Python实现LSB隐写算法

转载

「已注销」 于 2018-09-06 22:42:08 发布 19243 收藏 122
分类专栏： ctf


ctf 专栏收录该内容
21 篇文章 0 订阅
订阅专栏

目录

预备知识

LSB算法

在二进制数中意为**最低有效位**，一般来说，MSB(最高有效位)位于二进制数的最左侧，**LSB位于二进制数的最右侧**。

由于图像的每一个像素点都是由RGB（红、绿、蓝）三原色组成，而这三种颜色又可以组合成各种其它颜色，每个颜色占8位(如#FFFFFF)，LSB隐写即是修改每个颜色值的最低一位，将其替换为我们想要嵌入的信息中的内容，以此来实现数据隐藏。

一个像素点包含三种颜色，每个颜色修改最后1位，这样一个像素点就可以携带3位信息。

应用LSB算法的图像格式需为位图形式，即图像不能经过压缩，如LSB算法多应用于png、bmp等格式，而jpg格式较少。

详细参考：https://wenku.baidu.com/view/ff590e9d5f0e7cd1842536d7.html

Python Imaging Library

简称PIL，为Python解释器提供了**图像处理**的功能，结合PIL可以方便的编写Python脚本处理图片隐写问题。

StegSolve

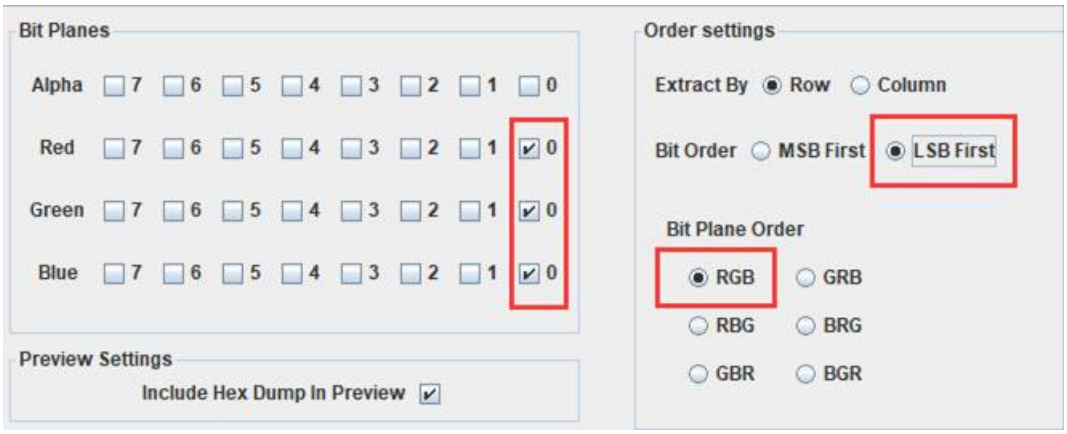
StegSolve是一款基于Java开发的流行**图片隐写分析软件**，其支持常见的图片文件格式，可以对不同的文件进行结合（包括XOR、ADD、SUB等操作），可以对图片文件格式进行分析，可以提取GIF文件中的帧等，覆盖了基本的图片隐写分析需求。

实验步骤

1. 用工具提取隐写信息

（1）.使用StegSolve打开图片后，选择如下选项进行分析。

（2）. 勾选如下标注的选项。（我们所说的RGB即是Red，Green，Blue三个颜色，而下图中的7,6，…1,0即是颜色用二进制表示时的高位到低位，我们是LSB最低有效位，所以自然选择0）



（3）. 点击下方的Preview即可查看最低位的隐藏信息，如下图。

从图上方我们可以看到flag字样，即隐藏的信息。

2. 用Python隐藏信息

使用Python实现LSB算法，将信息写入图片中。

1.以下便是一个简单的实现LSB算法的脚本代码，代码中的注释详细解释了算法实现的步骤。

```
# -*- coding: UTF-8 -*-

from PIL import Image



def plus(str):

      #Python zfill() 方法返回指定长度的字符串，原字符串右对齐，前面填充0。

    return str.zfill(8)



def get_key(strr):

      #获取要隐藏的文件内容

    tmp = strr

    f = file(tmp,"rb")

    str = ""

    s = f.read()
```

```
    for i in range(len(s)):
          #逐个字节将要隐藏的文件内容转换为二进制，并拼接起来
          #1.先用ord()函数将s的内容逐个转换为ascii码
          #2.使用bin()函数将十进制的ascii码转换为二进制
          #3.由于bin()函数转换二进制后，二进制字符串的前面会有"0b"来表示这个字符串是二进制形式，所以用replace()替换为空
          #4.又由于ascii码转换二进制后是七位，而正常情况下每个字符由8位二进制组成，所以使用自定义函数plus将其填充为8位
         str = str+plus(bin(ord(s[i])).replace('0b',''))
         #print str
    f.closed
    return str


def mod(x,y):
    return x%y;
#str1为载体图片路径，str2为隐写文件，str3为加密图片保存的路径
def func(str1,str2,str3):
    im = Image.open(str1)
    #获取图片的宽和高
    width = im.size[0]
    print "width:"+str(width)+"\n"
    height = im.size[1]
    print "height:"+str(height)+"\n"
    count = 0
    #获取需要隐藏的信息
    key = get_key(str2)
    keylen = len(key)
    for h in range(0,height):
        for w in range(0,width):
            pixel = im.getpixel((w,h))
            a=pixel[0]
            b=pixel[1]
```

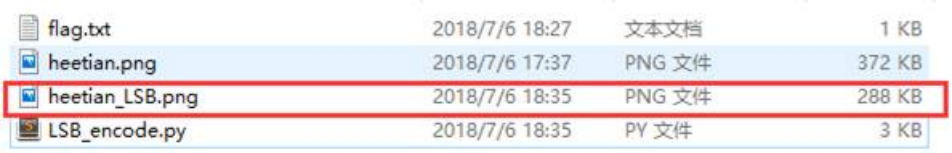
```
            c=pixel[2]
            if count == keylen:
                break
            #下面的操作是将信息隐藏进去
            #分别将每个像素点的RGB值余2，这样可以去掉最低位的值
            #再从需要隐藏的信息中取出一位，转换为整型
            #两值相加，就把信息隐藏起来了
            a= a-mod(a,2)+int(key[count])
            count+=1
            if count == keylen:
                im.putpixel((w,h),(a,b,c))
                break
            b =b-mod(b,2)+int(key[count])
            count+=1
            if count == keylen:
                im.putpixel((w,h),(a,b,c))
                break
            c= c-mod(c,2)+int(key[count])
            count+=1
            if count == keylen:
                im.putpixel((w,h),(a,b,c))
                break
            if count % 3 == 0:
                im.putpixel((w,h),(a,b,c))
    im.save(str3)
#原图
old = "C:\Users\lenovo\Desktop\LSB\demo1\heetian.png"
#处理后输出的图片路径
new = "C:\Users\lenovo\Desktop\LSB\demo1\heetian_LSB.png"
#需要隐藏的信息
```

```
#需要隐藏的信息

enc = "C:\Users\lenovo\Desktop\LSB\demo1\\flag.txt"

func(old,enc,new)
```

2. 运行脚本，将“H33Tian_BJHIT”写入图片之中，先创建文件flag.txt，并将字符串写入其中（本实验demo2文件夹中已创建）。

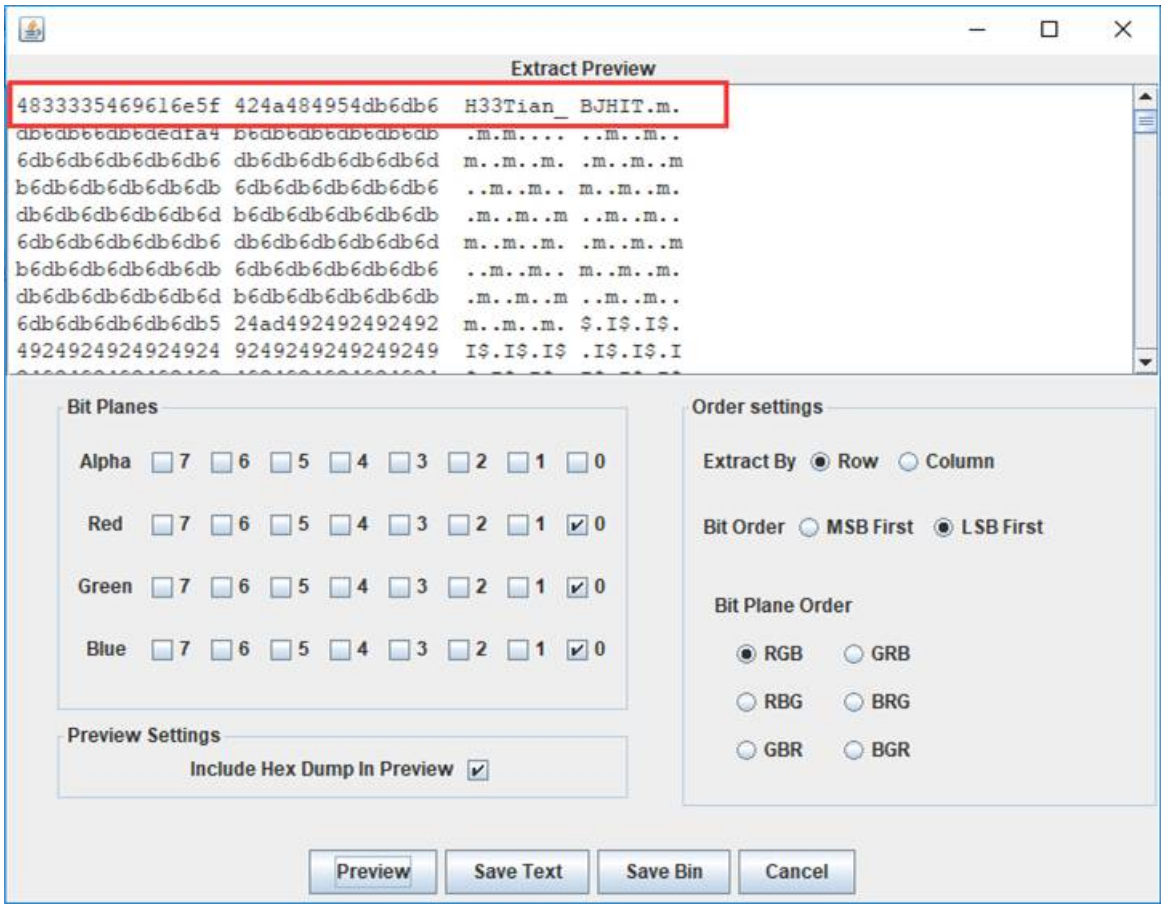
创建好后运行脚本（实验环境的文件所在路径与指导书不同，可自行进行更改）：

查看文件夹下，确实有新的图片生成。



3.对生成的图片进行验证，确认字符串是否写入成功，按实验步骤一的操作对新生成的图片进行操作。

字符串成功写入。



3. 用Python提取信息

编写脚本对实验步骤二所得的图片进行处理得到隐藏信息。

1. 实现脚本代码如下。

```
# -*- coding:UTF-8 -*-

from PIL import Image



def mod(x,y):

    return x%y;



def toasc(strr):

    return int(strr, 2)

#le为所要提取的信息的长度，str1为加密载体图片的路径，str2为提取文件的保存路径

def func(le,str1,str2):

    a=""

    b=""

    im = Image.open(str1)

    lenth = le*8

    width = im.size[0]

    height = im.size[1]

    count = 0

    for h in range(0, height):

        for w in range(0, width):

             #获得(w,h)点像素的值

            pixel = im.getpixel((w, h))

            #此处余3，依次从R、G、B三个颜色通道获得最低位的隐藏信息

            if count%3==0:

                count+=1

                b=b+str((mod(int(pixel[0]),2)))

                if count ==lenth:

                    break

            if count%3==1:

                count+=1
```

```
                b=b+str((mod(int(pixel[1]),2)))
                if count ==lenth:
                    break
            if count%3==2:
                count+=1
                b=b+str((mod(int(pixel[2]),2)))
                if count ==lenth:
                    break
        if count == lenth:
            break
    with open(str2,"wb") as f:
        for i in range(0,len(b),8):
             #以每8位为一组二进制，转换为十进制
            stra = toasc(b[i:i+8])
            #将转换后的十进制数视为ascii码，再转换为字符串写入到文件中
            f.write(chr(stra))
            stra =""
    f.closed
#文件长度
le = 30
#含有隐藏信息的图片
new = "C:\Users\lenovo\Desktop\LSB\demo2\heetian_LSB.png"
#信息提取出后所存放的文件
tiqu = "C:\Users\lenovo\Desktop\LSB\demo2\get_flag.txt"
func(le,new,tiqu)
```

2. 运行脚本，提取到指定长度的最低位信息，保存在get_flag.txt文件夹内。

3. 信息成功提取出来，但是脚本提取信息的缺点就是我们无法在事先知道所隐藏的信息有多长，所以一般直接设定脚本所提取的长度为目标图片的大小，将最低位的信息全部提取。

> 原文来源：合天网安实验室