colorful code

发表于 2021-05-25 更新于 2021-05-26 分类于 Challenge， 2021， 第四届红帽杯网络安全大赛， Misc
Challenge | 2021 | 第四届红帽杯网络安全大赛 | Misc | colorful code

点击此处获得更好的阅读体验

WriteUp来源

来自Bellin的博客

题目描述

> *colorful code is beautiful, isn't it? 最终得到flag的格式中加上flag{}包裹再提交*

解题思路

MISC的题大都是脑洞题。这个题只提示了colorful code，一开始我以为只是说的是RGB的颜色代码如#FFFFC0等，但是思路也基本正确。给了data1和data2两个文件。观察两个文件如下。

第一个文件全是数字与空格，拿python统计一下数字范围在0~19内，再看第二个文件，后面都是三个相同的字母连起来，其十六进制值从0x14到0xFF，这些信息都是没用的，那么大概率只有前面的60字节是有用的信息，每三个一组作为颜色代码的RGB值，刚好20个和data1的数字范围对应。那么data作为像素值代入进行图像的矩阵构建，应该就是正确的思路了。

还有一个问题是我们不知道图像的宽和高。对data1内像素的数量进行简单计算：

```
data = open("data1", "r").read().split(" ")
data = list(filter(None, data))
print(len(data))
# 7076=37*191
```

*那么图像的宽高应该是37*191或者反过来。*

转换图像的代码如下：

```
from PIL import Image
color = [
    (0x00, 0x00, 0x00), (0x00, 0x00, 0xc0), (0x00, 0xff, 0xff), (0x00, 0xff, 0x00),
    (0xff, 0xc0, 0xff), (0xff, 0xc0, 0xc0), (0xc0, 0xc0, 0xff), (0xc0, 0xc0, 0x00),
    (0xff, 0x00, 0xff), (0xff, 0x00, 0x00), (0xc0, 0x00, 0x00), (0xc0, 0x00, 0xc0),
    (0xff, 0xff, 0xff), (0xff, 0xff, 0x00), (0xff, 0xff, 0xc0), (0x00, 0xc0, 0x00),
    (0x00, 0xc0, 0xc0), (0xc0, 0xff, 0xff), (0xc0, 0xff, 0xc0), (0x00, 0x00, 0xff)
]
data=open('data1',"r").read().split(" ")[:-1]
img = Image.new("RGB",(37,191))
for i in range(37):
    for j in range(191):
        img.putpixel((i,j),color[int(data[i*191+j])])
img.save('flag.png')
```

```
from PIL import Image
im = Image.new("RGB", (37,191))
# im = Image.new("RGB", (191,37))

infile = open('data1', 'r')
hide = [
    '00', '00', '00', '00',
    '00', 'C0', '00', 'FF',
    'FF', '00', 'FF', '00',
    'FF', 'C0', 'FF', 'FF',
    'C0', 'C0', 'C0', 'C0',
    'FF', 'C0', 'C0', '00',
    'FF', '00', 'FF', 'FF',
    '00', '00', 'C0', '00',
    '00', 'C0', '00', 'C0',
    'FF', 'FF', 'FF', 'FF',
    'FF', '00', 'FF', 'FF',
    'C0', '00', 'C0', '00',
    '00', 'C0', 'C0', 'C0',
    'FF', 'FF', 'C0', 'FF',
    'C0', '00', '00', 'FF'
]
data = infile.read().strip('\n').split(" ")
data = list(filter(None, data))

key_set = []
for i in range(20):
    print(data.count(str(i)))
for i in range(len(hide)//3):
    key = hide[3*i]+hide[3*i+1]+hide[3*i+2]
    key_set.append(key)
print(key_set)

rgblist = []

for x in data:
    v = key_set[int(x)]
    rgb = (int(v[:2],16),int(v[2:4],16),int(v[4:],16))
    rgblist.append(rgb)
im.putdata(rgblist)
im.save("flag.png")
```

这里给了两个脚本，想说明一下这两个脚本解出来的图像是不一样的，因为python的`putdata`或者用`reshape`来自动化重构像素矩阵，它默认的像素序列的填充方式是先横轴再纵轴，而这个题像素重构的正确方向是先纵轴再横轴，所以后一个脚本得到的图像拿不到正确的flag。

后面就是真正的colorful code所代表的的编程语言PIET，然后这里mark一下一些脑洞大开的编程语言，可能有一些奇奇怪怪的隐写题会用到。

最后把得到的图像放到在线编译网站即可得到flag。（或者可以去github下载python对应的编译包）

Flag

```
flag{88842f20-fb8c-45c9-ae8f-36135b6a0f11}
```


Challenge # 2021 # Misc # 第四届红帽杯网络安全大赛

primegame

签到