# 第七章 秘而不宣：隐写术

## 7.1 隐写术基本知识

#### 7.1.1 隐写术定义

隐写术是关于信息隐藏，即不让计划的接收者之外的任何人知道信息的传递事件（而不只是信息的内容）的一门技巧与科学。隐写术英文作“Steganography”，来源于约翰尼斯·特里特米乌斯一本看上去是有关黑魔法，实际上是讲密码学与隐写术的一本书Steganographia中。此书书名来源于希腊语，意为“隐秘书写”。

有两种办法可用来隐藏明文信息。隐写术，它可以隐藏信息的存在；而密码学则是通过对文本信息的不同转换而实现信息的对外不可读。

#### 7.1.2 隐写术的方法和释义

弗兰西斯·培根提出过一种隐藏消息的方法。来自现代实践的一个例子，掩饰文本相对隐秘文本的大小（指数据含量，以比特计）越大，隐藏后者就越加容易。因为这个原因，数字图像（包含有大量的数据）在因特网和其他传媒上被广泛用于隐藏消息。这种方法使用的广泛程度无从查考。例如：一个24位的位图中的每个象素的三个颜色分量（红，绿和蓝）各使用8个比特来表示。如果我们只考虑蓝色的话，就是说有2^8种不同的数值来表示深浅不同的蓝色。而像11111111和11111110这两个值所表示的蓝色，人眼几乎无法区分。因此，这个最低有效位就可以被(在某种程度上检测不到地)用来存储颜色之外的某些信息。如果我们对红色和绿色进行同样的工作的话，我们可以在（差一点不到）三个的像素中存储一个字节的信息。

更正式一点地说，使隐写的信息难以探测的，也就是保证“有效载荷”（需要被隐蔽的信号）对“载波”（即原始的信号）的调制对载波的影响看起来（理想状况下甚至在统计上）可以忽略。这就是说，这种改变应该无法与载波中的噪声加以区别。

（从信息论的观点来看，这就是说信道的容量必须大于传输“表面上”的信号的需求。这就叫做信道的冗余。对于一幅数字图像，这种冗余可能是成像单元的噪声；对于数字音频，可能是录音或者放大设备所产生的噪声。任何有着模拟放大级的系统都会有所谓的热噪声（或称“1/f”噪声)，这可以用作掩饰。另外，有损压缩技术（如JPEG）会在解压后的数据中引入一些误差，利用这些误差作隐写术用途也是可能的。）

隐写术也可以用作数字水印，这里一条消息（往往只是一个标识符）被隐藏到一幅图像中，使得其来源能够被跟踪或校验。

#### 7.1.3 隐写术的研究与应用

近几年来,隐写术领域已经成为了信息安全的焦点。因为每个Web站点都依赖多媒体,如音频、视频和图像。隐写术这项技术可以将秘密信息嵌入到数字媒介中而不损坏它的载体的质量。第三方既觉察不到秘密信息的存在,也不知道存在秘密信息。因此密钥、数字签名和私密信息都可以在开放的环境(如Internet,或者内联网)中安全的传送。主要介绍隐写术的基本原理以及各方面的应用。

## 7.2 隐写术类型简介

（以下题目中的图片，打包在：<http://pan.baidu.com/s/1mg1Khw0>）

#### 7.2.1 增加数据

最开始接触到隐写术，是看到一种叫做图种的东西，当时不懂，只说要另存为zip，然后解压出来就可以了，当时觉得特别神奇，就像发现了新大陆，然后就尝试了一下，发现可以用另存为zip的方式，用7z或者是winzip等工具打开，然后就可以看到福利了。

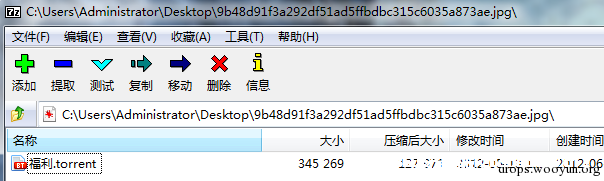
[](http://p2.qhimg.com/t010b2489268b47f3f2.png)

图1.png

后来才懂得了，先制作一个1.zip，把想要隐藏的东西放进去，再需要一张jpg图片2.jpg，然后就可以执行一个命令 copy /b 2.jpg+1.zip output.jpg。就可以得到一张图种，这是利用了copy命令，将两个文件已二进制方式连接起来，生成output.jpg的新文件。而在jpg 中，是有结束符的，16进制是FF D9，利用winhex可以看到正常的jpg结尾都是FF D9的，图片查看器会忽视jpg结束符之后的内容，所以我们附加的zip，自然也就不会影响到图像的正常显示。

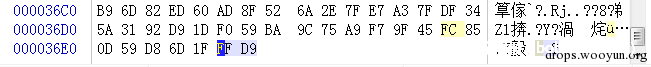
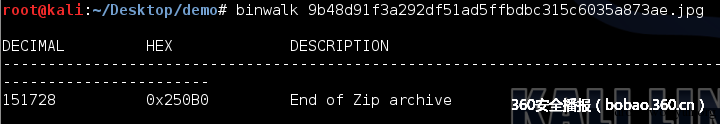
[](http://p1.qhimg.com/t0183a0ac310128fced.png)

图2.png

这种类型的隐写也是比较容易被发现的，如果发现是jpg图片的话，观察文件结束符之后的内容，查看是否附加的内容，正常图片都会是FF D9结尾的。还有一种方式来发现就是利用binwalk这个工具，在kali下自带的一个命令行工具。

[](http://p6.qhimg.com/t011e2ce3bd6ef780a7.png)

图片3.png

利用binwalk可以自动化的分析图片中附加的其他的文件，其原理就是检索匹配文件头，常用的一些文件头都可以被发现，然后利用偏移可以配合winhex或者是dd分割出隐藏的部分。

#### 7.2.2 修改数据

上面说到的隐藏方式，是利用了增加数据的方式，把数据直接增加在了jpg后面。还有另一类隐藏的方法，就是利用了修改数据的方式来隐藏自己传递的信息。

一种常见的方式是利用LSB来进行隐写，LSB也就是最低有效位 (Least Significant Bit)。原理就是图片中的像数一般是由三种颜色组成，即三原色，由这三种原色可以组成其他各种颜色，例如在PNG图片的储存中，每个颜色会有 8bit，LSB隐写就是修改了像数中的最低的1bit，在人眼看来是看不出来区别的，也把信息隐藏起来了。譬如我们想把’A’隐藏进来的话，如下图，就 可以把A转成16进制的0x61再转成二进制的01100001，再修改为红色通道的最低位为这些二进制串。

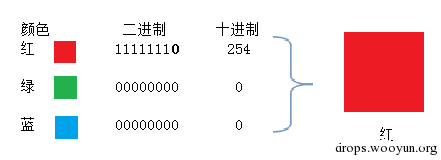
[](http://p0.qhimg.com/t01383e5f817a8a43b2.png)

图4.png

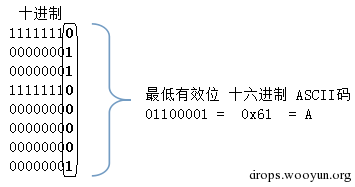
[](http://p6.qhimg.com/t0117d66e668716f02b.png)

图5.png

如果是要寻找这种LSB隐藏痕迹的话，有一个工具是个神器，可以来辅助我们进行分析。Stegsolve这个软件的下载地址是

http://www.caesum.com/handbook/Stegsolve.jar

打开之后，使用Stegsolve——Analyse——Frame Browser这个可以浏览三个颜色通道中的每一位，可以在红色通道的最低位，发现一个二维码，然后可以扫描得到结果。

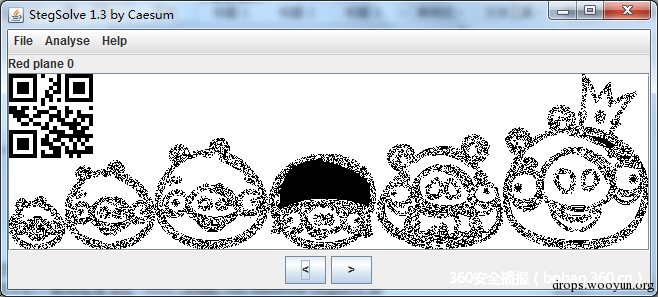
[](http://p7.qhimg.com/t01bfc89c0b6246fdae.png)

图6.png

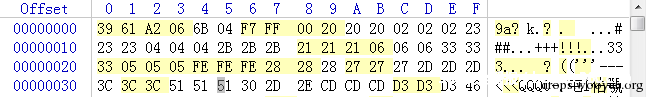
再解一下qrcode，用在线的就可以http://tool.chinaz.com/qrcode/，得到了flag{AppLeU0}，如果是 隐写的使用了ascii的话，可以使用Stegsolve——Analyse——Data Extract来查看ascii码。

在这个过程中，我们要注意到，隐写的载体是PNG的格式，如果是像之前的jpg图片的话就是不行的，原因是jpg图片对像数进行了有损的压缩，你修 改的信息可能会被压缩的过程破坏。而PNG图片虽然也有压缩，但却是无损的压缩，这样子可以保持你修改的信息得到正确的表达，不至于丢失。BMP的图片也 是一样的，是没有经过压缩的，可以发现BMP图片是特别的大的，因为BMP把所有的像数都按原样储存，没有压缩的过程。

#### 7.2.3 隐写与加密

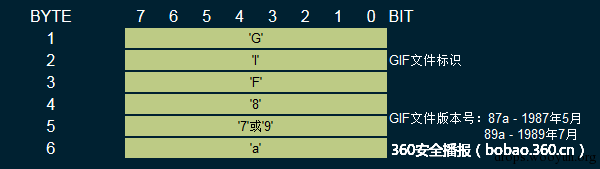
我们先要区分一个概念，隐写术和加解密的区别。其实说起来很简单，加解密的话，就是会出现一些神秘的，可疑的字符串或者是数据之类的。而隐写术的 话，就是信息明明就在你的面前，你却对他视而不见。隐写术在CTF中出现时，常常会和加解密结合起来一起出现，或者是一些编码方式一起出现，以提高题目的 难度。

用一个ctf的题目作为例子吧，iscc2014中有一个题目，给了一个名为 此为gif图片.gif的文件，打开发现了报错。有的时候，会需要我们去修复图片，这对我们对于图片的文件结构要有了解。找到gif的文件格式，然后对照 这个破损的文件。Gif的图片格式文档可以查看这个链接，http://dev.gameres.com/Program/Visual/Other /GIFDoc.htm

[](http://p1.qhimg.com/t01eab91a189cc54ad3.png)

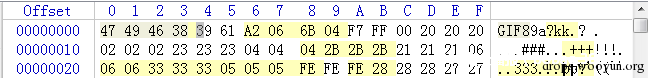
图片8.png

用winhex打开，我们会发现他和普通的GIF图片不一样，头部缺少了东西，在对比一些文档，会发现是少了GIF8。

[](http://p2.qhimg.com/t01eeb6b0884cbd9d50.png)

图片9.png

我们手动修复一下，增加GIF8。

[](http://p3.qhimg.com/t01d29119ecf2e4a101.png)

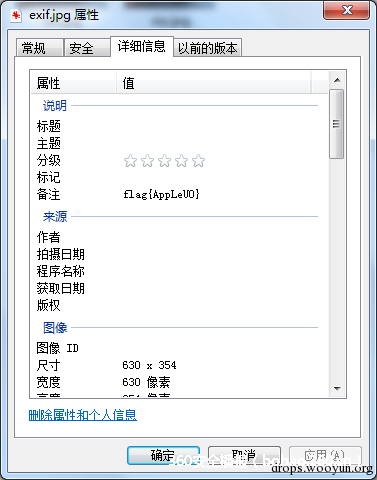
图片10.png

然后浏览图片后会发现，有个PASSWORD一闪而过，gif和别的图片最大的区别就是gif是动态图，它是可以由多帧组成的可以顺序播放的，有的题就是把播放的时间弄得特别慢，几乎就不会动的，所以我们可以用工具一帧一帧的观察图片。Stegsolve就带有这种功能。

Stegsolve——Analyse——Frame Brower就可以看到是有8帧的图片，有点重叠不太好观察，也可以用Namo\_GIF\_gr这个工具。得到了PASSWORD is Y2F0Y2hfdGhlX2R5bmFtaWNfZmxhZ19pc19xdW10ZV9zaW1wbGU=。很明显，这个时候PASSWORD是经过 的编码的，我们可以看到字符范围是0-9a-Z结尾还有=，所以判断是base64编码，解码得到了 catch\_the\_dynamic\_flag\_is\_qumte\_simple。这个就是和编码方式结合，传递一些可疑的数据，隐写术常常会与加解密或 编码结合在一起，对一些常见的编码和加密方法也要了解，得到密文的字符范围和长度能发现这是什么加密或者是编码。

#### 7.2.4 载体

数据在隐藏的时候，我们常常是需要先分析是数据隐藏在哪里，也就是他在利用是什么做载体，之后才可以进一步的分析是加密或编码的。这也就是说我们要 对一个图片的格式要有了解，才能知道哪些地方是可疑的，哪些是可以隐藏起信息的，会有冗余的成分在。举个例子吧，比如给了一个jpg的图片。除了我们之前 说到的隐藏在结束符之后的信息，jpg图片还可以把信息隐藏的exif的部分。exif的信息是jpg的头部插入了数码照片的信息，比如是用什么相机拍摄 的。这些信息我们也是可以控制的，用查看属性的方式可以修改一部分的信息，还可以用exif编辑器来进行编辑。Power\_exif这个可以用来编辑。

[](http://p4.qhimg.com/t01101e3e86f5db4677.png)

图片11.png

可以看到flag{AppLeU0}，就是需要了解隐藏信息的地方，隐写术有的时候难，就是难在了一张图片有太多的地方可以隐藏信息了，有的时候根本连隐藏的载体都找不到，在你的眼里他就是一张正常的图片。

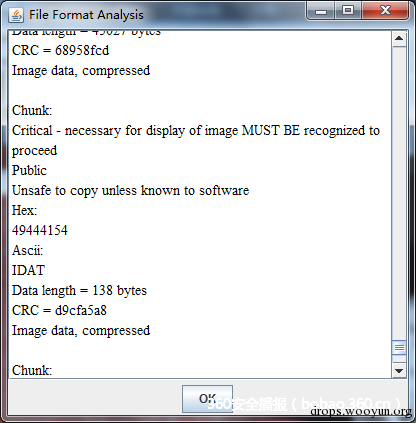
#### 7.2.5 编程辅助

有一些情况下，我们也是没有现成的工具来完成的，可以自己写一些简单的程序来辅助我们进行分析，或者是加解密。比如sctf的misc400的题目，就需要用到一些简单的编程。题目给出了一个png图片，需要我们找到有SCTF{}标志的flag。

这个题需要我们对于png图片的格式有一些了解，先用stegsolve查看一下，其他的LSB之类的并没有发现什么问题，然后看了一下结构发现，有一些异常的IDAT块。IDAT是png图片中储存图像像数数据的块。Png图片格式的扩展阅读可以看看这篇

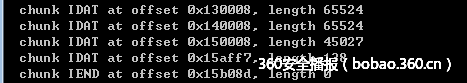
http://www.cnblogs.com/fengyv/archive/2006/04/30/2423964.html

有详细的介绍。

[](http://p6.qhimg.com/t01a4eb29be11731f31.png)

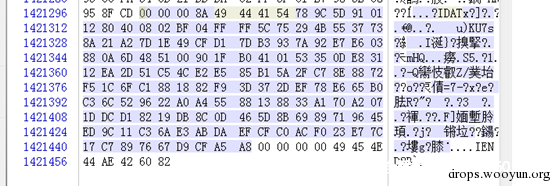
图片12.png

可以用pngcheck来辅助我们观察，可以看得更加清晰。pngcheck.exe -v sctf.png

[](http://p0.qhimg.com/t0198e90206e24c011c.png)

图片13.png

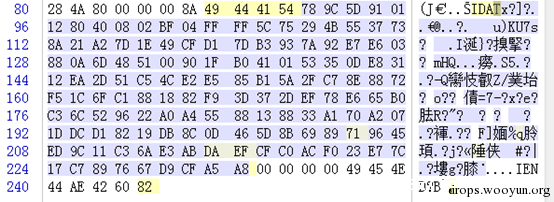
可以看到，正常的块的length是在65524的时候就满了，而倒数第二个IDAT块长度是45027，最后一个长度是138，很明显最后一个IDAT块是有问题的，因为他本来应该并入到倒数第二个未满的块里。

[](http://p3.qhimg.com/t013b79c94499ae8ed3.png)

图片14.png

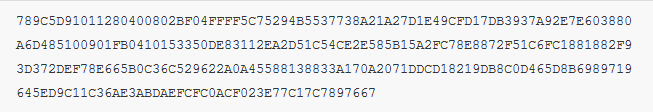
我们用winhex把这一部分异常的IDAT块给扣出来。然后就是要研究研究这个块是什么情况，发现了载体之后就是要想办法找出他的规律。观察那一 部分的数据，可以看到是16进制的78 9C开头的，百度一下分析是zlib压缩的标志。在png的百度百科里也可以查到PNG的IDAT是使用从LZ77派生的无损数据压缩算法，可以用 zlib解压。那么就尝试用zlib来解一下这段数据。Zlib的扩展阅读http://zlib.net/

我们使用python来编程，先把那段数据处理一下，保存成16进制的。

[](http://p7.qhimg.com/t014cbf159720a4fd5a.png)

图片15.png

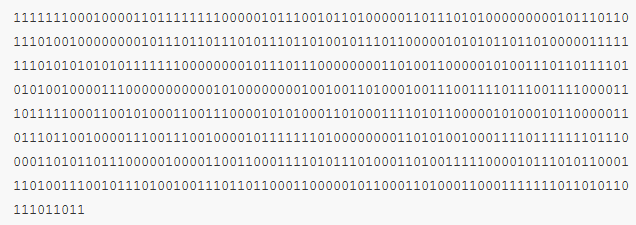
得到16进制的以方便python处理，前面的4字节是长度 然后是标志位IDAT 然后开始是数据，直到 D9 CF A5 A8是crc32校验位。 所以实际的数据是：

[](http://p2.qhimg.com/t01133592fbbb79650e.png)

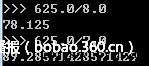
然后用python来写zlib解压

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | #! /usr/bin/env python  import zlib  import binascii  IDAT = "789C5D91011280400802BF04FFFF5C75294B5537738A21A27D1E49CFD17DB3937A92E7E603880A6D485100901FB0410153350DE83112EA2D51C54CE2E585B15A2FC78E8872F51C6FC1881882F93D372DEF78E665B0C36C529622A0A45588138833A170A2071DDCD18219DB8C0D465D8B6989719645ED9C11C36AE3ABDAEFCFC0ACF023E77C17C7897667".decode('hex')  #print IDAT  result = binascii.hexlify(zlib.decompress(IDAT))  print result  #print result.decode('hex') |

发现解出来了一些3031的字符串，30和31是hex的 0和1的编码，再解一次hex得到一串625长度的01字符串。

[](http://p2.qhimg.com/t01c9a2bd97d3e2c236.png)

得到的01 串的长度是625，除以8 除以7 都无法整除，也就是说没法直接转换成ascii码。

[](http://p8.qhimg.com/t01d6e059e505318d55.png)

图片16.png

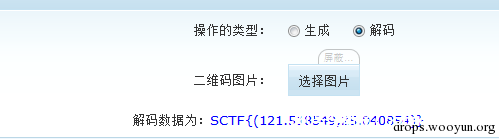
然后发现625 = 25\*25，刚好是个正方形的形状，那么尝试一下 把这些01 组成一个正方形 看看是什么，可以用python的PIL编程可以很方便的画图，在kali自带就可以有，win的环境需要安装PIL的第三方库。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | #!/usr/bin/env python  import Image  MAX = 25  pic = Image.new("RGB",(MAX, MAX))  str = "1111111000100001101111111100000101110010110100000110111010100000000010111011011101001000000001011101101110101110110100101110110000010101011011010000011111111010101010101111111000000001011101110000000011010011000001010011101101111010101001000011100000000000101000000001001001101000100111001111011100111100001110111110001100101000110011100001010100011010001111010110000010100010110000011011101100100001110011100100001011111110100000000110101001000111101111111011100001101011011100000100001100110001111010111010001101001111100001011101011000111010011100101110100100111011011000110000010110001101000110001111111011010110111011011"  i=0  for y in range (0,MAX):      for x in range (0,MAX):          if(str[i] == '1'):              pic.putpixel([x,y],(0, 0, 0))          else:              pic.putpixel([x,y],(255,255,255))          i = i+1  pic.show()  pic.save("flag.png") |

发现是一个二维码 可以编码来画出 0代表了是白色 而1代表了黑色，然后可能会需要旋转来调整一下，才能扫描出来。处理一下得到了一个二维码。然后扫描得到了flag。

[](http://p0.qhimg.com/t01ff0dc093daeb6d6c.png)

图片17.png

[](http://p1.qhimg.com/t016c379f4177f73202.png)

图片18.png

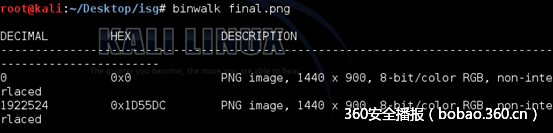
SCTF{(121.518549,25.040854)}，成功得到了flag。

在有的情况下，是没法用现成的工具来处理的，所以就要我们用编程来设法解决。Python的PIL是个好东西。批量处理图片的时候可能会需要它。

#### 7.2.6 双图

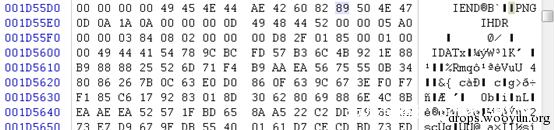
还有一种情况是比较特殊的，有的时候会给出两张图片，或者是需要你去寻找原来的图片来进行对比寻找隐藏的信息。这个一般是因为一张图片给出来的隐藏 信息太过于隐蔽，无法找不到具体的位置，具体的信息。这个时候就要用到一些对比的技巧来查找了。比如ISG2014的misc200就是用到的这种给出了 两张图的。有的情况下，第二张图是需要你自己去找到的。

我们来看isg2014-misc200的题，题目给了一张png图片，png的图片，就怕里面插个什么rar之类的，所以先用linux下的binwalk命令跑一跑。

[](http://p1.qhimg.com/t01d13df604415acfb4.png)

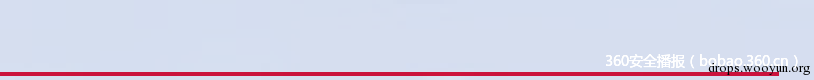
图片19.png

跑一跑，发现了有两个PNG图片，binwalk会给出偏移，确定了偏移是0x1D55DC之后，用winhex把图片扣出来，保存成2.png。 原来的图final.png删除后面那的一部分，保存成1.png。肉眼查看了一下，发现两张图片没有太大的区别，我们用软件来帮助我们区分他。

[](http://p8.qhimg.com/t010255ac4abafa96ad.png)

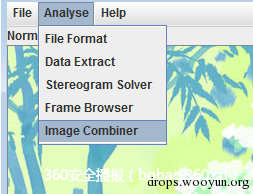
图片20.png

用linux下的命令可以进行对比，生成一个有差异的图片diff.png。compare 1.png 2.png diff.png 观察一下发现了左下角有异常，png图片像数保存是从左到右，从下往上排列的。

[](http://p9.qhimg.com/t011a951f438d06ae37.png)

图片21.png

发现了左下的第二条像素有异常，对比一下1.png 2.png发现了2.png有问题 那么我们可以用神器stegsolve来辅助，stegsolve——Analyse——Image Combiner对比两个文件。查看Sub或Xor，可以发现左下角，第二条像数条是有异常的，有红色的出现。

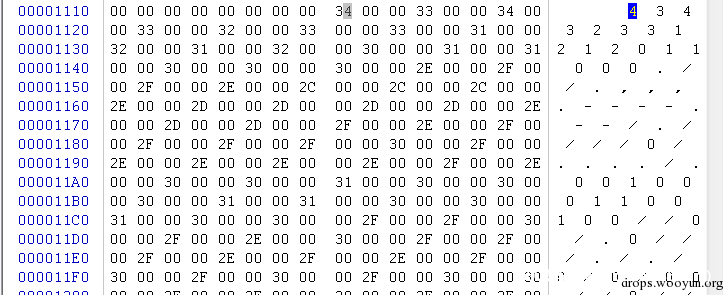
[](http://p7.qhimg.com/t01f9d066077744336f.png)

图片22.png

把1.png和2.png进行一下sub方法 把结果保存成solved.bmp。

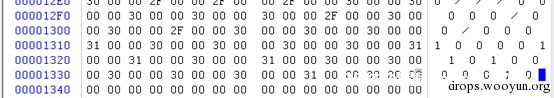
然后把2.png保存成2.bmp 24位位图的格式，这个是因为png图片经过了压缩，不好直接对比每个字节，而bmp图片是没有压缩的，直接保存各个像数点的数据。

这个题还有一个坑点就是偏移的问题 png图片的扫描是从左向右，从下往上来的。而坑的是这个图的信息隐藏并没有在一开头的像数，而是是第二行像数，所以就需要利用bmp的优势，储存无压 缩，方便寻找到偏移，从而找到信息隐藏的地方。利用winhex打开，黑色的像数的在bmp中的hex的00保存的，那么我们就寻找不是00的地方。在偏 移0x1110的地方可以发现

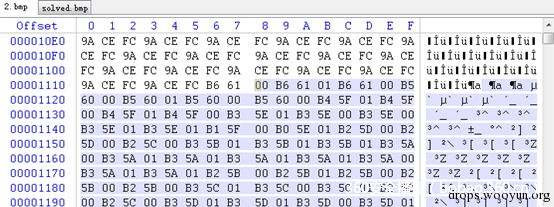
[](http://p7.qhimg.com/t01af7bc8e27ab3d95b.png)

图片23.png

有不是00的字节，一开始还以为这些就是flag的信息了，后来才发现是因为两个图片sub影响到了效果，真正的信息是隐藏在2.png中的，所以 打开由2.png转换的2.bmp来对，通过之前diff得到的偏移，寻找到0x1110的地方，直到0x1330结束，这是隐藏的信息。

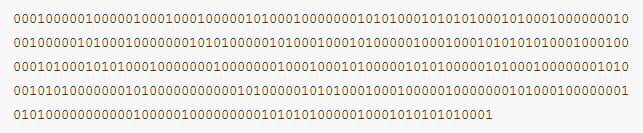
[](http://p1.qhimg.com/t01a3c65591d523c2b2.png)

图片24.png

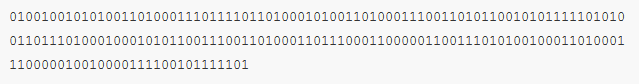
[](http://p5.qhimg.com/t01c88d365a9dc99f3f.png)

图片25.png

只保留00 01，这个是因为RGB的关系，只隐藏在R通道里面了，其他通道都是图片的正常像数信息，过滤掉就可以了。

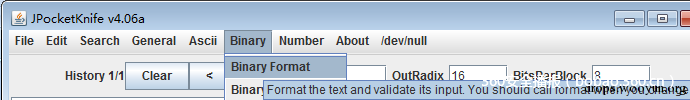
[](http://p5.qhimg.com/t01a17a2edc5d723bba.png)

观察一下可以发现，而奇数位都是0，是多余的，把这些去除掉。直接把00 替换成0，01替换成1就可以了。

[](http://p8.qhimg.com/t014cc341c41c5ac033.png)

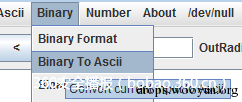
得到了这个之后，可以发现他的长度是184，是8的倍数，把他转换成ascii码就可以了。可以使用JPK工具来进行转换，工具的下载的链接是www.wechall.net/applet/JPK\_406.jar。

对比2.bmp可以发现隐藏了一些00 01这些信息，把这一部分扣出来。

[](http://p1.qhimg.com/t015197f9b59730d630.png)

图片26.png

JPK——binary——binary to ascii

[](http://p4.qhimg.com/t01aea8a815cc0fa834.png)

图片27.png

就得到了flag，ISG{E4sY\_StEg4n0gR4pHy}

这种就是利用的两张图片对比来寻找差异，从而找到信息隐藏的地方，这样子出题往往是因为一张图片能提供的信息太少。

#### 7.2.7 音频

这是一个关于音频隐写术英文的网站：<https://solusipse.net/blog/post/basic-methods-of-audio-steganography-spectrograms/>

音频类的题目和密码学有关，这里分享一道例题和writeup：<http://blog.csdn.net/captivate_guan/article/details/73715882>

还有一个MP3音频隐写的解题分享：<http://www.sohu.com/a/116298237_468673>

## 7.3 隐写术所用工具和做题技巧

#### 7.3.1 隐写术所用到的工具

（1）Stegsolve图片分析软件： 用于分析图片，图片是由各种不同颜色的像素点构成的（涉及LSB等图片图形学内容，感兴趣的可以自己去查），这个工具可以很容易的把不同颜色通道的图片信息显示出来。

（2） 二维码扫描： 在图片隐写中，扫描二维码得到答案的题目是比较常见的，但是有时候扫描之后出现的也不一定是最后答案，往往需要我们进一步的去破解，所以用手机扫描注定不方便，有个电脑上的软件会方便很多。

（3）十六进制分析工具：与这个相同的还有winhex，winhex更常用，且功能更强大，想使用哪个看个人喜好。计算机的世界，所有的东西都是由1和0组成的，二进制也可以转成十六进制，所以利用这个工具可以看到构成图片的最原始的数据，同时可以找到隐藏在图片里的秘密。

4）binwalk:这是在kali下自带的一个命令行工具，可以用来分析目标文件里是否有夹带其他东西，这在除了隐写以外的其他类型的题也是常用的工具。命令：binwalk –n 加文件 。比如说一张图片里面隐藏了一个压缩包，从表面上你是看不出什么的。但是使用binwalk命令，就可以分析出来。同样，把图片丢到十六进制查看器里面也可以看出来，但是新手对这些不熟悉很容易出错。毕竟压缩文件除了rar还有zip，又或许图片里面藏的是个txt。

另外，分析之后我们需要把这附带的其他文件提取出来，这就需要另外的一个命令binwalk –e 加文件。同样可以使用十六进制查看器，通过分离切割来得到，有兴趣的同样可以自己试试。

（5）PS:这个东西大家肯定不会陌生，有关图片隐写，ps有时候会用到，但几率不大。因为前面介绍的几种工具基本已经够用了，ps无非也就是有更进一步的能力而已。与ps相应的还有一个叫做FW的工具，也是Adobe系列的。要是一张图片真的什么都分析不出来，丢到PS里，说不定就会发现惊喜了，但是这样的题至今我也只遇过三题，其中两题都是跟FW有关的。

（6）密码学：这就涉及到一个很大的方面了。有时候我们找到了图片里面藏的密码，但是出题人不会很直白的告诉我们答案，就会进行加密。常见的密码有md5,base64，当然还有各种各样奇怪的密码。不过野不用着急，这都是一个积累的过程，先去了解md5和base64的特征，然后拿解到的密码去网上搜索。大部分常见的密码类型网上都有在线解码网站

#### 7.3.2 隐写术的做题技巧

总结1：

（1）一般拿到一张图片，先看图片格式。常见的有JPG，PNG和BMP。JPG比起PNG，是有过压缩的图片，所以JPG格式的图片通常是没必要放到图片分析器里去看的。由于进过压缩，所以里面是不能放下二维码的。而我们使用图片分析器来分析图片，60％是为了找到里面隐藏的二维码。

（2）JPG格式的图片通常里面都有藏着另外的东西，大体上是压缩文件之类的，当然，也有往里面放音频的……所以先丢去binwalk分析才是正确的。

（3）PNG格式的图片首先就可以考虑藏二维码的问题了，不过也不是说他里面不会藏有其他文件哟

（4）BMP是位图文件,能放的东西就更多了，可以拿去PS里看一下，说不定在就出现了两个图层了。

（5）无格式的东西，图片隐写的题，出题人很喜欢绕一个弯。有时候下载下来的东西是没有后缀格式的，这时候需要手动去添加。丢到十六进制查看器里去看它的头部十六进制，取前四个到百度搜一下就知道是什么格式了。同理，也有人会把头部文件给删除，这时候要靠他的后缀手动添加头部。这些都是在十六进制查看器里完成的。

（6）遇到音频的时候，音频也是可以藏东西的。一般是喜欢用AU，当然也有其他比较专门的工具，可以自己去找找。查看音频的波形，有时候里面会藏有一段莫尔斯电码。

总结2：

* 图种,只要将图片保存为zip压缩包格式,然后解压出来就可以了
* jpg的结束符:FF D9
* 观察文件结束符之后的内容，查看是否附加的内容，正常图片都会是FF D9结尾的。还有一种方式来发现就是利用binwalk这个工具，在kali下自带的一个命令行工具。利用binwalk可以自动化的分析图片中附加的其他的文件，其原理就是检索匹配文件头，常用的一些文件头都可以被发现，然后利用偏移可以配合winhex或者是dd分割出隐藏的部分。
* 还有另一类隐藏的方法就是利用修改数据的方式来隐藏自己传递的信息.利用LSB来进行隐写.如果是要寻找这种LSB隐藏痕迹的话，有一个工具是个神器，可以来辅助我们进行分析。Stegsolve这个软件….打开之后，使用Stegsolve——Analyse——Frame Browser这个可以浏览三个颜色通道中的每一位，可以在红色通道的最低位，发现一个二维码，然后可以扫描得到结果。
* 隐写的载体是PNG的格式，如果是像之前的jpg图片的话就是不行的，原因是jpg图片对像数进行了有损的压缩，你修改的信息可能会被压缩的过程破坏。而PNG图片虽然也有压缩，但却是无损的压缩，这样子可以保持你修改的信息得到正确的表达，不至于丢失。BMP的图片也是一样的，是没有经过压缩的，可以发现BMP图片是特别的大的，因为BMP把所有的像数都按原样储存，没有压缩的过程。
* qrcode，用在线的就可以解开<http://tool.chinaz.com/qrcode/>
* **Stegsolve**
* **Stegdetect**
* 一个jpg的图片。除了我们之前说到的隐藏在结束符之后的信息，jpg图片还可以把信息隐藏的exif的部分。exif的信息是jpg的头部插入了数码照片的信息，比如是用什么相机拍摄的。这些信息我们也是可以控制的，用查看属性的方式可以修改一部分的信息，还可以用exif编辑器来进行编辑。
* 隐写术有的时候难，就是难在了一张图片有太多的地方可以隐藏信息了，有的时候根本连隐藏的载体都找不到，在你的眼里他就是一张正常的图片。
* 这个题需要我们对于png图片的格式有一些了解，先用stegsolve查看一下，其他的LSB之类的并没有发现什么问题，然后看了一下结构发现，有一些异常的IDAT块。IDAT是png图片中储存图像像数数据的块。
* 解隐写题的思路:
  + 首先看看图片是不是图种
  + 然后使用binwalker分析图片,如果是有文件合成,分离
  + 有时信息藏在备注中
  + 工具stegsolve
  + 编程分析
* png图片像数保存是从左到右，从下往上排列的。
* png图片经过了压缩，不好直接对比每个字节，而bmp图片是没有压缩的，直接保存各个像数点的数据。
* **mp3stego**

## 7.4 后记

#### 7.4.1 关于隐写术文件的介绍：

1. 隐写术入门篇——图像文件后缀一览表：

文件格式是存储文本、图形或者图像数据的一种[数据结构](http://lib.csdn.net/base/datastructure)。在文字处理中，存储文本文件要使用文件格式。例如，使用微软公司的Word处理器编写的文件，可根据不同的应用环境用不同的格式存储。如果使用多信息文本格式(Rich Text Format，RTF)存储，这个文件就可在其他的平台(如Mac机)或者使用其他的字处理器进行处理。同样，存储图像也需要有存储格式，从20世纪70年代图像开始进入计算机以来，开发了许许多多的图像文件存储格式，而且互相不兼容，需要使用针对特定格式的处理软件。现在都意识到，不兼容的格式给用户造成很多的不便，因此有些格式也逐渐被淘汰。

在计算机中，有两种类型的图：矢量图(vector graphics)和位映象图(bitmapped graphics)。矢量图是用数学方法描述的一系列点、线、弧和其他几何形状，如图6-17(a)所示。因此存放这种图使用的格式称为矢量图格式，存储的数据主要是绘制图形的数学描述；位映象图(bitmapped graphics)也称光栅图(raster graphics)，这种图就像电视图像一样，由象点组成的，如图6-17(b)，因此存放这种图使用的格式称为位映象图格式，经常简称为位图格式，存储的数据是描述像素的数值。

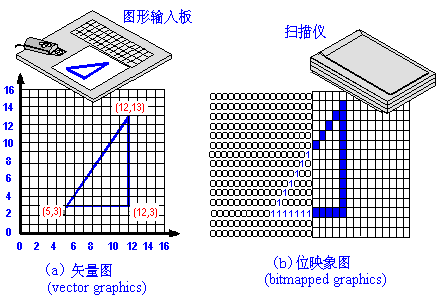


图6-17 矢量图与位映象图

除了本章介绍的4种常用格式之外，在我们的工作中还会遇到其他图像格式。为方便查阅，现将部分图形与图像文件的后缀和名称列在表6-11@和表6-12@中。如果编写程序需要很专业的图像格式资源，包括一些源程序(source code)，可以访问站点：http://www.wotsit.org/，你可饱览多媒体世界中的各种媒体的存储格式。

表6-11@ 位映象图格式/光栅图光栅(bitmapped formats / raster graphics)

|  |  |  |  |
| --- | --- | --- | --- |
| 后缀 | 文件名称 | 后缀 | 文件名称 |
| AG4 | Access G4 document imaging | JFF | JPEG (JFIF) |
| ATT | AT&T Group IV | JPG | JPEG |
| BMP | Windows & OS/2 | KFX | Kofax Group IV |
| CAL | CALS Group IV | MAC | MacPaint |
| CIT | Intergraph scanned images | MIL | Same as GP4 extension |
| CLP | Windows Clipboard | MSP | Microsoft Paint |
| CMP | Photomatrix G3/G4 scanner format | NIF | Navy Image File |
| CMP | LEAD Technologies | PBM | Portable bitmap |
| CPR | Knowledge Access | PCD | PhotoCD |
| CT | Scitex Continuous Tone | PCX | PC Paintbrush |
| CUT | Dr. Halo | PIX | Inset Systems (HiJaak) |
| DBX | DATABEAM | PNG | Portable Network Graphics |
| DX | Autotrol document imaging | PSD | Photoshop native format |
| ED6 | EDMICS (U.S. DOD) | RAS | Sun |
| EPS | Encapsulated PostScript | RGB | SGI |
| FAX | Fax | RIA | Alpharel Group IV document imaging |
| FMV | FrameMaker | RLC | Image Systems |
| GED | Arts & Letters | RLE | Various RLE-compressed formats |
| GDF | IBM GDDM format | RNL | GTX Runlength |
| GIF | CompuServe | SBP | IBM StoryBoard |
| GP4 | CALS Group IV - ITU Group IV | SGI | Silicon Graphics RGB |
| GX1 | Show Partner | SUN | Sun |
| GX2 | Show Partner | TGA | Targa |
| ICA | IBM IOCA (see MO:DCA) | TIF | TIFF |
| ICO | Windows icon | WPG | WordPerfect image |
| IFF | Amiga ILBM | XBM | X Window bitmap |
| IGF | Inset Systems (HiJaak) | XPM | X Window pixelmap |
| IMG | GEM Paint | XWD | X Window dump |

表6-12@ 矢量图格式(vector graphics formats)

|  |  |  |  |
| --- | --- | --- | --- |
| 后缀 | 文件名称 | 后缀 | 文件名称 |
| 3DS | 3D Studio | GEM | GEM proprietary |
| 906 | Calcomp plotter | G4 | GTX RasterCAD - scanned images into vectors for AutoCAD |
| AI | Adobe Illustrator | IGF | Inset Systems (HiJaak) |
| CAL | CALS subset of CGM | IGS | IGES |
| CDR | CorelDRAW | MCS | MathCAD |
| CGM | Computer Graphics Metafile | MET | OS/2 metafile |
| CH3 | Harvard Graphics chart | MRK | Informative Graphics markup file |
| CLP | Windows clipboard | P10 | Tektronix plotter (PLOT10) |
| CMX | Corel Metafile Exchange | PCL | HP LaserJet |
| DG | Autotrol | PCT | Macintosh PICT drawings |
| DGN | Intergraph drawing format | PDW | HiJaak |
| DRW | Micrografx Designer 2.x, 3.x | PGL | HP plotter |
| DS4 | Micrografx Designer 4.x | PIC | Variety of picture formats |
| DSF | Micrografx Designer 6.x | PIX | Inset Systems (HiJaak) |
| DXF | AutoCAD | PLT | HPGL Plot File (HPGL2 has raster format) |
| DWG | AutoCAD | PS | PostScript Level 2 |
| EMF | Enhanced metafile | RLC | Image Systems "CAD Overlay ESP" vector files overlaid onto raster images |
| EPS | Encapsulated PostScript | SSK | SmartSketch |
| ESI | Esri plot file (GIS mapping) | WMF | Windows Metafile |
| FMV | FrameMaker | WPG | WordPerfect graphics |
| GCA | IBM GOCA | WRL | VRML |
|  |  | 后缀 | 文件名称 |

<http://blog.csdn.net/zhang_dashuai/article/details/48173663>

1. 隐写术入门篇——bmp：<http://blog.csdn.net/zhang_dashuai/article/details/48173793>
2. 隐写术入门篇——jpeg：<http://blog.csdn.net/zhang_dashuai/article/details/48173737>
3. 隐写术入门篇——png：<http://blog.csdn.net/zhang_dashuai/article/details/48173673>
4. 隐写术入门篇——GIF：<http://blog.csdn.net/zhang_dashuai/article/details/48173783>

#### 7.2.2信息来源：

<https://baike.baidu.com/item/%E9%9A%90%E5%86%99%E6%9C%AF/9851084?fr=aladdin>

<https://zhuanlan.zhihu.com/p/27915431>

<http://blog.csdn.net/fengshenyue/article/details/51638986>

<http://bobao.360.cn/learning/detail/243.html>