【安全技术揭秘系列】探索图片隐写的奥秘

原创

白面安全猿 于 2021-10-15 15:39:19 发布 2405 收藏 9
分类专栏： 网络安全 文章标签： html5 系统安全 web安全 安全架构 安全

本文链接：https://blog.csdn.net/MSB_WLAQ/article/details/120784347


网络安全 专栏收录该内容
30 篇文章 7 订阅
订阅专栏

什么是隐写？由于我们识别声音或图片的能力有限，因此稍微改动信息的某字节位的数据是不会影响我们识别声音或图片的。举个最通俗的例子，古人的藏头诗就是隐写的一种方式：

芦花丛中一扁舟，
俊杰俄从此地游。
义士若能知此理，
反躬难逃可无忧。

而CTF图片隐写术就是利用图片来隐藏一些机密信息，一张看起来很正常很普通的图片其实内部隐藏了其他玄机。

图种（多文件压缩）

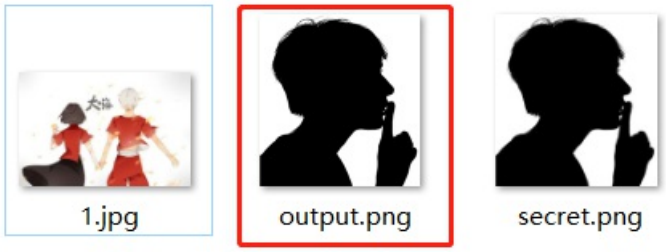
所谓图种，就是先把要想隐藏的东西打包压缩，然后再跟一张正常的图片结合起来，达到隐藏信息的目的。

可以直接使用CMD "**copy/b**" 直接压缩图片：

```
copy/b a.png + b output.png
```

例如，将一张jpg隐藏到另一张png图片中：

```
E:\CTF\test>copy/b secret.png + 1.jpg output.png
secret.png
1.jpg
已复制         1 个文件。
```



用kali中的"**binwalk**"工具可以根据检索匹配文件头的原理轻松地检索图片文件中隐藏的其他文件，还是以这张图片为例：

```
┌──(root💀kali)-[/root/CTF/test]
└─# binwalk -e output.png

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PNG image, 472 x 504, 8-bit/color RGB, non-interla
ced
31114         0x798A          JPEG image data, JFIF standard 1.01
```

找到隐藏的文件在kali 中可以使用"**foremost**"或者"**dd**"，将output.png进行分离，分离出的隐藏的文件：

```
DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PNG image, 472 x 504, 8-bit/color RGB, non-interla
ced
31114         0x798A          JPEG image data, JFIF standard 1.01


┌──(root💀kali)-[/root/CTF/test]
└─# foremost output.png
Processing: output.png
|*|

┌──(root💀kali)-[/root/CTF/test]
└─# dd if=output.png of=1.jpg bs=1 skip=31114
记录了16976+0 的读入
记录了16976+0 的写出
16976字节（17 kB，17 KiB）已复制，0.0213577 s，795 kB/s
```

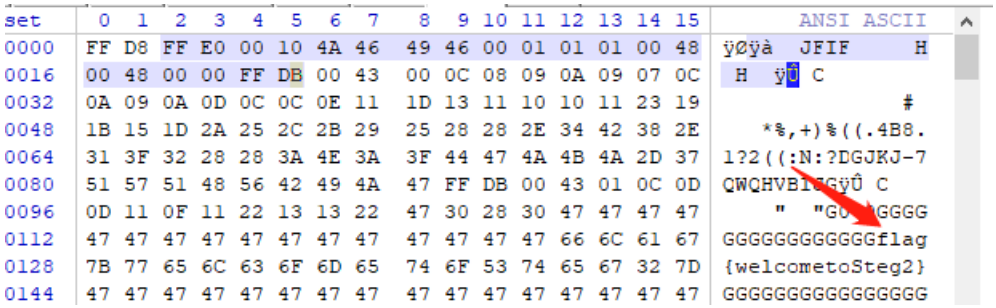
开始的字节

各位置信息隐藏

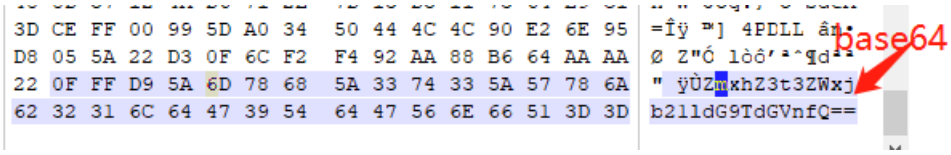
2.1 文件头尾

较为直接的方式，直接将文字信息隐藏在图片文件的头尾内容之中，例如下图：

使用16进制编辑工具（winhex）直接打开，可以发现再图片基础信息字段之后，直接插入了flag信息：



还有些是在文件尾部插入信息：



2.2 IHDR隐藏信息

文件头数据块IHDR(header chunk)：它包括有PNG文件里存储的图像数据的基本信息，并要作为第一个数据块出如今PNG数据流中，并且一个PNG数据流中仅仅能有一个文件头数据块。

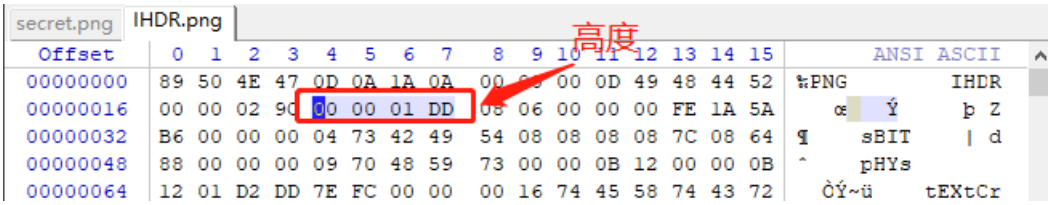
文件头数据块由13字节组成，它的格式如下表：

域的名称	字节数	说明
Width	4 bytes	图像宽度，以像素为单位
Height	4 bytes	图像高度，以像素为单位

Bit depth	1 byte	图像深度： 索引彩色图像：1，2，4或8 灰度图像：1，2，4，8或16 真彩色图像：8或16
ColorType	1 byte	颜色类型： 0：灰度图像, 1，2，4，8或16 2：真彩色图像，8或16 3：索引彩色图像，1，2，4或8 4：带α通道数据的灰度图像，8或16 6：带α通道数据的真彩色图像，8或16
Compression method	1 byte	压缩方法(LZ77派生算法)
Filter method	1 byte	滤波器方法
Interlace method	1 byte	隔行扫描方法： 0：非隔行扫描 1： Adam7(由Adam M. Costello开发的7遍隔行扫描方法)

在CTF图片隐写中，常常通过改变Height（4 bytes）来隐藏有效信息，例如下图：

使用16进制编辑工具（winhex）直接打开，尝试改变IHDR中的高度信息（PS:一般图片宽度信息无法修改，会造成图片乱码）“01 DD”——>“02 DD”：



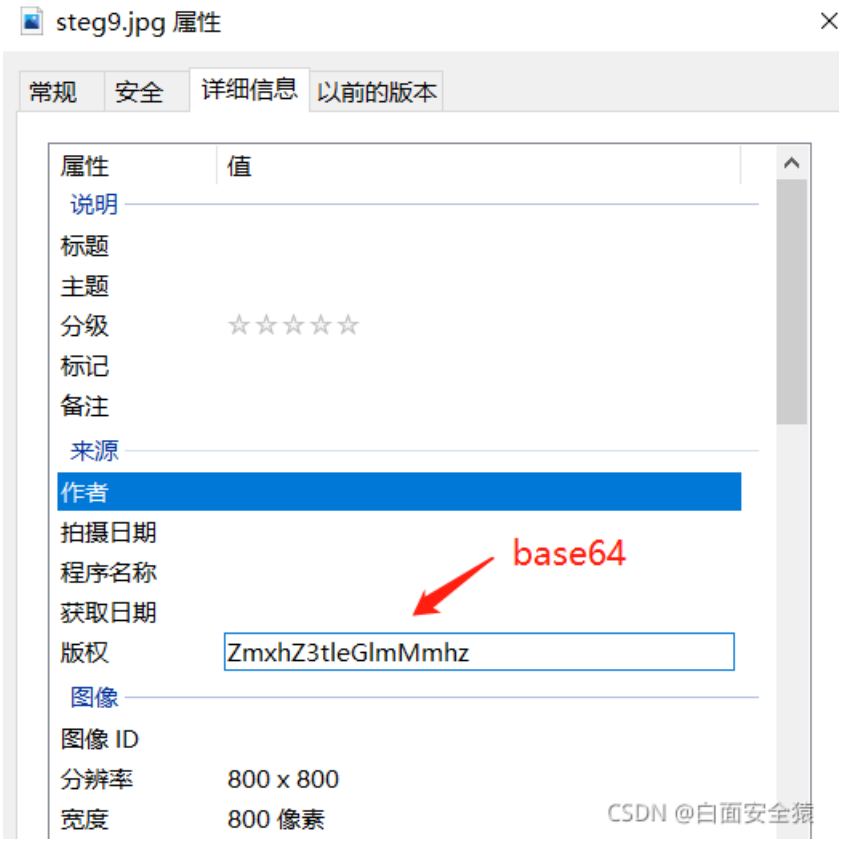
隐藏的flag信息通过高度调整，被暴露出来：

2.3 exif信息隐藏

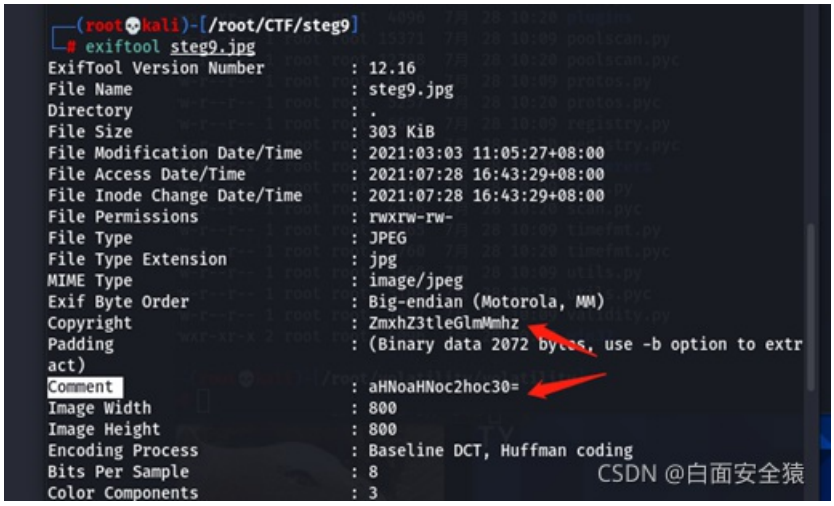
按照国际Exif2.1标准，当你用单反相机或手机拍摄完一张照片，相机会生成一个如JPEG格式的电子文件保存起来，这个jpeg图片它不只保存了你拍摄的画面，还保存很多其他的一些信息如：**相机品牌、手机型号、闪光灯、快门速度、光圈大小、感光度及GPS坐标等**，这些信息就是EXIF信息。最常见的就是：右键——>属性，这样只能够查看部分的exif信息。

当然这个信息是可以被篡改的，例如使用“PowerExif”等工具，在kali中，可以使用"**exiftool**"直观的查看图片各类exif信息：

例如下面这张照片，我们“右键->属性”只能看到部分的base64编码信息：



借助"**exiftool**"在Comment中发现另一部分编码信息：



文件修复

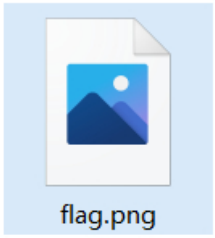
将文件头内容（16进制编码）修改或删除，会造成文件无法打开，损坏的问题，可以通过16进制编码器（winhex）进行修复。

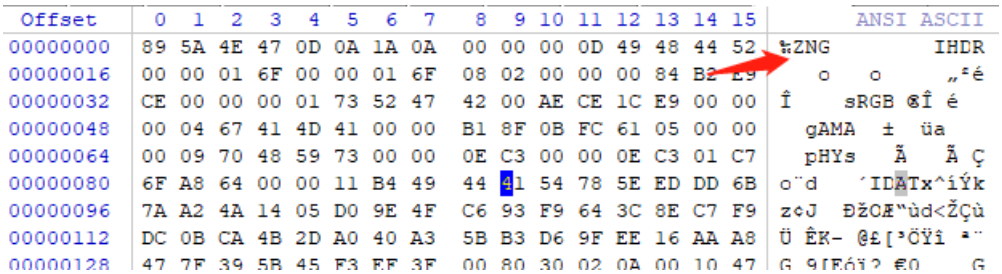
需要我们比较熟悉已知常见文件的文件头标准编码：

文件名	文件头	文件尾
JPEG (jpg)	FFD8FF	FFD9
PNG (png)	89504E47	AE426082

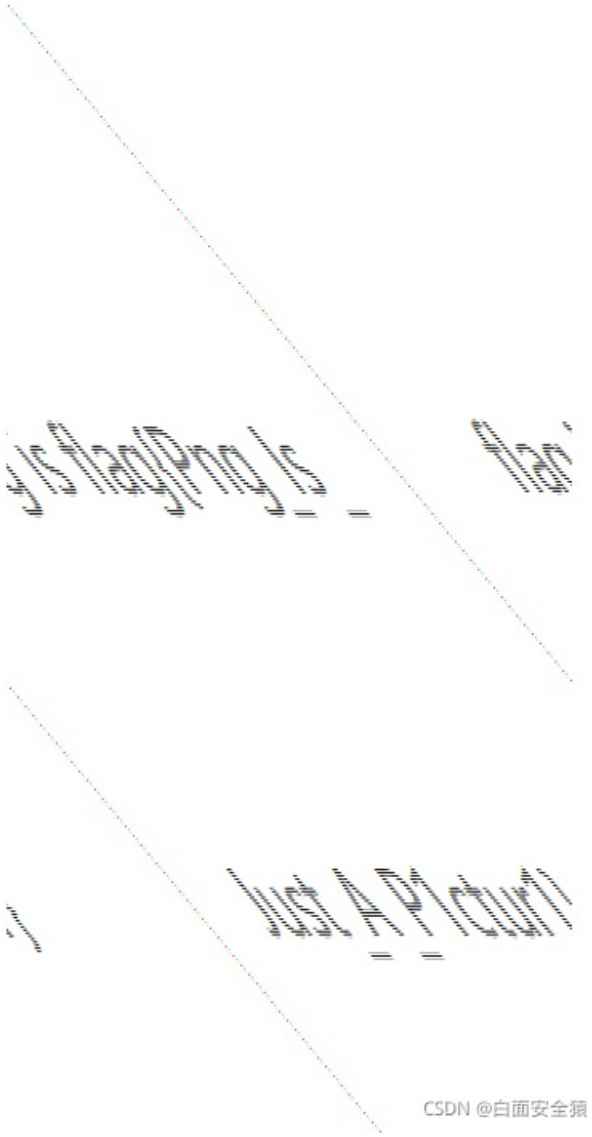
GIF (gif)	47494638	003B
ZIP Archive (zip)	504B0304	504B
TIFF (tif)	49492A00	
Windows Bitmap (bmp)	424D	
CAD (dwg)	41433130	
Adobe Photoshop (psd)	38425053	
Rich Text Format (rtf)	7B5C727466	
...更多参考		

例如下面这张无法打开的png图片，导入winhex后，发现文件头明显不符合PNG图片的标准编码：





"89 5A"——>"80 5A"改为PNG标准编码，发现图片可以正常打开了：



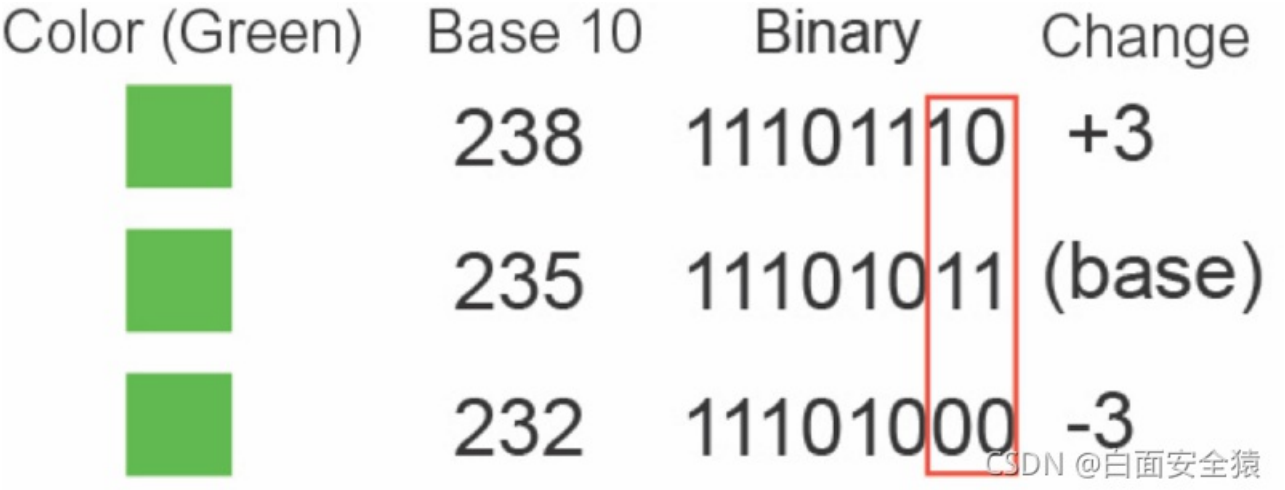
LSB隐写

LSB全称为 least significant bit，是最低有效位的意思。Lsb图片隐写是基于Lsb算法的一种图片隐写术，一般能使用LSB进行隐写的图片需要是无损压缩（png图片）或者无压缩的图片（BMP图片），因为这能保证我们修改的信息不至于丢失，也就能得到正确的表达。

以png图片为例子，png图片中的图像像数一般是由RGB三原色（红绿蓝）组成，每一种颜色占用8位，取值范围为0x00~0xFF，即有256种颜色，一共包含了256的3次方的颜色，即16777216种颜色。而人类的眼睛可以区分约1000万种不同的颜色，这就意味着人类的眼睛无法区分余下的颜色大约有6777216种。

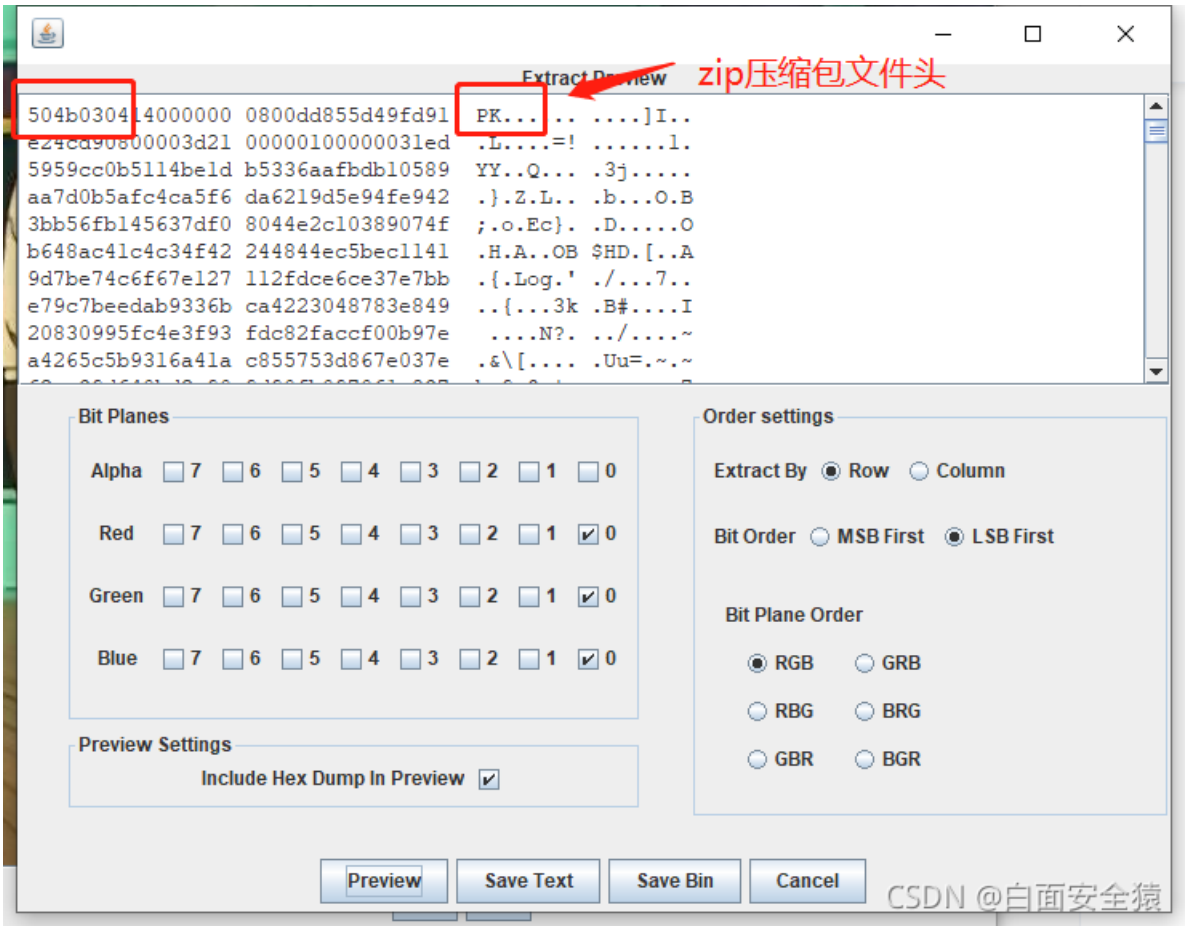
而LSB隐写算法就是去修改RGB颜色分量的最低二进制位也就是最低有效位（LSB），人类的眼睛不会注意到这前后的变化，每个像素可以携带3比特的信息。

例如下面这张经典的图：绿色，十进制的235，二进制的11101011，通过二进制运算加、减3修改最低有效位，发现肉眼根本区分不出来差别：



像下面这张图，通过"**StegSlove**"工具打开：

利用LSB算法提取最低有效位信息，可以有效的发现隐藏的压缩包文件：

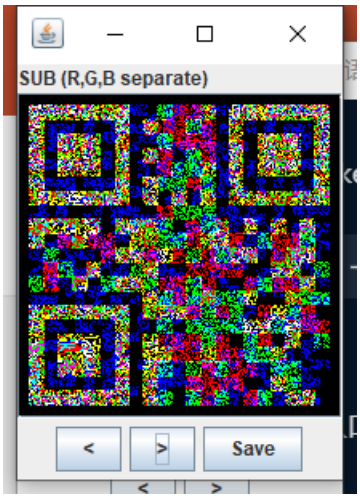


还有一种双图比较的题型，也是利用RGB图层更改的方式来隐藏信息：

first.png

second.png

将上面两张肉眼无法区分的图片，放入"**StegSlove**"中进行RGB图层处理，在SUB(R,G,B separate)发现了一张模糊二维码：



再将这张模糊二维码进行RGB图层处理，在RED、GREEN、BLUE plane 0 下分别发现3张清晰的二维码，扫码就得到了隐藏的具体信息：


3.png


4.png


5.png

总结

以上的几种就是CTF中常见的图片隐写方法和题型，而在实际的隐写技术中还有更多的隐写算法和隐藏技术。这里在总结下上述的思路：

在图像数据中加入数据，不影响视觉效果情况下修改像素数据，加入信息；（图种）
在图片右击查看属性，在详细信息（exif）中隐藏数据；（exif信息隐藏）
根据各种类型图像的固定格式，在各位置添加数据；（头、尾、IHDR信息隐藏）
在编译器中修改图像开始的标准编码，改变其原来图像格式；（文件修复）
将数据类型进行改写（rar或者zip数据改为jpg等格式）；（文件修复）
利用隐写算法将数据隐写到图片中而不影响图像,隐写常用的算法有LSB,jpghide；（LSB算法）

掌握类似的图片隐写技术，一方面可以帮助我们了解暗水印的原理，另一方面对我们在恶意文件隐藏（类似图片木马）检测的防御中，更有效的去取证溯源。