**Dissolved**

Dissolved... Image... What does that even mean?!

stego.png

For steganography challenges, a good first step is to use [Stegsolve](https://github.com/zardus/ctf-tools) to find any odd patterns in the image. By carefully looking at each bitplane, we'll notice there are a few different pixels in bits 0 and 2 of the alpha channel.

The hypothesis is that the flag data is hidden in these pixels. To make it easier to manipulate and experiment, let's get those pixels in Python:

**from** scipy **import** misc

arr = misc.imread('stego.png')

pixels = []

**for** i **in** range(0, len(arr)):

**for** j **in** range(0, len(arr[0])):

**if** arr[i, j, 3] != 255:

pixels.append(arr[i, j])

The image is read as a multidimensional array, where arr[5, 3, 0] represents the red ('RGBA'[0]) value at position (3, 5). We copied to the pixels list every pixel where the alpha channel is not 255 (i.e. the weird pixels we found).

Let's analyze what we have: len(pixels) returns 312, which is divisible. by 8. Also, 312÷8 = 39, which is a reasonable size for a flag. Thus, each pixel might be hiding 1 bit of data, probably in the least significant bit (LSB). We don't know which channel contains the data, so let's get the LSB of each channel separately and see what we get:

r\_bitstream = ''

g\_bitstream = ''

b\_bitstream = ''

**for** x **in** pixels:

r\_bitstream += str(x[0] & 1)

g\_bitstream += str(x[1] & 1)

b\_bitstream += str(x[2] & 1)

print('Red: ' + ''.join(chr(int(r\_bitstream[i\*8:i\*8+8],2)) **for** i **in** range(len(r\_bitstream)//8)))

print('Green: ' + ''.join(chr(int(g\_bitstream[i\*8:i\*8+8],2)) **for** i **in** range(len(g\_bitstream)//8)))

print('Blue: ' + ''.join(chr(int(b\_bitstream[i\*8:i\*8+8],2)) **for** i **in** range(len(b\_bitstream)//8)))

And here's the result:

Red: JP±j£Ò

¹2íÞepF¼

NmüåÛ'÷ÏQÂ·ðA

Green: ?-Â`zHý5Ë¡Òø3òyE?&z¸%Ï\*ÅÁÊ9QxØâ]PØßA

Blue: VolgaCTF{Tr@nspar3ncy\_g1ves\_fLag\_aw@y!}

Flag: VolgaCTF{Tr@nspar3ncy\_g1ves\_fLag\_aw@y!}

# 溶解

已解散...图片...这甚至意味着什么？

stego.png

对于隐写术挑战，一个很好的第一步是使用[Stegsolve](https://github.com/zardus/ctf-tools)查找图像中的任何奇怪图案。通过仔细查看每个位平面，我们会注意到alpha通道的位0和2中有一些不同的像素。

假设是标志数据隐藏在这些像素中。为了更易于操作和实验，让我们在Python中获取这些像素：

**from** scipy **import** misc

arr = misc.imread('stego.png')

pixels = []

**for** i **in** range(0, len(arr)):

**for** j **in** range(0, len(arr[0])):

**if** arr[i, j, 3] != 255:

pixels.append(arr[i, j])

图像被读取为多维数组，其中arr[5, 3, 0]代表'RGBA'[0]位置（3，5）的红色（）值。我们将pixelsalpha通道不为255的每个像素（即，我们发现的怪异像素）复制到列表中。

让我们分析一下我们所拥有的：len(pixels)返回312，这是可分割的。乘以8。此外，312÷8 = 39，这是一个标志的合理大小。因此，每个像素可能隐藏1位数据，可能在最低有效位（LSB）中。我们不知道哪个通道包含数据，因此让我们分别获取每个通道的LSB并查看得到的结果：

r\_bitstream = ''

g\_bitstream = ''

b\_bitstream = ''

**for** x **in** pixels:

r\_bitstream += str(x[0] & 1)

g\_bitstream += str(x[1] & 1)

b\_bitstream += str(x[2] & 1)

print('Red: ' + ''.join(chr(int(r\_bitstream[i\*8:i\*8+8],2)) **for** i **in** range(len(r\_bitstream)//8)))

print('Green: ' + ''.join(chr(int(g\_bitstream[i\*8:i\*8+8],2)) **for** i **in** range(len(g\_bitstream)//8)))

print('Blue: ' + ''.join(chr(int(b\_bitstream[i\*8:i\*8+8],2)) **for** i **in** range(len(b\_bitstream)//8)))

结果如下：

Red: JP±j£Ò

¹2íÞepF¼

NmüåÛ'÷ÏQÂ·ðA

Green: ?-Â`zHý5Ë¡Òø3òyE?&z¸%Ï\*ÅÁÊ9QxØâ]PØßA

Blue: VolgaCTF{Tr@nspar3ncy\_g1ves\_fLag\_aw@y!}

旗： VolgaCTF{Tr@nspar3ncy\_g1ves\_fLag\_aw@y!}