	00	2C	00	00	00	00	84	03	FF	00	00	05	, " y	
00000128	FF	60	20	8E	64	69	9E	68	AA	AE	6C	EB	BE	70	2C	CF	y` ždižh*@le³p, İ	
00000144	74	6D	DF	78	AE	EF	7C	EF	FF	C0	A0	70	48	2C	1A	8F	tmBx@i iYÀ pH,	
00000160	C8	A4	72	C9	6C	3A	9F	D0	A8	74	4A	AD	5A	AF	D8	AC	È=rÉl:YÐ``tJ-Z~ø~	
00000176	76	CB	ED	7A	BF	E0	B0	78	4C	2E	9B	CF	E8	B4	7A	CD	vËízç;à°xL. >İè'zÍ	
00000192	6E	BB	DF	F0	B8	7C	4E	AF	DB	EF	F8	BC	7E	CF	EF	FB	n»ßø, N`Ôiø~İiù	
00000208	FF	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	yè ,f,.....t+^%š<G ž	
00000224	8F	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	Y`çE`B`q`ž`š`œ`ž`	
00000240	9F	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	CSDN @墨墨同学	

{there_is_no_flag_here}

ctfshow{59c8bc525426166b1c893fe12a387fd7}

CSDN @墨墨同学

misc29

对gif全部帧进行替换。

搜索全部的96 00换成FF 00

{there_is_no_flag_here}

ctfshow{03ce5be6d60a4b3c7465ab9410801440}

CSDN @ 墨墨同学

Hex editor interface showing search results for '96 00' replaced with 'FF 00'. The search menu is open, and the results table is visible below.

地址	内容
00022288	45 96 0D 98 A0 45 59 6B 6D 8A B1 D3 6A 6C F9
00022304	B7 74 7B B1 FB 47 75 97 19 90 FF 99 5B DA 29
00022320	7F 28 B8 00 07 91 B4 06 BB 78 BA 7C 0E B7 7E
00022336	C1 C1 99 96 16 1E 8C 18 21 FC C1 66 31 C2 8E
00022352	C2 24 3C 16 28 3C 6A 89 9B C2 70 B1 C2 2C 1C
00022368	2E 7C 17 30 5C 18 62 37 C3 60 71 C3 8F 51 C3
00022384	9C 5F 32 DC C3 40 1C C4 42 3C C4 44 5C C4 4E
00022400	C4 48 9C C4 4A 0C 20 01 01 00 21 F9 04 01 32
00022416	02 00 2C 00 00 00 00 84 03 FF 00 87 00 00 0C
00022432	FF FF FF FF FF 00 00 00 00 00 00 00 00 0C
00022448	00 00 00 00 00 00 00 00 00 00 00 00 00 0C

ctfshow{03ce5be6d60a4b3c7465ab9410801440}

misc30

在宽度的位置进行修改成950的二进制03 B6，注意要倒着写。

chall	misc30.bmp															ANSI	ASCII	
Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
00000000	42	4D	50	87	06	00	00	00	00	00	36	00	00	00	28	00	BMP+	6 (
00000016	00	00	B6	03	00	00	96	00	00	00	01	00	18	00	00	00	█	-
00000032	00	00	1A	87	06	00	12	0B	00	00	12	0B	00	00	00	00	+	
00000048	00	00	00	00	00	00	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	Y	Y
00000064	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	Y	Y
00000080	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	Y	Y
00000096	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	Y	Y
00000112	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	Y	Y

ctfshow{6db8536da312f6aeb42da2f45b5f213c}

ctfshow{6db8536da312f6aeb42da2f45b5f213c}

misc31

通过这篇博客了解一下bmp文件结构

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487253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：(487253-53)/3=162400

F FF FF	FF FF FF	FF FF FF	FF FF FF	FF FF FF	FF FF FF	FF FF FF	FF FF FF	Y
F FF FF	FF FF FF	FF FF FF	FF FF FF	FF FF FF	FF FF FF	FF FF FF	FF FF FF	Y
F FF FF	FF FF FF	FF FF FF	FF FF FF	FF FF FF	FF FF FF	FF FF FF	FF FF FF	Y
F FF FF	FF FF FF	FF FF FF	FF FF FF	45	4E	00	00	Y

偏移地址: 427,853 CSDN @ 罡罡同学

这题的高度是对的，所以正确的宽度是

162400/150=1082

bmp文件修改宽度，借助winhex，我这里宽改到43A（也就是3A 04）

ctfshow{fb09dcc9005fe3feefb73646b55efd5}

CSDN @罡罡同学

ctfshow{fb09dcc9005fe3feefb73646b55efd5}

misc32

CRC爆破宽度

```
PCRT
PNG Check & Repair Tool

Project address: https://github.com/sherlly/PCRT
Author: sherlly
Version: 1.1

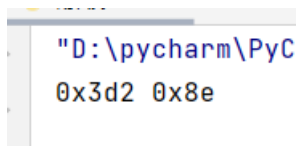
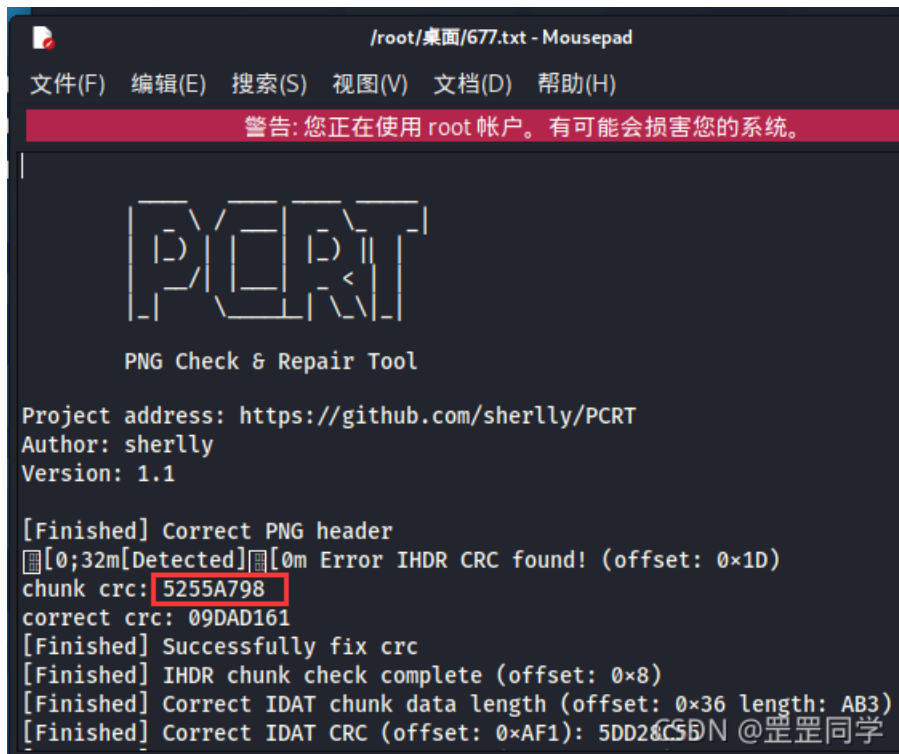
[Finished] Correct PNG header
[0;32m[Detected][0m Error IHDR CRC found! (offset: 0x1D)
chunk crc: E14A4C0B
correct crc: 09DAD161
[Finished] Successfully fix crc
[Finished] IHDR chunk check complete (offset: 0x8)
[Finished] Correct IDAT chunk data length (offset: 0x21 length: AFF)
[Finished] Correct IDAT CRC (offset: 0xB28): 62CB2905
```

```
import struct
import binascii
import os

m = open("misc32.png", "rb").read()
k = 0
for i in range(5000):
    if k == 1:
        break
    for j in range(5000):
        c = m[12:16] + struct.pack('>i', i) + struct.pack('>i', j) + m[24:29]
        crc = binascii.crc32(c) & 0xffffffff
        if crc == 0xE14A4C0B:
            k = 1
            print(hex(i), hex(j))
            break
```


ctfshow{685082227bcf70d17d1b39a5c1195aa9}

misc33



ctfshow{03070a10ec3a3282ba1e352f4e07b0a9}

ctfshow{03070a10ec3a3282ba1e352f4e07b0a9}

misc34

由于CRC也被修改了，所以我们对宽度进行爆破，然后自己找图片


```
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()
```

ctfshow{03e102077e3e5de9dd9c04aba16ef014}

ctfshow{03e102077e3e5de9dd9c04aba16ef014}

misc37

misc43

提示：错误中隐藏着通往正确答案的道路

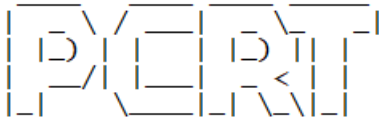
猜测与crc错误有关。

利用PCRT提取

PCRT：一款自动化检测修复PNG损坏的取证工具

可以在kali中安装PCRT：git clone https://github.com/sherlly/PCRT.git

然后 python PCRT.py -y -v -i misc44.png > 666.txt



PNG Check & Repair Tool

Project address: https://github.com/sherlly/PCRT

Author: sherlly

Version: 1.1

```
[Finished] Correct PNG header
[Finished] Correct IHDR CRC (offset: 0x1D): 09DAD161
[Finished] IHDR chunk check complete (offset: 0x8)
[Finished] Correct IDAT chunk data length (offset: 0x21 length: 180)
[0;32m[Detected][0m Error IDAT CRC found! (offset: 0x1A9)
chunk crc: E59387E5
correct crc: 8385F691
[Finished] Successfully fix crc
[Finished] Correct IDAT chunk data length (offset: 0x1AD length: 180)
[0;32m[Detected][0m Error IDAT CRC found! (offset: 0x335)
chunk crc: 93A62E63
correct crc: 42434298
[Finished] Successfully fix crc
[Finished] Correct IDAT chunk data length (offset: 0x220 length: 180)
```

https://blog.csdn.net/m0_46625346

把错误的code提出来用hex转字符

将这些提取出来，然后转16进制就可以。

E59387E5 93A62E63 74667368 6F777B36 65623235 38396666 66663565 33393066 65366238 37353034 64626330
3839327D

```
import binascii
str='E59387E593A62E63746673686F777B36656232353839666666663565333930666536623837353034646263303839327D'
print(binascii.a2b_hex(str))
```

b'\xe5\x93\x87\xe5\x93\xa6.ctfshow{6eb2589ffff5e390fe6b87504dbc0892}'

misc44

提示：错误中还隐藏着坑

一种CRC32隐写，错误的CRC32和正确的CRC32分别代表着01，再8位一组转字符，劝大家不要拖进tweakpng，因为会有几百个弹窗。。。

至于如何提取正确和错误的CRC32，我的做法是用PCRT识别再放入txt，再写个脚本，就比较容易

PCRT：一款自动化检测修复PNG损坏的取证工具

可以在kali中安装PCRT：git clone https://github.com/sherlly/PCRT.git

然后 python PCRT.py -y -v -i misc44.png > 666.txt

python脚本代码：

```

f = open('666.txt')
res = ''
while 1:
    c = f.readline()
    if c:
        if 'chunk crc' in c:
            # print(c)
            res+='0'
        elif 'Correct IDAT CRC' in c:
            res+='1'
        else:
            break

print(res)
print(len(res))
for i in range(len(res)//8):
    a = res[i*8:i*8+8]
    try:
        print(chr(int(a,2)),end='')
    except:
        pass

```

ctfshow{cc1af32bf96308fc1263231be783f69e}

misc45

提示：有时候也需要换一换思维格式
一个新的知识点。

具体做法就是：先把图片从png格式转为bmp格式，然后直接binwalk提取就能得到flag.png了。
我是matlab转的图片格式，两行代码就可以哦，当然大家也可以去在线网站转换格式。

matlab代码：

```

I=imread('misc45.png');
imwrite(I,'d:\455.bmp');

```

先读取png图片，然后重写转换图片格式为bmp，路径大家自行调整即可。

```

(root@kali2021)-[~/桌面]
# binwalk 455.bmp

```

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	PC bitmap, Windows 3.x format,, 900 x 150 x 24
65536	0x10000	gzip compressed data, has original file name: "flag.png", from Unix, last modified: 2021-03-29 15:44:52

然后我们用binwalk 455.bmp -e 分离文件即可。

```

(root@kali2021)-[~/桌面]
# binwalk 455.bmp -e

```

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	PC bitmap, Windows 3.x format,, 900 x 150 x 24
65536	0x10000	gzip compressed data, has original file name: "flag.png", from Unix, last modified: 2021-03-29 15:44:52

ctfshow{057a722a5587979c34966c2436283e70}

misc46

用gif每一帧的偏移量作为坐标来画图即可，这里gif的偏移量我是用identify命令直接获取的

kali中的identify需要安装一下，在root权限下：`apt-get install imagemagick`

`identify misc46.gif > 2.txt`

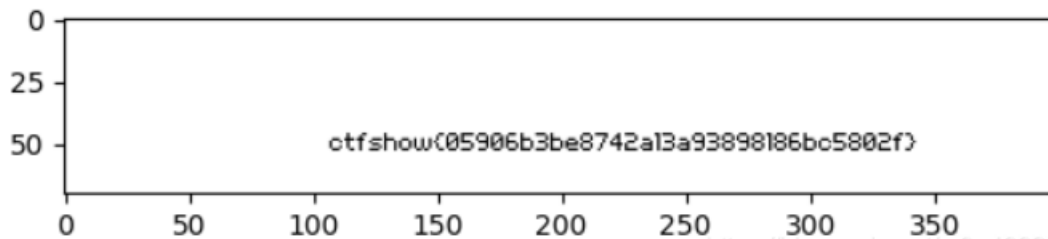
```
文件(F) 编辑(E) 搜索(S) 视图(V) 文档(D) 帮助(H)
警告:您正在使用 root 帐户。有可能会损害您的系统。
misc46.gif[0] GIF 900x150 900x150+0+0 8-bit sRGB 2c 0.010u 0:00.022
misc46.gif[1] GIF 450x50 900x150+174+49 8-bit sRGB 16c 0.010u 0:00.035
misc46.gif[2] GIF 450x50 900x150+196+47 8-bit sRGB 16c 0.010u 0:00.035
misc46.gif[3] GIF 450x50 900x150+256+49 8-bit sRGB 16c 0.010u 0:00.036
misc46.gif[4] GIF 450x50 900x150+293+52 8-bit sRGB 16c 0.010u 0:00.036
misc46.gif[5] GIF 450x50 900x150+220+49 8-bit sRGB 16c 0.010u 0:00.036
misc46.gif[6] GIF 450x50 900x150+245+47 8-bit sRGB 16c 0.010u 0:00.035
misc46.gif[7] GIF 450x50 900x150+172+48 8-bit sRGB 16c 0.010u 0:00.035
misc46.gif[8] GIF 450x50 900x150+342+49 8-bit sRGB 16c 0.010u 0:00.035
misc46.gif[9] GIF 450x50 900x150+331+46 8-bit sRGB 16c 0.010u 0:00.035
misc46.gif[10] GIF 450x50 900x150+319+52 8-bit sRGB 16c 0.010u 0:00.035
misc46.gif[11] GIF 450x50 900x150+200+49 8-bit sRGB 16c 0.010u 0:00.035
misc46.gif[12] GIF 450x50 900x150+233+52 8-bit sRGB 16c 0.010u 0:00.035
misc46.gif[13] GIF 450x50 900x150+144+49 8-bit sRGB 16c 0.010u 0:00.035
misc46.gif[14] GIF 450x50 900x150+254+52 8-bit sRGB 16c 0.010u 0:00.035
```

再写个脚本画图即可（首次使用，要安装一下matplotlib 用命令`pip install matplotlib`）

```
from PIL import Image
import matplotlib.pyplot as plt
f = open('2.txt')
pp = []
while 1:
    c = f.readline()
    if c:
        s = eval(c.split('+')[1]+'+',c.split('+')[2][:2])
        pp.append(s)
        print(s)
        # print(c)
    else:
        break

img = Image.new('RGB', (400, 70), (255, 255, 255))
for i in pp:
    new = Image.new('RGB', (1, 1), (0, 0, 0))
    img.paste(new, i)
plt.imshow(img)
plt.show()
```

Figure 1



https://blog.csdn.net/m0_46625346

ctfshow{05906b3be8742a13a93898186bc5802f}

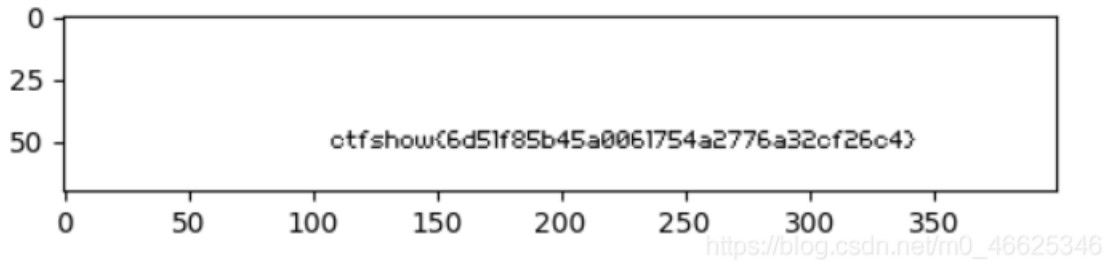
misc47

给了一个png，打开发现没内容，用浏览器打开，确认是apng

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

```
import struct
from PIL import Image
import matplotlib.pyplot as plt
f = open('misc47.png', 'rb')
c = f.read()
c = c[c.index(bytes.fromhex('6663544C00000001')):]
pp = []
for i in range(1,1124,2):
    start = c.index(bytes.fromhex('6663544C0000')+struct.pack('>h',i))
    # start = c.index(bytes.fromhex('6663544C000000'+hex(i)[2:]))
    # print(start)
    fc = c[start:start+30]
    print(fc[18:20],fc[22:24])
    print(struct.unpack('>h',fc[18:20])+struct.unpack('>h',fc[22:24]))
    pp.append(struct.unpack('>h',fc[18:20])+struct.unpack('>h',fc[22:24]))
    # print(fc.index(b'\xb6'),fc.index(b'\x34'))
# print(c[:100])
img = Image.new('RGB', (400,70), (255,255,255))
for i in pp:
    new = Image.new('RGB', (1,1), (0,0,0))
    img.paste(new,i)
plt.imshow(img)
plt.show()
```

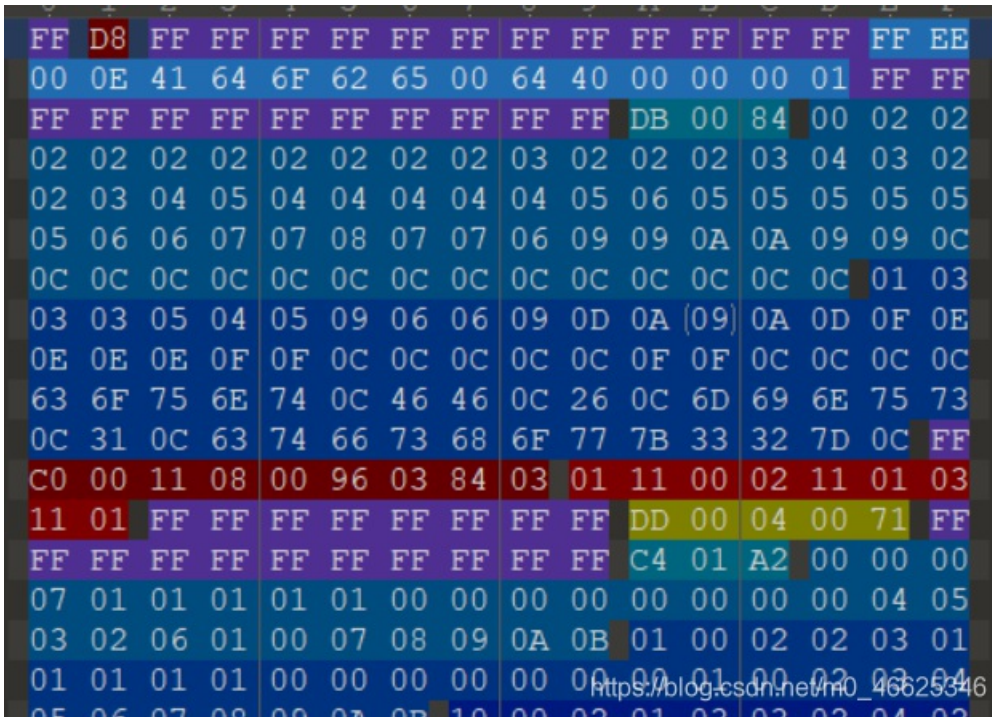
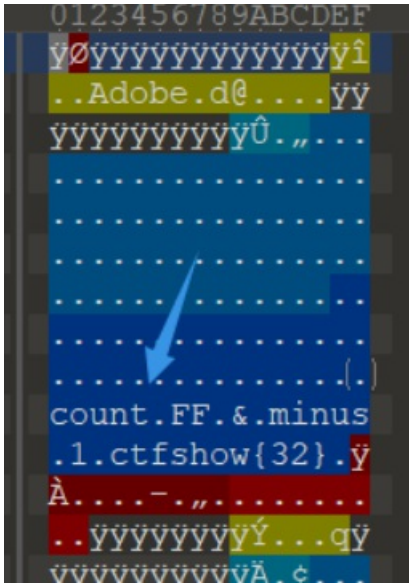


ctfshow{6d51f85b45a0061754a2776a32cf26c4}

misc48

用winhex打开，发现右侧文本信息有提示

- 1、统计FF的数量，再减去1
- 2、ctfshow{}中包含32个字符



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 07 10 13 13 9 09 13 0 13 6 0 10 9 2 1 0 1 10 8 11 5 12 7 2 2 3 10

用小脚本跑一下

```
s = '0 12 11 07 10 13 13 9 09 13 0 13 6 0 10 9 2 1 0 1 10 8 11 5 12 7 2 2 3 10'
d = '0123456789abcdef'
for i in s.split(' '):
    print(d[int(i)],end='')
```


0cb07add909d0d60a92101a8b5c7223a

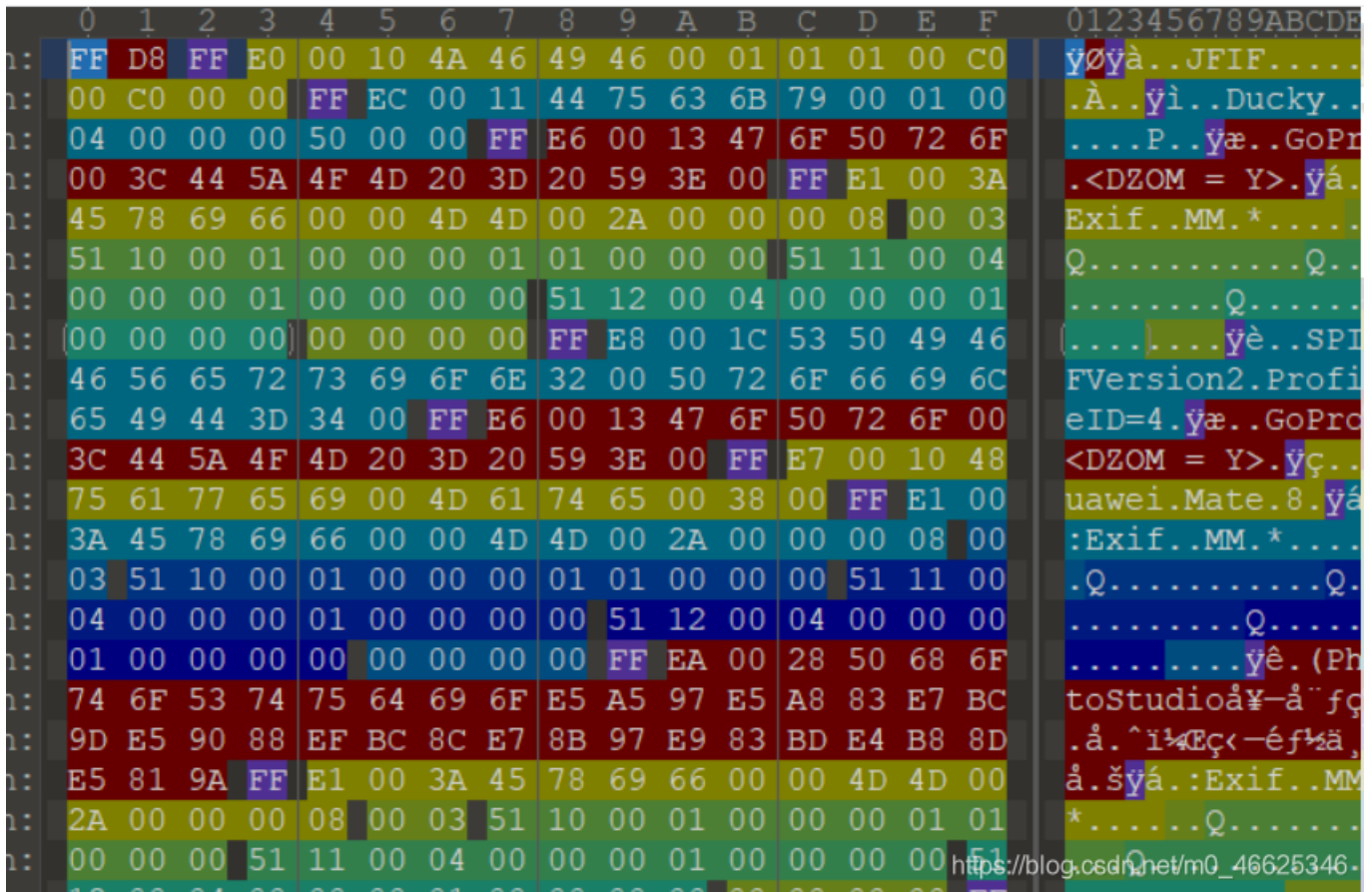
ctfshow{0cb07add909d0d60a92101a8b5c7223a}

misc49

提示：它们一来就是十六种。本题略脑洞，可跳过
用winhex打开，能看到很多字符串

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	huawei 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 https://blog.csdn.net/m0_46625346 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



把所有十六进制数保存在1.txt中，用一个小脚本处理一下
如果带有\x，在记事本中直接替换掉所有的即可。

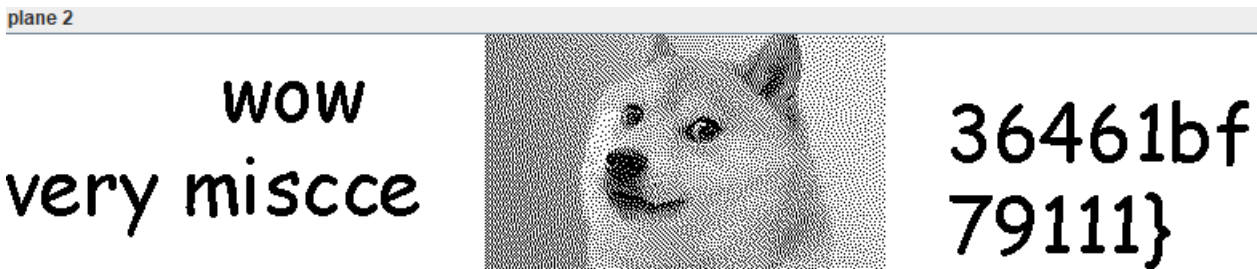
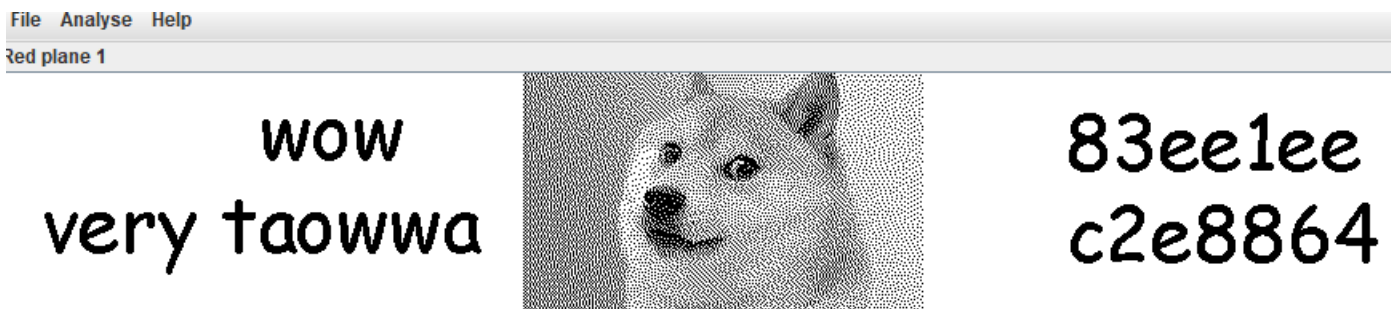
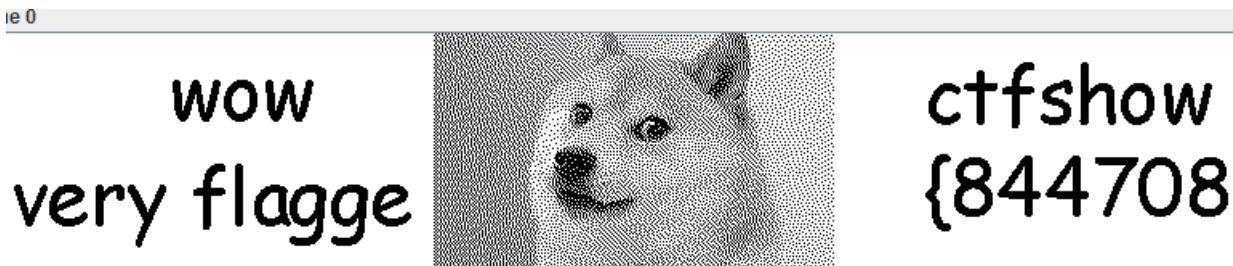

```
f=open("1.txt","r")
txt=f.read().replace("\n","")
f.close()

l=txt.split("FFE")
flag=""
for i in range(1,len(l)):
    flag += l[i][0]
print(flag.lower()[:32]) #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。
ctfshow{0c618671a153f5da3948fdb2a2238e44}

misc50

提示说：有时候视线也要放低一些。第一感觉用winhex修改图片高度，然而并没有用，多出来的部分一片黢黑。。。用stegsolve工具打开，不停的换色道，发现flag踪迹（狗头）



ctfshow{84470883ee1eec2e886436461bf79111}