用图片隐藏信息的技术实现

1、图片隐藏信息的用途

图片可以用来隐藏加密的信息。关于加密的用途及重要性，俺在“文件加密的扫盲介绍”中，已经强调过加密性。用图片来隐藏加密信息，除了具有加密的效果，还具有很大的欺骗性——因为外人难以知道一张图片是否包含有加密信息。

2、准备工作——先压缩

下面，俺会介绍几种不同的隐藏方式。在动手之前，先说一下准备工作——把要隐藏得文件先用压缩工具（比如 7zip 或 WinRAR）压缩一下。

压缩有如下几个好处：

优点 1：如果你要隐藏的文件是文本格式或者 Office 格式，它内部的内容是明码的。如果里面包含敏感词，在通过网络传输时，会遭遇敏感词过滤。而压缩后的文件，原有的内容已经变得面目全非，可以规律敏感词过滤。

优点 2：压缩之后，体积变小，有利于增加隐蔽性。因此，应尽量使用"最大压缩"的选项。

优点 3：对于后面介绍的 2 种方法（尾部追加法、内容覆盖法），如果你隐藏的文件是压缩格式的，到时候提取信息会很简便——直接用压缩工具来解压，即可。

3、尾部追加法

先介绍最简单的一种方法。

3.1 技术原理

顾名思义，"尾部追加法"就是把要隐藏的文件追加到图片尾部。这种方法不会破坏图片原有的任何数据，因此，图片看起来和原来一模一样。

3.2 隐藏信息的步骤

隐藏的过程很简单，用 Windows 内置的文件拷贝命令，即可完成。假设你的图片文件叫 A.JPG，需要隐藏的压缩文件叫 B.ZIP，那你只需要执行如下命令，就可以把两个文件合并成一个新文件。

copy /b A.JPG + B.zip C.JPG

执行完如上命令，即可得到一个新的图片文件 C.JPG。这个图片文件的大小是前两者的总和。你可以用各种看图工具来打开 C.JPG，不会看到什么异常。

3.3 提取信息的步骤

由于你追加的是压缩文件，提取的时候就简单了——只要用压缩工具打开 C.JPG，就可以直接看到压缩包里面的内容了。

优点：

1、制作简单，一条 copy 命令就可以搞定；如果隐藏的是压缩文件，提取的过程也很简单。

2、用看图工具看生成的新文件，还是跟原来一样。

3、隐藏的文件，大小不受限制。比如，你可以在一张 100K 的图片尾部，追加 200K 的隐藏数据。

缺点

1、由于隐藏的文件附加在尾部。当你把这个新的图片文件上传到某些贴图的网站，（假如这个网站对图片格式的校验比较严格）它有可能会发现图片尾部

有多余的数据，并且会把这个多余的数据丢弃掉。

2、追加后，图片的文件尺寸变大了。如果你追加的文件太大，容易被发现破绽。

比方说，一张 640*480 的 JPEG 图片，大小竟然有好几兆，对于有经验的 IT 技术人员，一下子就会觉得有猫腻。

4、内容覆盖法

说完尾部追加的办法，再来介绍内容覆盖的办法。

4.1 技术原理

通常，图片文件都有包含 2 部分：文件头和数据区。而"内容覆盖法"，就是把要隐藏的文件，直接覆盖 到图片文件的数据区 的尾部 。比方说，某图片有 100K，其中文件头占 1K，那么，数据区就是 99K。也就是说，最多只能隐藏 99K 的文件。

切记：覆盖的时候，千万不可破坏文件头 。文件头一旦破坏，这个图片文件就不再是一个合法的图片文件了。

使用这种方法，对图片文件的格式，是有讲究的——最好用 24 位色的 BMP 格式 。一来，BMP 格式本身比较简单，数据区随便覆盖，问题不大；二来，24 位色的 BMP 相对其它的格式 BMP，文件尺寸更大，可以隐藏更多内容。

4.2 隐藏信息的步骤

用这个招数来隐藏信息，稍微有点麻烦，需要借助一些小工具。对于这种简单的活计，俺通常用 Python 脚本来搞定。以下是俺写的一个简单 Python 脚本。你的电脑中如果有 Python 环境，可以直接拿这个脚本去用。

事先声明 :如下代码没有严格计算 BMP 的文件头尺寸 ,俺只是大致预留了 1024 字节，感觉应该够了。

```
import sys
def embed(container_file, data_file, output_file) :
container = open(container_file, "rb").read()
data = open(data_file, "rb").read()
if len(data)+1024 >= len(container) :
print "Not enough space to save", data_file
else :
f = open(output_file, "wb")
f.write(container[ : len(container)-len(data)])
f.write(data)
f.close()
if "__main__" == __name__ :
try :
if len(sys.argv) == 4 :
embed(sys.argv[1], sys.argv[2], sys.argv[3])
else :
print "Usage:"
print sys.argv[0], "container data output"
except Exception,err :
print err
```

上述 Python 的代码，很好懂，有编程基础的同学，10 分钟之内就可以用自己熟悉的语言重写一个类似的。

4.3 提取信息的步骤

和前一种方法类似。如果你覆盖的是压缩文件，提取的时候，可以用压缩工具打开图片，就可以直接看到压缩包里面的内容了。

优点：

1、图片的文件尺寸没变。

2、虽然隐藏文件覆盖到数据区，破坏了原图像的内容。但是从格式上来讲，该图片文件的格式还是合法的。因此，你可以把这种图片上传到各种贴图的网站，技术上不会出问题。

3、如果隐藏的是压缩文件，提取的过程很简单。

缺点：

1、由于隐藏的文件覆盖了数据区，因此，图片在显示的时候，会有一块区域变成灰蒙蒙的。

2、隐藏文件的大小，有一定的限制——不能大于图片数据区的尺寸。

3、对图片格式有一定要求。此处再啰嗦一下，建议用 24 位色的 BMP 格式。

5、隐写法

最后，来介绍一种最复杂，但是也最隐蔽的方法——隐写术。

5.1 技术原理

此方法会涉及较深奥的技术领域，俺也就知道个大概。通俗地说：如果把图片的某个像素的颜色，进行微小的调整，肉眼是看不出来的；因此，专门的软件，利用某些高深的算法，就可以在变化的像素中隐藏信息。

有兴趣的同学，可以看"这里 "的介绍；懂洋文的，还可以看更详细的介绍，

在"这里 "。

5.2 隐藏息 / 提取信息的步骤

使用这种方法，你需要用专门的工具来进行信息的隐藏和提取。在进行隐藏时，你除了指定图片文件和被隐藏的文件，还需要设置一个密码。隐写工具会把你的隐藏文件先加密，然后再进行隐写；提取的时候，需要用同一款隐写工具进行提取，并输入同样的密码，才能提取出来。

假如图片文件落入攻击者手中，他必须同时知道 2 个信息（你用哪款隐写工具，你隐写时设置的密码），才有可能破解出隐含的信息。因此，安全性很高。

5.3 相关工具

下面介绍几款工具，大伙儿可以根据自己喜好，挑选一个试试看。

名称	界面	类型
Silent Eye	图形界面	开源软件
Steg Hide	命令行界面	开源软件
Ultima Steganography	图形界面	商业软件

优点：

1、隐蔽性非常好。图片看上去几乎没变（其实是有极其轻微的变化，但是肉眼看不出）。并且，图片文件的大小也没变化。即使是专业人士，也很难判断一张图片是否包含了隐写术的数据。

缺点：

1、隐藏信息和提取信息比较麻烦，需要使用专门的工具。

2、只能隐藏较少的信息。此方法能隐藏的信息量，和图片面积有关，和图片格式无关。比如一张 1600*1200 尺寸的，无论哪种格式，大约只能隐藏几 KB

的数据。

6、结尾

刚才介绍的几个招数，除了可以用于图片文件，也可以用在其它的多媒体文件中（比如：音频文件、视频文件）。有兴趣的网友，可以自个儿研究一下。