

# DASCTF: BitMap

原创

末初 于 2021-10-14 23:01:12 发布 434 收藏 3

分类专栏: [CTF\\_MISC\\_Writeup](#) 文章标签: [CTF-bmp](#)

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/mochu7777777/article/details/120773359>

版权



[CTF\\_MISC\\_Writeup](#) 专栏收录该内容

246 篇文章 46 订阅

订阅专栏

来源于今天一位师傅给的题目附件, 从解出来的flag只知道是安恒的题目, 但是具体哪里的就不清楚了, 觉得题目还有记录一下的价值

BitMap 文件很明显是 bmp 图片修改了 BMP 文件头和 位图信息头, 不过一些还原信息的必要数据位置还是给了出来的, 比如 位深度

起首页 BitMap.bmp 未标题-1.bmp BitMap x

编辑方式: 十六进制(H) 运行脚本 运行模板

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789A B C D E F	
0000h:	42	4D	00	00	00	00	00	00	00	00	00	00	00	00	00	28	00	BM..... (.
0010h:	00	00	00	00	00	00	D4	FE	FF	FF	01	00	20	00	00	00	00	.....ôpyÿ. . .
0020h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0030h:	00	00	00	00	00	00	DF	CF	C5	00	FF	FF	FF	00	FF	FF	FF	.....βİÄ.ÿÿÿ.ÿÿ
0040h:	FF	00	FF	FF	FF	00	FF	FF	FF	00	FF	FF	FF	00	FF	FF	FF	ÿ.ÿÿÿ.ÿÿÿ.ÿÿÿ.ÿÿ
0050h:	FF	00	FF	FF	FF	00	FF	FF	FF	00	FF	FF	FF	00	FF	FF	FF	ÿ.ÿÿÿ.ÿÿÿ.ÿÿÿ.ÿÿ
0060h:	FF	00	FF	FF	FF	00	FF	FF	FF	00	FF	FF	FF	FF	FF	FF	FF	ÿ.ÿÿÿ.ÿÿÿ.ÿÿÿÿÿÿ
0070h:	FF	00	FF	FF	FF	00	FF	FF	FF	00	FF	FF	FF	FF	FF	FF	FF	ÿ.ÿÿÿ.ÿÿÿ.ÿÿÿÿÿÿ
0080h:	FF	00	FF	FF	FF	00	FF	FF	FF	00	FF	FF	FF	FF	FF	FF	FF	ÿ.ÿÿÿ.ÿÿÿ.ÿÿÿ.ÿÿ
0090h:	FF	00	FF	FF	FF	00	FF	FF	FF	00	FF	FF	FF	FF	FF	FF	FF	ÿ.ÿÿÿ.ÿÿÿ.ÿÿÿÿÿÿ
00A0h:	FF	00	FF	FF	FF	00	FF	FF	FF	00	FF	FF	FF	FF	FF	FF	FF	ÿ.ÿÿÿ.ÿÿÿ.ÿÿÿ.ÿÿ
00B0h:	FF	00	FF	FF	FF	00	FF	FF	FF	00	FF	FF	FF	FF	FF	FF	FF	ÿ.ÿÿÿÿÿÿÿÿÿÿÿÿ.ÿÿ
00C0h:	FF	00	FF	FF	FF	00	FF	FF	FF	00	FF	FF	FF	FF	FF	FF	FF	ÿ.ÿÿÿ.ÿÿÿÿÿÿÿÿÿÿÿÿ
00D0h:	FF	00	FF	FF	FF	00	FF	FF	FF	00	FF	FF	FF	FF	FF	FF	FF	ÿ.ÿÿÿ.ÿÿÿ.ÿÿÿ.ÿÿ
00E0h:	FF	00	FF	FF	FF	00	FF	FF	FF	00	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿ.ÿÿÿ.ÿÿÿ.ÿÿ
00F0h:	FF	00	FF	FF	FF	00	FF	FF	FF	00	FF	FF	FF	FF	FF	FF	FF	ÿ.ÿÿÿ.ÿÿÿÿÿÿÿÿÿÿÿÿ
0100h:	FF	00	FF	FF	FF	00	FF	FF	FF	00	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿ.ÿÿÿ.ÿÿÿ.ÿÿ
0110h:	FF	00	FF	FF	FF	00	FF	FF	FF	00	FF	FF	FF	FF	FF	FF	FF	ÿ.ÿÿÿ.ÿÿÿ.ÿÿÿ.ÿÿ
0120h:	FF	00	FF	FF	FF	00	FF	FF	FF	00	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿ.ÿÿÿÿÿÿÿÿÿÿÿÿ
0130h:	FF	00	FF	FF	FF	00	FF	FF	FF	00	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿ.ÿÿÿ.ÿÿÿ.ÿÿ
0140h:	FF	00	FF	FF	FF	00	FF	FF	FF	00	FF	FF	FF	FF	FF	FF	FF	ÿ.ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ.ÿÿ
0150h:	FF	00	FF	FF	FF	00	FF	FF	FF	00	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿ.ÿÿÿ.ÿÿÿ.ÿÿ
0160h:	FF	00	FF	FF	FF	00	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿ.ÿÿÿ.ÿÿÿÿÿÿÿÿÿÿÿÿ.ÿÿ
0170h:	FF	00	FF	FF	FF	00	FF	FF	FF	00	FF	FF	FF	FF	FF	FF	FF	ÿ.ÿÿÿ.ÿÿÿ.ÿÿÿÿÿÿÿÿÿÿÿÿ
0180h:	FF	FF	FF	FF	FF	00	FF	FF	FF	00	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿ.ÿÿÿ.ÿÿÿÿÿÿÿÿÿÿÿÿ
0190h:	FF	00	FF	FF	FF	00	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿ.ÿÿÿ.ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
01A0h:	FF	00	FF	FF	FF	00	FF	FF	FF	00	FF	FF	FF	FF	FF	FF	FF	ÿ.ÿÿÿ.ÿÿÿ.ÿÿÿ.ÿÿ

详细的还原过程可以参考我之前给白帽子社区端午节活动的出的题目: [白帽子社区端午节活动-白帽寻宝记-纪念屈原Writeup](#)

这里可以添加一个后缀 `.bmp`，然后使用 `010 Editor` 打开；借助 `010 Editor` 强大的插件分析功能，还是能显示出文件基本结构可以看到位深度已经给了出来：`32 bit`

名称	值	开始	大小	颜色
struct BITMAPFILEHEADER bmh		0h	14h	Fe: Bg: [ ]
> CHAR bfType[2]	BM	0h	2h	Fe: Bg: [ ]
DWORD bfSize	0	2h	4h	Fe: Bg: [ ]
WORD bfReserved1	0	6h	2h	Fe: Bg: [ ]
WORD bfReserved2	0	8h	2h	Fe: Bg: [ ]
DWORD bfOffBits	0	Ah	4h	Fe: Bg: [ ]
struct BITMAPINFOHEADER bmh		Eh	28h	Fe: Bg: [ ]
DWORD biSize	40	Eh	4h	Fe: Bg: [ ]
LONG biWidth	0	12h	4h	Fe: Bg: [ ]
LONG biHeight	-300	16h	4h	Fe: Bg: [ ]
WORD biPlanes	1	1Ah	2h	Fe: Bg: [ ]
WORD biBitCount	32	1Ch	2h	Fe: Bg: [ ]
DWORD biCompression	0	1Eh	4h	Fe: Bg: [ ]
DWORD biSizeImage	0	22h	4h	Fe: Bg: [ ]
LONG biXPelsPerMeter	0	26h	4h	Fe: Bg: [ ]
LONG biYPelsPerMeter	0	2Ah	4h	Fe: Bg: [ ]
DWORD biClrUsed	0	2Eh	4h	Fe: Bg: [ ]
DWORD biClrImportant	0	32h	4h	Fe: Bg: [ ]
> struct BITMAPLINE lines[300]		36h	0h	Fe: Bg: [ ]

CSDN @末初

`BMP文件头` 和 `位图信息头` 的一些信息段直接填即可；需要稍微注意的是这里的文件大小应该为：`1080056 Bytes`

因为这里还有 `文件尾(共2bytes: 00 00)` 没有加；然后根据计算公式

$\text{分辨率}(\text{width} * \text{height}) \times (\text{颜色深度}/8) + \text{bmp文件头}(\text{共}14\text{Bytes}) + \text{位图信息头}(\text{共}40\text{Bytes}) + \text{文件尾}(\text{共}2\text{bytes: } 00\ 00)$   
= 图像文件大小

可以得到

$\text{分辨率}(\text{width} * \text{height}) = 270000$

利用脚本分解一下，看看有哪些解

```
for width in range(1,1000):
    for height in range(1,1000):
        if width * height == 270000:
            print("{} * {} = {}".format(width, height,width*height))
            print("width: {}({})\nheight: {}({})\n".format(width, hex(width)[2:].upper(), height, hex(height)[2:].upper()))
```

```

300 * 900 = 270000
width: 300(12C)
height: 900(384)

360 * 750 = 270000
width: 360(168)
height: 750(2EE)

375 * 720 = 270000
width: 375(177)
height: 720(2D0)

400 * 675 = 270000
width: 400(190)
height: 675(2A3)

432 * 625 = 270000
width: 432(1B0)
height: 625(271)

450 * 600 = 270000
width: 450(1C2)
height: 600(258)

500 * 540 = 270000
width: 500(1F4)
height: 540(21C)

```

位图信息头中给了一个 -300，其实也是在提示图片的正确高度为 300，那么宽就是 900

- DWORD bsize: 1080056
- DWORD bfOffBits: 54
- LONG biWidth: 900
- LONG biHeight: 300

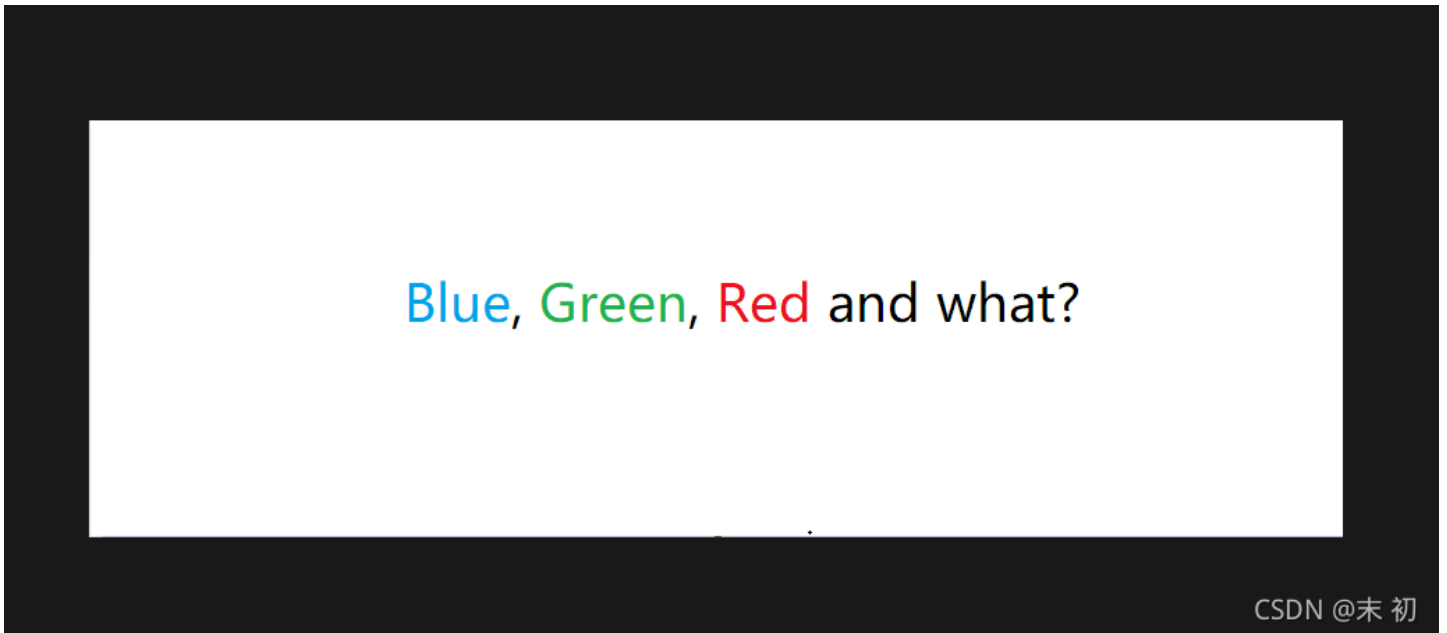
补上这些信息即可正常显示了

名称	值
struct BITMAPFILEHEADER bmfh	
CHAR bftype[2]	BM
DWORD bSize	1080056
WORD bfReserved1	0
WORD bfReserved2	0

WORD biReserved2	0
DWORD biOffsetBits	54
▼ struct BITMAPINFOHEADER bmih	
DWORD biSize	40
LONG biWidth	900
LONG biHeight	300
WORD biPlanes	1
WORD biBitCount	32
DWORD biCompression	0
DWORD biSizeImage	0
LONG biXPelsPerMeter	0
LONG biYPelsPerMeter	0
DWORD biClrUsed	0
DWORD biClrImportant	0
> struct BITMAPLINE lines[300]	

CSDN @末初

得到的图片用 PS 水平翻转之后再来个一百八十度反转



一开始以为指的是 alpha 通道，可是 bmp 图片好像也没有 alpha 通道

```

PS C:\Users\Administrator\Downloads> python .\code.py
('R', 'G', 'B')
RGB
BMP
PS C:\Users\Administrator\Downloads>

```

```

C:\Users\Administrator\Downloads\code.py - Sublime Text (UNREGIST
File Edit Selection Find View Goto Tools Project Preferences
code.py data.py qr.py
1 from PIL import Image
2
3 img = Image.open("BitMap.bmp")
4
5 print(img.getbands())
6 print(img.mode)
7 print(img.format)
8

```

CSDN @末初

继续利用 010 Editor 分析文件发现 struct BITMAPLINE lines[300]->struct BITMAPLINE lines[0]->struct RGBQUAD colors[900]->struct RGBQUAD colors[0] 下的 UBYTE rgbReserved 位有些为 00 有些为 FF；这个位置默认都是为 00

名称	值	开始	大小	颜色
struct BITMAPLINE lines[300]		36h	107AC0h	Fg: Bg:
struct BITMAPLINE lines[0]		36h	E10h	Fg: Bg:
struct RGBQUAD colors[900]		36h	E10h	Fg: Bg:
struct RGBQUAD colors[0]	#00C5CFDF	36h	4h	Fg: Bg:
UBYTE rgbBlue	223	36h	1h	Fg: Bg:
UBYTE rgbGreen	207	37h	1h	Fg: Bg:
UBYTE rgbRed	197	38h	1h	Fg: Bg:
UBYTE rgbReserved	0	39h	1h	Fg: Bg:
> struct RGBQUAD colors[1]	#00FFFFFF	3Ah	4h	Fg: Bg:
struct RGBQUAD colors[2]	#00FFFFFF	3Bh	4h	Fg: Bg:
UBYTE rgbBlue	255	3Bh	1h	Fg: Bg:
UBYTE rgbGreen	255	3Ch	1h	Fg: Bg:
UBYTE rgbRed	255	3Dh	1h	Fg: Bg:
UBYTE rgbReserved	0	3Eh	1h	Fg: Bg:
> struct RGBQUAD colors[3]	#00FFFFFF	42h	4h	Fg: Bg:
> struct RGBQUAD colors[4]	#00FFFFFF	46h	4h	Fg: Bg:
> struct RGBQUAD colors[5]	#00FFFFFF	4Ah	4h	Fg: Bg:
> struct RGBQUAD colors[6]	#00FFFFFF	4Eh	4h	Fg: Bg:
UBYTE rgbBlue	255	4Eh	1h	Fg: Bg:
UBYTE rgbGreen	255	4Fh	1h	Fg: Bg:
UBYTE rgbRed	255	50h	1h	Fg: Bg:
UBYTE rgbReserved	0	51h	1h	Fg: Bg:
> struct RGBQUAD colors[7]	#00FFFFFF	52h	4h	Fg: Bg:
> struct RGBQUAD colors[8]	#00FFFFFF	56h	4h	Fg: Bg:
> struct RGBQUAD colors[9]	#00FFFFFF	5Ah	4h	Fg: Bg:
> struct RGBQUAD colors[10]	#00FFFFFF	5Eh	4h	Fg: Bg:
> struct RGBQUAD colors[11]	#00FFFFFF	62h	4h	Fg: Bg:
> struct RGBQUAD colors[12]	#00FFFFFF	66h	4h	Fg: Bg:
> struct RGBQUAD colors[13]	#FFFFFF	6Ah	4h	Fg: Bg:
UBYTE rgbBlue	255	6Ah	1h	Fg: Bg:
UBYTE rgbGreen	255	6Bh	1h	Fg: Bg:
UBYTE rgbRed	255	6Ch	1h	Fg: Bg:
UBYTE rgbReserved	255	6Dh	1h	Fg: Bg:
> struct RGBQUAD colors[14]	#00FFFFFF	6Eh	4h	Fg: Bg:
> struct RGBQUAD colors[15]	#00FFFFFF	72h	4h	Fg: Bg:
> struct RGBQUAD colors[16]	#00FFFFFF	76h	4h	Fg: Bg:
> struct RGBQUAD colors[17]	#FFFFFF	7Ah	4h	Fg: Bg:
UBYTE rgbBlue	255	7Ah	1h	Fg: Bg:
UBYTE rgbGreen	255	7Bh	1h	Fg: Bg:
UBYTE rgbRed	255	7Ch	1h	Fg: Bg:
UBYTE rgbReserved	255	7Dh	1h	Fg: Bg:
> struct RGBQUAD colors[18]	#00FFFFFF	7Eh	4h	Fg: Bg:
> struct RGBQUAD colors[19]	#00FFFFFF	82h	4h	Fg: Bg:
> struct RGBQUAD colors[20]	#00FFFFFF	86h	4h	Fg: Bg:
> struct RGBQUAD colors[21]	#00FFFFFF	8Ah	4h	Fg: Bg:
> struct RGBQUAD colors[22]	#00FFFFFF	8Eh	4h	Fg: Bg:

选定: 1 个字节 (范围: 109 [6Dh] 到 109 [6Dh])

CSDN @末初

明显修改了数据，那我们就将这些位置的数据提取出来；只有 **00** 和 **FF** 两种情况，常见的要么是二进制数据转字符，要么就是二进制数据画图(黑白图，例如二维码)

```
from binascii import *

data_list = []
with open("BitMap.bmp", 'rb') as f:
    seek = 0x36
    f.seek(seek)
    for n in range(270000):
        data_list.append(hexlify(f.read(4)).upper())
        seek += 4
    f.seek(seek)

bin_data = ''
for data in data_list:
    data = data.decode()[6:]
    if data == '00':
        bin_data += '0'
    elif data == 'FF':
        bin_data += '1'
    else:
        print(data)
print(bin_data)
with open('data.txt', 'w') as f1:
    f1.write(bin_data)
```

二进制数据转字符发现都是乱码，没有什么线索；尝试画图，先确定图片宽高；数据总长度为 **270000**，上面也跑过，可以直接先试一试 **300\*900**

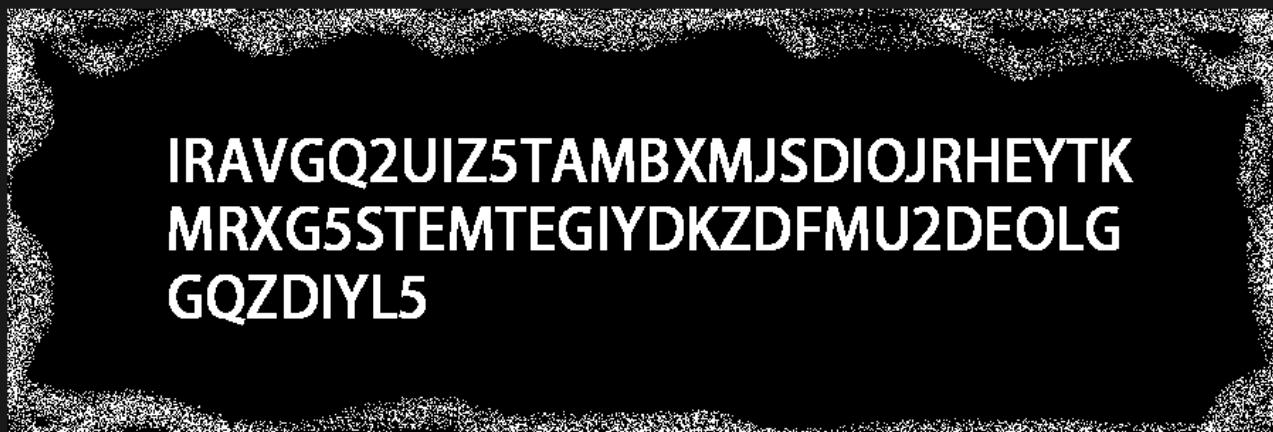
```

from PIL import Image

with open("data.txt", 'r') as f:
    data = f.read()
    width = 300
    height = 900
    idx = 0
    img = Image.new("RGB", (width, height))
    for w in range(width):
        for h in range(height):
            if data[idx] == '0':
                img.putpixel((w, h), (0, 0, 0))
            elif data[idx] == '1':
                img.putpixel((w, h), (255, 255, 255))
            else:
                print(data[idx])
                break
            idx += 1
    img.save('flag.png')
    img.show()

```

得到的图片 **PS** 水平翻转，然后180度旋转



CSDN @末初

```

>>> from base64 import *
>>> b32decode("IRAVGQ2UIZ5TAMBXMJSDIOJRHEYTKMRXG5STEMTEGIYDKZDFMU2DEOLGGQZDIYL5")
b'DASCTF{007bd491915277e22d205dee429f424a}'

```