

# Misc——文件隐写

主讲人：F0rga1n

# Misc(杂项)

Misc是ctf比赛中的一个重要方向，在国外的比赛中其实又被具体划分为各个小块，包括：

Recon（信息搜集）

主要考察一些获取信息的渠道和一些利用百度、谷歌等搜索引擎的技巧

Encode（编码转换）

主要考察在 CTF 比赛中一些常见的编码形式以及转换的技巧和常见方式

Forensic && Stego（数字取证 && 隐写分析）

隐写取证是 Misc 中最为重要的一块，包括文件分析、隐写、内存镜像分析和流量抓包分析等等，涉及巧妙的编码、隐藏数据、层层嵌套的文件中的文件，灵活利用搜索引擎获取所需要的信息等等。

隐写术通过对原文件或原文本进行修改，使之涵盖了我们要隐藏的信息且不易被察觉  
按照隐藏信息的原数据不同可分成文件隐写和文本隐写两大类型：



# 一.前置

## 1.文件类型和文件扩展名的关系

文件扩展名是标识计算机用何种应用程序打开此文件的标识名，文件扩展名可以与文件实际类型不同，且随意修改不破坏文件本身。

文件的实际类型由文件数据中的文件头标志决定。在每一个文件（包括图片，视频或其他的非ASCII文件）的开头实际上都有一片区域来标识这个文件的实际用法，不同的文件通过文件头来区分文件的类型。有些文件如.jpg文件同样会有固定的文件尾进行标识。如果破坏文件头或文件尾，文件就不能正常打开。

如一些典型的文件头包括：

文件类型	文件头	文件尾	特征
JPEG (jpg)	FFD8FF	FF D9	ÿØÿà
PNG (png)	89504E47	AE 42 60 82	.PNG...
GIF (gif)	47494638	00 3B	GIF89a
ZIP Archive (zip)	504B0304	50 4B	PK..
RAR Archive (rar)	52617221		Rar!...

## 2.查看文件类型的方法

(1).010editor/winhex

(2).file指令

128个常见的文件头信息对照表\_Static-AJ的博客-CSDN博客\_docx文件头信息

<https://blog.csdn.net/ccj2020/article/details/87603903>

文件头文件尾全面总结\_Ahuuua的博客-CSDN博客\_文件头文件尾

<https://blog.csdn.net/Ahuuua/article/details/109165473>

### 3.文件的分离

misc中，经常会涉及到一个文件中包含多个文件的情况，需要将他们分离

工具：binwalk，foremost，dd等

binwalk会自动识别文件中多个文件以及他们的文件类型

binwalk -e 文件名 执行分离

foremost -i 文件名 执行分离

建议都试一试，有的时候会出奇奇怪怪的问题

## 二.图片隐写

jpg文件，正规说法叫JPEG文件，支持有损压缩，不支持透明，不支持动画，非矢量，采用YCrCb色彩模型而并非rgb（所以没有lsb隐写）主要由八个段构成：

### 1.jpg类隐写

#### i. jpg图片结构

misc赛题中经常能遇到文件头损坏，或文件头与后文文件不对应的情况，需要我们手动进行修复

每个段具有相同的结构：

段标识（FF）+段类型（标记码）+段长度+段内容,段长度本身的长度固定为2byte,段长度包括段内容和段长度本身,不包括段标识和段类型。

段类型表：

- 名称 标记码 说明
- SOI D8 文件头
- EOI D9 文件尾
- SOF0 C0 帧开始（标准JPEG）
- SOF1 C1 同上
- DHT C4 定义Huffman表（霍夫曼表）
- SOS DA 扫描行开始
- DQT DB 定义量化表
- DRI DD 定义重新开始间隔
- APP0 E0 定义交换格式和图像识别信息
- COM FE 注释



## ii.exif信息

---

EXIF（可交换图像文件格式）可以用来记录数码照片的属性信息和拍摄数据，EXIF可以被附加在JPEG、TIFF、RIFF等文件中，为其增加有关数码相机拍摄信息的内容。缩略图或图像处理软件的一些版本信息。储存在APP1(0xFFE1)数据区中，接下来两字节保存APP1数据区(即Exif数据区)的大小，接着为Exif Header，固定结构：0x457869660000，后面为Exif的数据

windows系统中查看图片属性就可以看到这些exif信息

---

同样的，在010editor中文件头之后的app1数据区就能看到exif信息

### iii.冗余信息插入

jpg文件中由于没有png的crc冗余检测，所以很容易插入一些信息，如：

#### i.插入尾部注释

直接插入在FF D9之后，不会影响正常的图片观看，同时010editor模板中呈现灰色

#### ii.插入自定义COM注释

插入位置一般在DHT块FF C4之前，以FF FE或者FF FF开头，长度由隐藏信息长度而定

#### iii.插入可以被忽略的标记码

插入方式同上，如：

00

01 \*TEM

d0 \*RST0

dc DNL

ef APP15

#### iv:修改dqt

DQT: Define Quantization Table 标识码为0xdb 接下来两字节表示长度 接下来一字节表示QT设置信息 前4bit为QT号 后4bit为QT精度,0=8bit,否则为16bit 最后是QT信息，长度为64的整数倍，可篡改后面64位的内容。

## iv:特定隐写工具

这些工具都是很成熟的隐写工具，对图片几乎没有任何影响，需要特定工具检查  
常见：silenteye，steghide，JPHide，OutGuess，F5等等

还有些冷门的隐写工具如：JSteg，Invisible Secrets，appendX和Camouflage等

(1).silenteye是一款开源针对jpg，bmp，wav(音频文件)文件的隐写工具，是否需要密码均可具有图形界面，操作简便，010editor中具有一些隐写特征

<https://github.com/achorein/silenteye/>

(2).steghide同样是针对jpg，bmp，wav文件的隐写工具，是否需要密码均可kali自带，可直接使用

使用方法：

```
steghide embed -cf picture.jpg -ef secret.txt
```

然后设置密码。

该命令将文件secret.txt嵌入到封面文件picture.jpg中；

同时在您嵌入您的秘密数据之后，您可以将文件picture.jpg发送给应该收到密码的人员。接收方必须以下列方式使用steghide：

```
steghide extract -sf picture.jpg (-p passwd)
```

运用如上命令，然后输入设置方设置的密码就可以得到隐藏文件；

如果您收到包含嵌入式数据的文件，并且要在提取该文件之前获取相关信息，请使用info命令：

```
steghide info picture.jpg
```

输入密码后就可以得到该文件的相关信息；

下列工具隐写是基于DCT域的JPG图片隐写，因为DCT是一种有损压缩技术，但一般不会影响图像的视觉效果，可以通过这个特性来隐藏信息。在这个隐写类型中常见的隐写方法有JSteg，JPHide，Outguess，F5等等

(3).F5隐写是基于矩阵编码的一种隐写方式，针对jpg文件，**需要密码**

工具：F5-steganography

<https://github.com/matthewgao/F5-steganography>

java Extract “待提取的图片路径” -p passwd

(4).outguess隐写是基于频率转换的隐写方式，是否需要密码均可

kali自带，可直接使用

加密：

outguess -k "my secret key" -d hidden.txt demo.jpg out.jpg

加密之后，demo.jpg会覆盖out.jpg，

hidden.txt中的内容是要隐藏的东西

解密：

outguess -k "my secret key" -r out.jpg hidden.txt

解密之后，解密内容放在hidden.txt中

## 附:特殊隐写检测工具stegdetect

Stegdetect主要用于分析JPEG文件，可以检测到通过JSteg、JPHide、OutGuess、Invisible Secrets、F5、appendX和Camouflage等这些隐写工具隐藏的信息

分析：

Stegdetect.exe -tjopi -s 10 图片名

破解密码：

Stegdetect.exe -r rules.ini -f password.txt -t p 图片名

-t：设置要检测哪些隐写工具（默认检测j opi), 可设置的选项如下：

-j：检测图像中的信息是否是用jsteg嵌入的。

-o：检测图像中的信息是否是用out guess嵌入的。

-p：检测图像中的信息是否是用jphide嵌入的。

-i：检测图像中的信息是否是用 invisible secrets嵌入的

## 2.png类隐写

### i.png文件结构

png文件主要由多个数据块组成

PNG文件结构很简单，最少包含4个数据块。

| PNG标识符 | PNG数据块 (IHDR) | PNG数据块(其他类型数据块) | ... | PNG  
结尾数据块(IEND) |

PNG定义了两类型的数据块，一种是称为关键数据块(critical chunk)，这是标准的数据块，另一种叫做辅助数据块(ancillary chunks)，这是可选的数据块。

关键数据块定义了4个标准数据块，每个PNG文件都必须包含它们，PNG读写软件也都必须要支持这些数据块。虽然PNG文件规范没有要求PNG编译器对可选数据块进行编码和译码，但规范提倡支持可选数据块。

PNG文件中，每个数据块由4个部分组成。

## 文件头数据块IHDR(header chunk)

它包含有PNG文件中存储的图像数据的基本信息，并要作为第一个数据块出现在PNG数据流中，而且一个PNG数据流中只能有一个文件头数据块。

文件头数据块由13字节组成，它的格式如下所示。

(固定) 八个字节89 50 4E 47 0D 0A 1A 0A为png的文件头

(固定) 四个字节00 00 00 0D (即为十进制的13) 代表数据块的长度为13

(固定) 四个字节49 48 44 52 (即为ASCII码的IHDR) 是文件头数据块的标示 (IDCH)

(可变) 13位数据块 (IHDR)

前四个字节代表该图片的宽

后四个字节代表该图片的高

后五个字节依次为：

Bit depth、ColorType、Compression method、Filter method、Interlace method

ii.冗余数据

iii.宽高隐藏

通过修改图片的长宽查看被遮挡的字样

往往misc题目中会给出一些看起来缺少一部分的png图片，需要我们去补全

对于png文件，宽高数据在IHDR数据块的开头，各四字节

宽高修改过度会造成图片损坏花屏

同时，高宽与原图不符会出现crc校验错误的问题

crc校验过程可逆，可以使用脚本强行爆破出原图高宽



## iv:IDAT隐写

IDAT 块只有当上一个块充满（正常length最大65524）时，才会继续一个新的块。程序读取图像的时候也会在第一个未满足的块停止（查了下W3C标准，其实是PNG图片在压缩的时候会在最后一个块的标记位标明这是最后一个数据块）。所以如果某一块没有满但后面却还有 IDAT 块则说明后面的块是“假”的。




我们可以在010editor中看到错误的IDAT块然后利用 python zlib 解压多余IDAT块的内容，此时注意剔除长度、数据块类型及末尾的CRC校验值。

## v.lsb隐写

png文件使用rgb颜色系统

LSB即为最低有效位 (Least Significant Bit, lsb)，我们知道，图片中的图像像素一般是由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。

LSB隐写就是修改RGB颜色分量的最低二进制位也就是最低有效位 (LSB)，而人类的眼睛不会注意到这前后的变化，每个像数可以携带3比特的信息。

Color (Green)	Base 10	Binary	Change
	238	11101110	+3
	235	11101011	(base)
	232	11101000	-3

lsb隐写具有多种情况：

可能是部分通道隐写，如仅b0通道隐写

也可能是并非0通道隐写，情况很多

可能是行列顺序不同，或是位平面顺序不同(如rgb变成rbg)

可能是隐写内容不同

如：

隐写文本

隐写文件

隐写图像

```
666c61677b633474 5f64306e745f3169 flag{c4t_d0nt_li
6b655f3173627d00 db6db6495b6d4aa5 ke_lsb}. .m.I[mJ.
```

```
89504e470d0a1a0a 00000000d49484452 .PNG.... .IHDR
0000011800000118 0802000000008ec7e .....~
db00000542494441 54789ceddd416e23 ....BIDA Tx...An#
391000416931ffff b2f707c480932e57 9..Ail.. .....W
cb1157c3564b5682 8702c9f7d7d7d70b ..W.VKV. ....
f837fffd403c027 1012048404012141 .7.....' .....!A
4048101012048404 012141e0cfe167ef @H..... .!A...g.
f77bec39729383e6 c30775788cbbdfca .(.9r... ..ux....
e5ffe5bbb7bcdff9 9f62458280902020 ..... .bE...
2408080902428280 9020202408080902 $....B... $....
```

Blue plane 0



## lsb隐写信道分析工具——stegsolve

图像处理主要是analyse这个模块，主要有这四个功能：

File Format: 文件格式，查看图片的具体信息

Data Extract: 数据抽取，提取图片中隐藏数据

Frame Browser: 帧浏览器，主要是对GIF之类的动图进行分解，动图变成一张张图片

Image Combiner: 拼图，图片拼接

在主界面通过左右箭头可以看到不同信道，不同透明度，不同灰度下的图片

尤其是lsb隐写如果隐写量较大，会在0信道看到非常明显的条状乱码

## lsb隐写爆破工具——zsteg

zsteg是针对png与bmp中的lsb隐写检测工具

可以迅速地检查所有信道组合，所有方向，所有位平面顺序可能的隐写情况

检测所有组合：

```
zsteg <文件名> --all
```

导出内容

```
zsteg -e "检测组合" 文件名 > 文件名
```

如：

```
zsteg -E "b1,bgr,lsb,xy" pcat.png > p.exe
```

# 三.压缩包隐写

## 1.zip伪加密（rar伪加密）

### 原理

ZIP伪加密是在文件头的加密标志位进行修改，进而再次打开文件时被识别为加密压缩包。

ZIP文件主要由三个部分组成：压缩源文件数据区 + 核心目录 + 目录结束标志

#### 无加密

压缩源文件数据区的全局加密应当为00 00

且压缩源文件目录区的全局方式位标记应当为00 00

#### 假加密

压缩源文件数据区的全局加密应当为00 00

且压缩源文件**目录**区的全局方式位标记应当为09 00

#### 真加密

压缩源文件数据区的全局加密应当为09 00

且压缩源文件目录区的全局方式位标记应当为09 00

## 2.zip,rar密码爆破

工具：**ARCHPR**，fcrackzip，rar password recovery等

常见为四位密码爆破

## 3.crc32碰撞

原理：CRC校验实用程序库 在数据存储和数据通讯领域，为了保证数据的正确，就不得不采用检错的手段。在诸多检错手段中，CRC是最著名的一种。CRC的全称是循环冗余校验。

每个文件都有唯一的CRC32值，即便数据中一个bit发生变化，也会导致CRC32值不同。若是知道一段数据的长度和CRC32值，便可穷举数据，与其CRC32对照，以此达到暴力猜解的目的。但通常只适用于较小文本文件。（类似于哈希碰撞的原理）

## 4.明文攻击

得到了加密压缩包中的某个文件，那么就可以通过明文攻击来获取压缩密码

zip使用传统加密方式ZipCrypto Store可以使用明文攻击

RAR 2.0使用AES-128-cbc，无法攻击

要求文件的名称以及内容完全相同，可以通过比较crc校验码来看

使用的工具依然是ARCHPR

我们通过将已经得到的文件使用**相同版本的压缩程序**进行压缩

再使用ARCHPR的明文攻击，就能拿到他的口令以及其余的文件

拿到的文件不得少于12byte，否则会攻击失败