DASCTF： BitMap

原创

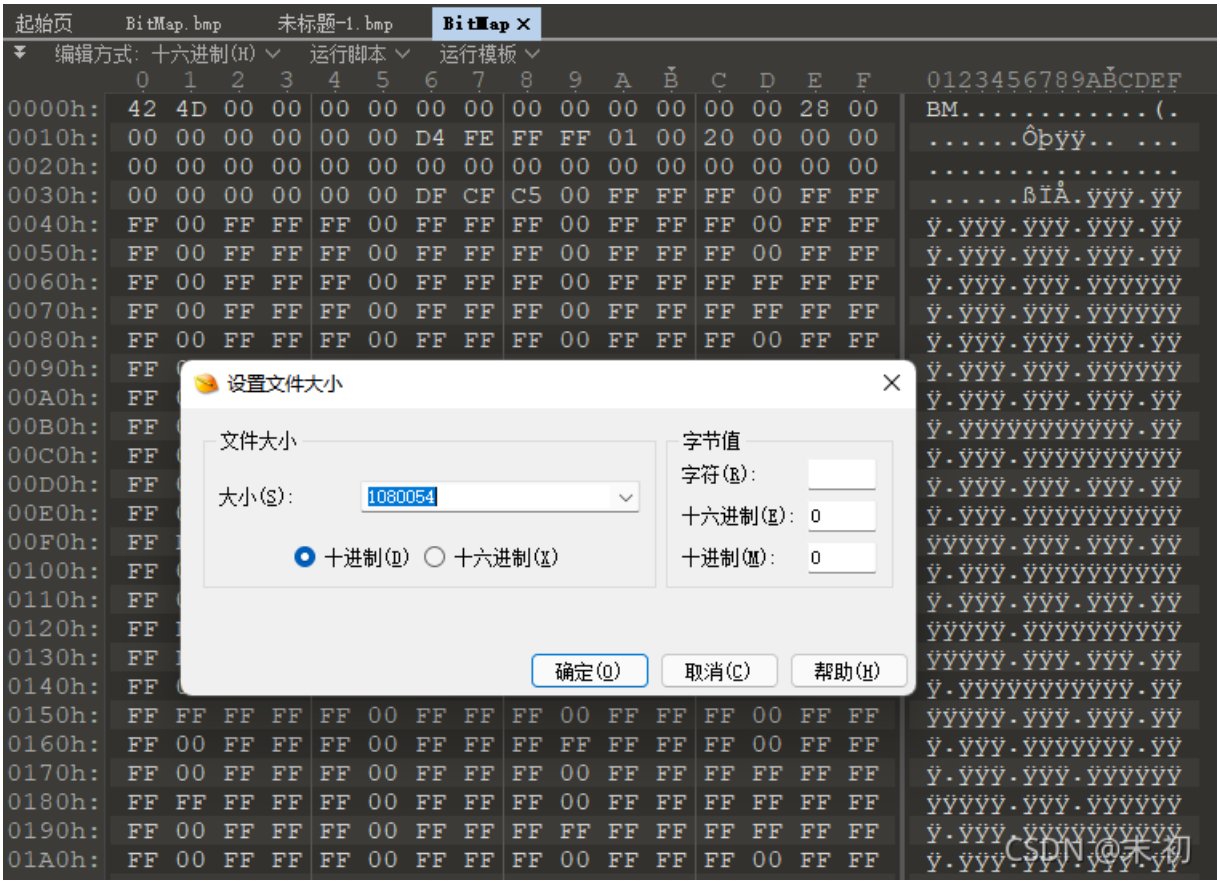
末 初 于 2021-10-14 23:01:12 发布 434 收藏 3

分类专栏： CTF_MISC_Writeup 文章标签： CTF-bmp

本文链接：https://blog.csdn.net/mochu7777777/article/details/120773359

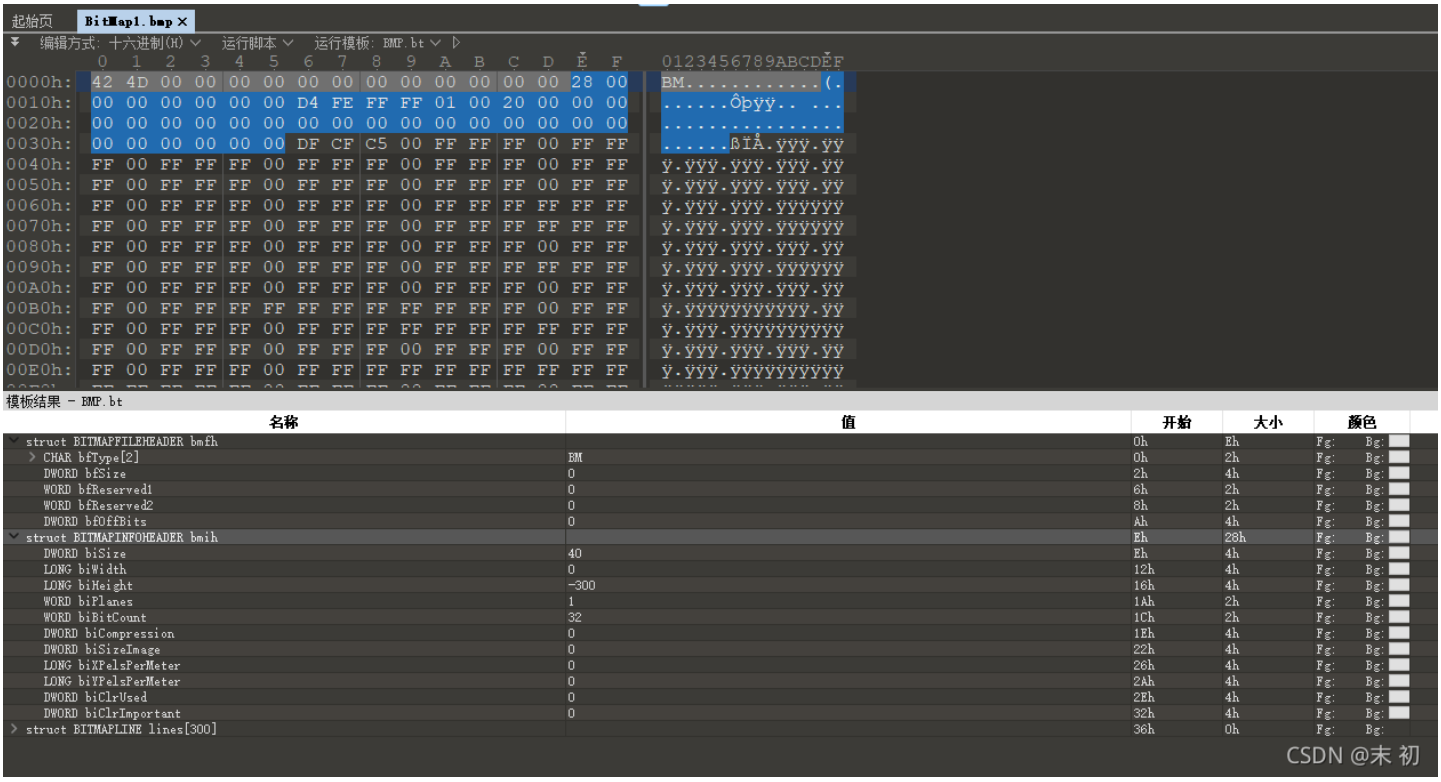
> 来源于今天一位师傅给的题目附件，从解出来的flag只知道是安恒的题目，但是具体哪里的就不清楚了，觉得题目还有记录一下的价值

`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`



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`



`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是为：`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

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

可以得到

```
分辨率(width * height) = 270000
```

利用脚本分解一下，看看有哪些解

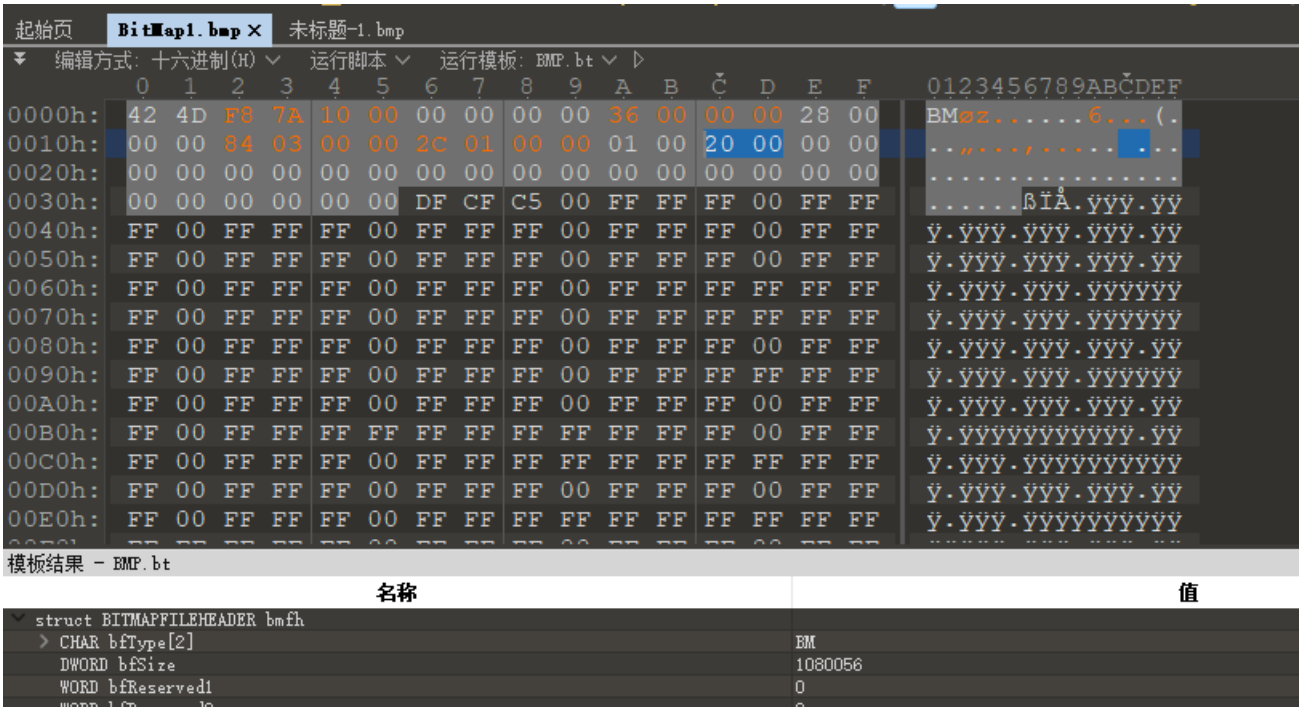
```
for width in range(1,1000):
    for height in range(1,1000):
        if width * height == 270000:
            print("{} * {} = {}".format(width, height,width*height))
            print("width: {}({})\nheight: {}({})\n".format(width, hex(width)[2:].upper(), height, hex(height)[2:
].upper()))
```

```
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 bfsize: `1080056`
- DWORD bfOffBits: `54`
- LONG biWidth: `900`
- LONG biHeight: `300`

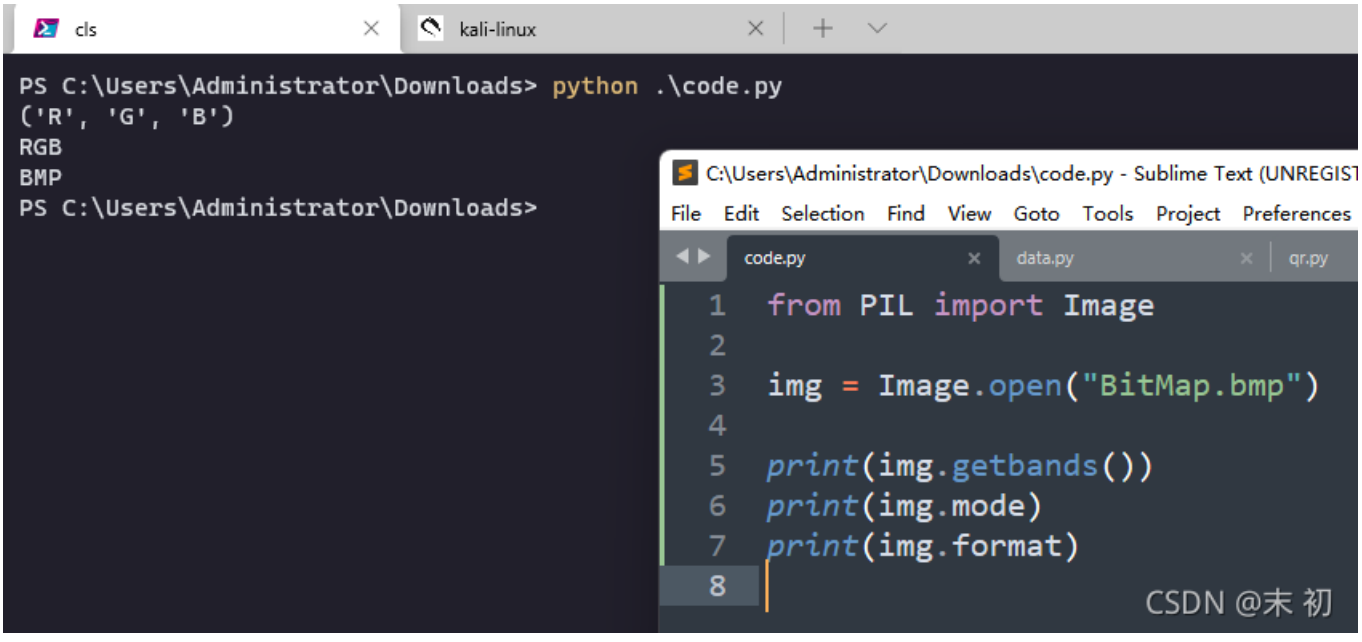
补上这些信息即可正常显示了



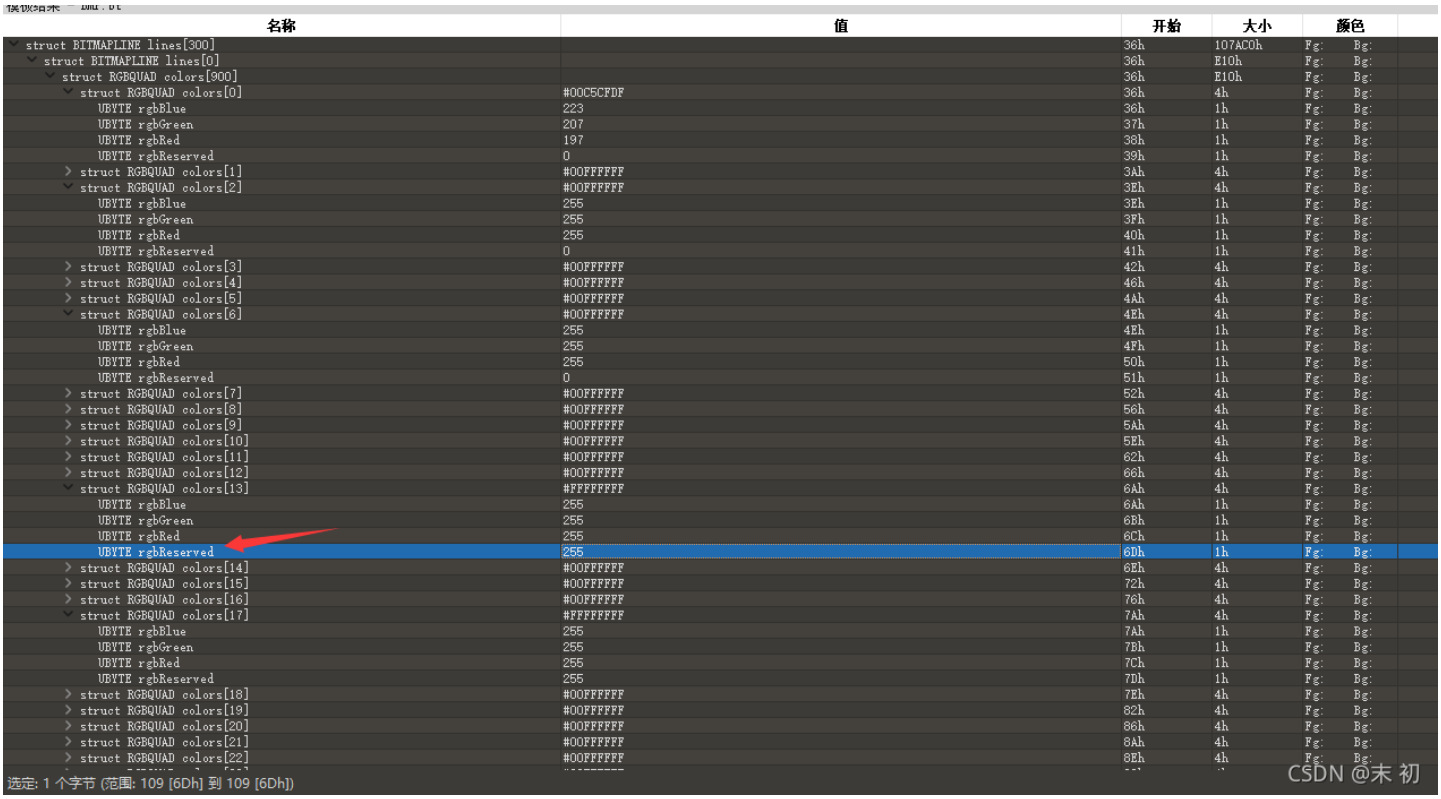
WORD bfReserved2	0
DWORD bfOffBits	54
struct BITMAPINFOHEADER bmih	
DWORD biSize	40
LONG biWidth	900
LONG biHeight	300
WORD biPlanes	1
WORD biBitCount	32
DWORD biCompression	0
DWORD biSizeImage	0
LONG biXPelsPerMeter	0
LONG biYPelsPerMeter	0
DWORD biClrUsed	0
DWORD biClrImportant	0
struct BITMAPLINE lines[300]	

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



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



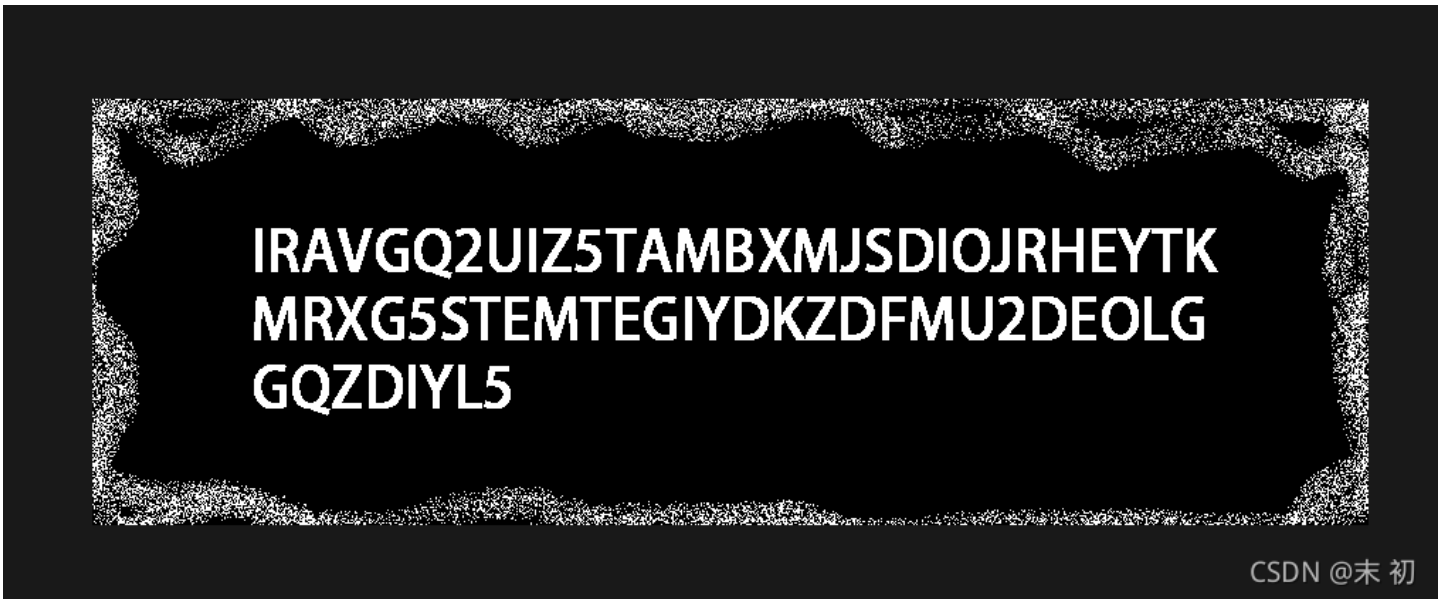
明显修改了数据，那我们就将这些位置的数据提取出来；只有`00`和`FF`两种情况，常见的要么是二进制数据转字符，要么就是二进制数据画图(黑白图，例如二维码)

```
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




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