GIF

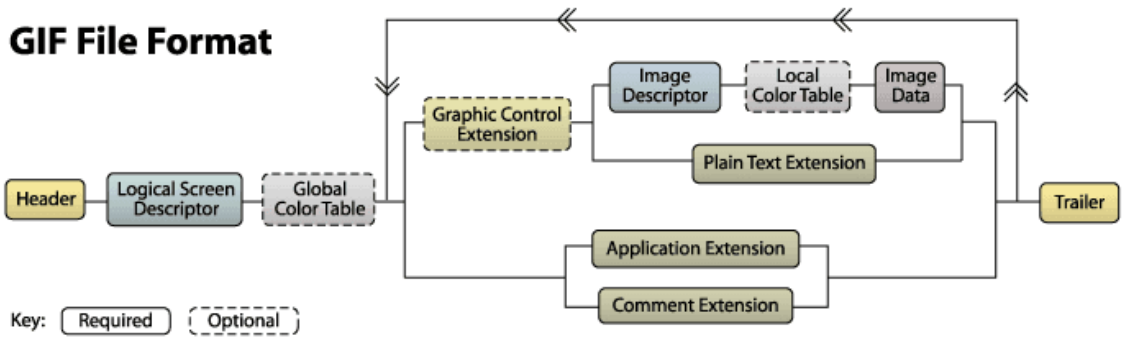
EN | ZH

文件结构

一个 GIF 文件的结构可分为

- 文件头（File Header）
  - GIF 文件署名（Signature）
  - 版本号（Version）
- GIF 数据流（GIF Data Stream）
  - 控制标识符
  - 图象块（Image Block）
  - 其他的一些扩展块
- 文件终结器（Trailer）

下表显示了一个 GIF 文件的组成结构：



中间的那个大块可以被重复任意次

文件头

GIF 署名（Signature）和版本号（Version）。GIF 署名用来确认一个文件是否是 GIF 格式的文件，这一部分由三个字符组成：`GIF`；文件版本号也是由三个字节组成，可以为 `87a` 或 `89a`。

逻辑屏幕标识符（Logical Screen Descriptor）

Logical Screen Descriptor（逻辑屏幕描述符）紧跟在 header 后面。这个块告诉 decoder（解码器）图片需要占用的空间。它的大小固定为 7 个字节，以 canvas width（画布宽度）和 canvas height（画布高度）开始。

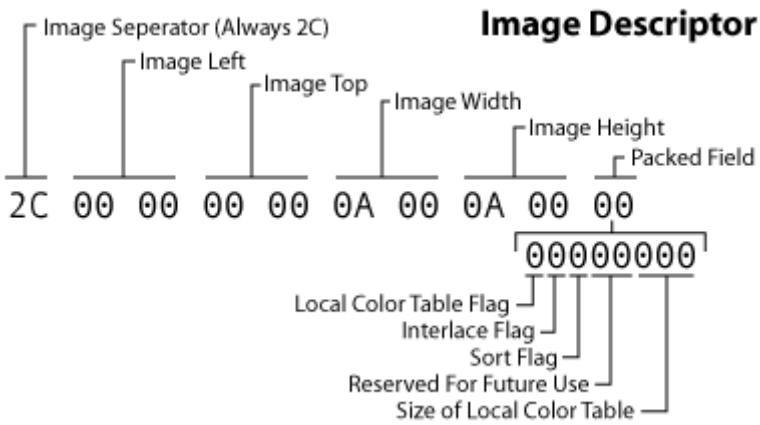
全局颜色列表（Global Color Table）

GIF 格式可以拥有 global color table，或用于针对每个子图片集，提供 local color table。每个 color table 由一个 RGB（就像通常我们见到的（255，0，0）红色 那种）列表组成。

图像标识符（Image Descriptor）

一个 GIF 文件一般包含多个图片。之前的图片渲染模式一般是将多个图片绘制到一个大的（virtual canvas）虚拟画布上，而现在一般将这些图片集用于实现动画。

每个 image 都以一个 image descriptor block（图像描述块）作为开头，这个块固定为 10 字节。



图像数据（Image Data）

终于到了图片数据实际存储的地方。Image Data 是由一系列的输出编码（output codes）构成，它们告诉 decoder（解码器）需要绘制在画布上的每个颜色信息。这些编码以字节码的形式组织在这个块中。

文件终结器（Trailer）

该块为一个单字段块，用来指示该数据流的结束。取固定值 0x3b.

更多参见 gif 格式图片详细解析

空间轴

由于 GIF 的动态特性，由一帧帧的图片构成，所以每一帧的图片，多帧图片间的结合，都成了隐藏信息的一种载体。

对于需要分离的 GIF 文件, 可以使用 `convert` 命令将其每一帧分割开来

`sourceCode shell root in ~/Desktop/tmp λ convert cake.gif cake.png root in ~/Desktop/tmp λ ls cake-0.png cake-1.png cake-2.png cake-3.png cake.gif

```
### 例题

> WDCTF-2017:3-2

打开gif后，思路很清晰，分离每一帧图片后，将起合并得到完整的二维码即可

``` sourceCode python
from  PIL import Image


flag = Image.new("RGB",(450,450))

for i in range(2):
    for j in range(2):
        pot = "cake-{}.png".format(j+i*2)
        potImage = Image.open(pot)
        flag.paste(potImage,(j*225,i*225))
flag.save('./flag.png')
```

扫码后得到一串 16 进制字符串

`03f30d0ab8c1aa5....74080006030908`

开头 `03f3` 为 `pyc` 文件的头，恢复为 `python` 脚本后直接运行得到 flag

时间轴

GIF 文件每一帧间的时间间隔也可以作为信息隐藏的载体。

例如在当时在 XMan 选拔赛出的一题

> XMAN-2017:100.gif

通过 `identify` 命令清晰的打印出每一帧的时间间隔

```
$ identify -format "%s %T \n" 100.gif
0 66
1 66
2 20
3 10
4 20
5 10
6 10
7 20
```
```

```
8 20
9 20
10 20
11 10
12 20
13 20
14 10
15 10
```

推断 `20 & 10` 分别代表 `0 & 1`，提取每一帧间隔并进行转化。

```
$ cat flag|cut -d ' ' -f 2|tr -d '66'|tr -d '\n'|tr -d '0'|tr '2' '0'
01011000010011010100000101001110011111011001110010011011000110101001101110011010
```

最后转 ASCII 码得到 flag。

隐写软件

- F5-steganography

评论