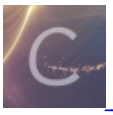
简单的LSB图片隐写

原创

LDAx 于 2021-05-10 23:55:11 发布 504 收藏 3

分类专栏： misc 水题日常

本文链接：https://blog.csdn.net/qq_31028197/article/details/116615455


misc 同时被 2 个专栏收录

2 篇文章 0 订阅

订阅专栏


水题日常

3 篇文章 0 订阅

订阅专栏

一个小作业，但坑却意外的多，记一下踩的坑

jpg会自动进行压缩操作

开始用jpg尝试写入数据，但是每次都会出现乱码，调试半天猛然想起jpg有自动压缩

测试代码

```
import cv2
import os

img = cv2.imread('b.jpg')
for row in range(0,3):
        for col in range(0,3):
                img[row, col, 0] = 234
                print(img[row,col])
cv2.imwrite("c.jpg",img)

print()

img = cv2.imread("c.jpg")
for row in range(0,3):
        for col in range(0,3):
                print(img[row,col])

输出：
[234   0   0]
[234   0   0]
[234   0   0]
[234   0   0]
[234   0   0]
[234   0   0]
[234   0   0]
[234   0   0]
[234   0   0]

[227   0   1]
[205   5   6]
[154  12  11]
[204   3   6]
[182   8   8]
[136  13  11]
[153  11  10]
[137  14  12]
[104  15  11]
```

之后查了一下

> JPEG图像格式使用离散余弦变换（Discrete Cosine Transform，DCT）函数来压缩图像，而这个图像压缩方法的核心是：通过识别每个8×8像素块中相邻像素的重复像素来减少显示图像所需的位数，并使用近似估算法降低其冗余度。可以使用JSteg、JPHide、Outguess、F5等等隐写方式

有时间再学这个。。

补零和分割符

因为选用了LSB的隐写方式，就需要把信息二进制化后储存在最末位
二进制转换代码

```
"".join([bin(ord(c)).replace('0b', '').zfill(8) for c in flag])
```

开始没注意python的二进制转换是舍去高位0的，比如49(10) => 110001(2)，为了方便读取需要在前面补上00让其为8位，这样就可以直接读取（然后每个都填了个00结果有5位和7位的，再次乱码），用zfill方法可以自动用0补齐8位

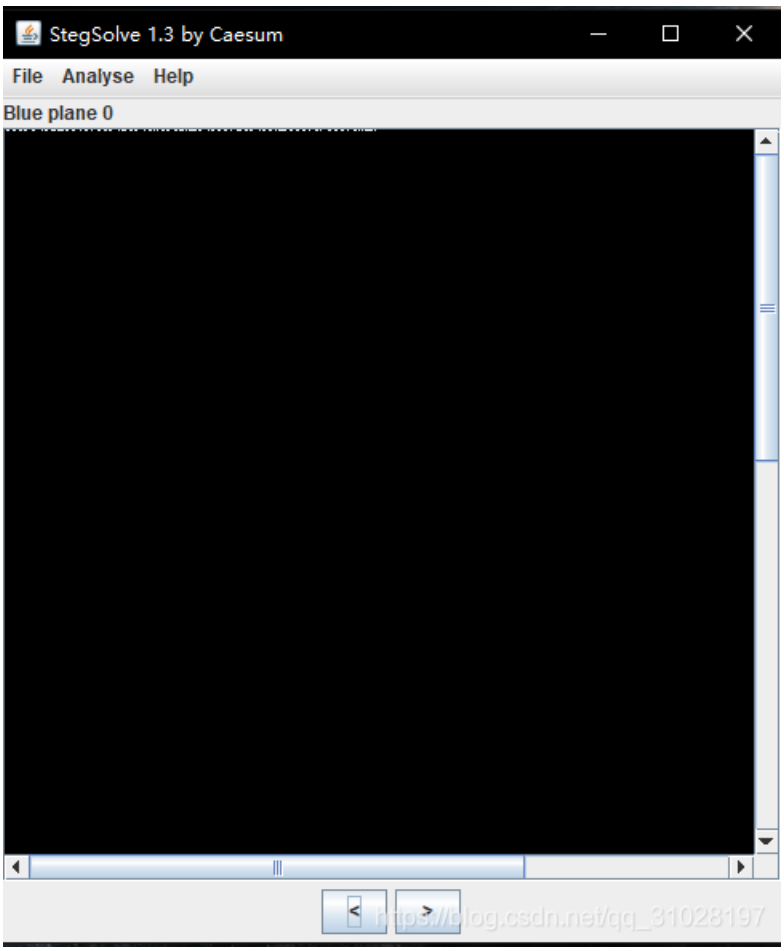
代码

然后这是无比凌乱的代码

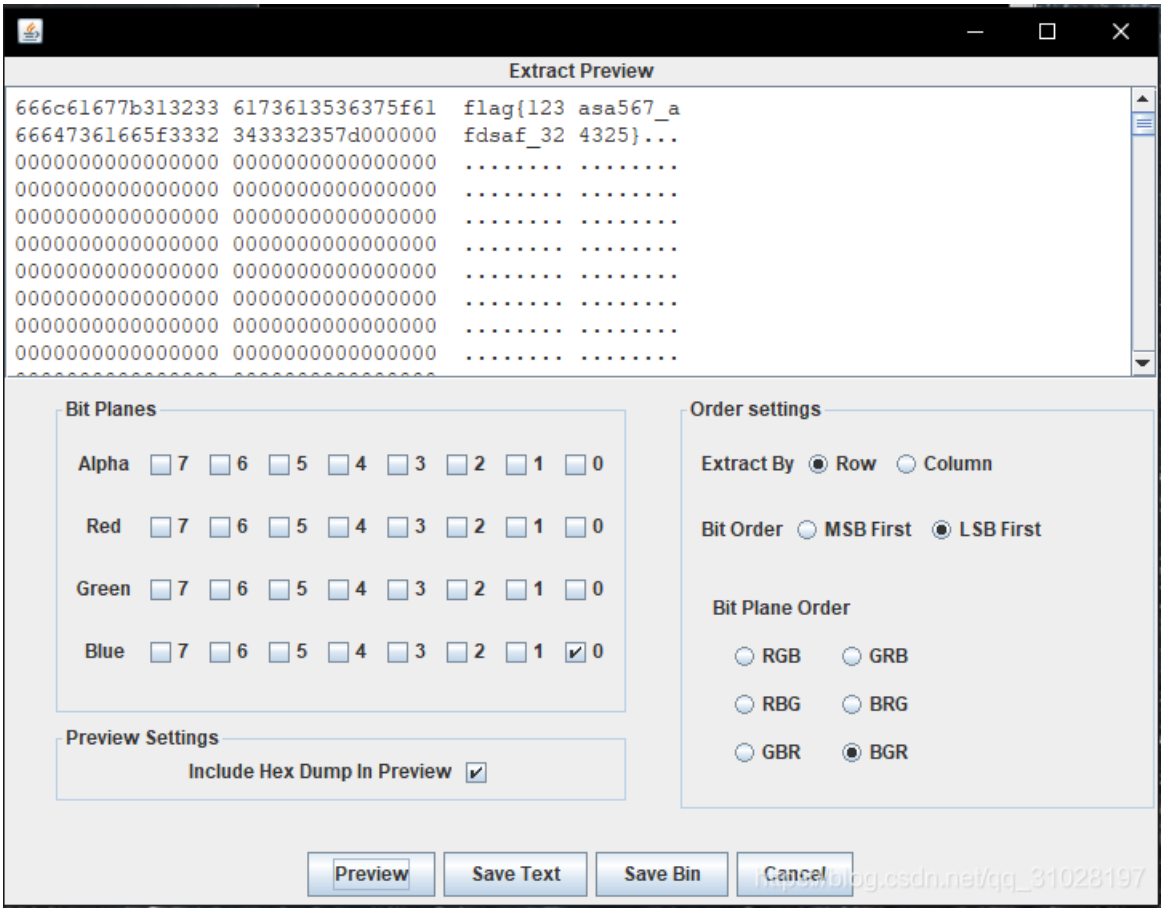
```
import cv2
import os

flag = 'flag{123asa567_afdsaf_324325}'
bits = "".join([bin(ord(c)).replace('0b', '').zfill(8) for c in flag])

bAddress = 'b.jpg'
img = cv2.imread(bAddress)
height = img.shape[0]
weight = img.shape[1]
channels = img.shape[2]

def set_bit_val(byte, index, val):
    """
    更改某个字节中某一位（Bit）的值

    :param byte: 准备更改的字节原值
    :param index: 待更改位的序号，从右向左0开始，0-7为一个完整字节的8个位
    :param val: 目标位预更改的值，0或1
    :returns: 返回更改后字节的值
    """
    if val == '1':
        return byte | (1 << index)
    else:
        return byte & ~(1 << index)


def photoBGR():
    '''
    把数据放在BGR三信道上
    '''
    i = 0
    for row in range(height):
        for col in range(weight):
            for c in range(channels):
                if i < len(bits):
                    value = img[row, col, c]
                    img[row, col, c] = set_bit_val(value, 0, bits[i])
                    i += 1
                else:
                    cv2.imwrite("c.png",img)
                    return

def photoB():
    '''
    单B信道
    '''
    i = 0
    for row in range(height):
        for col in range(weight):
                if i < len(bits):
                    value = img[row, col, 0]
                    img[row, col, 0] = set_bit_val(value, 0, bits[i])
                    i += 1
                else:
                    cv2.imwrite("c.png",img)
                    return
```

```
photoB()
```

使用Stegsolve进行一下检验



这是一张纯黑色的图片，在切换到Blue plane 0的时候可以发现左上角有一些白色小点



进入二进制分析，选择Blue的最低位，可以看到我们写入的数据