G-101

Write-up

GIFs are used by most of us (especially memers) but do we know its structure ?

You will need to dig deep in the GIF file format structure:

https://www.fileformat.info/format/gif/egff.htm

NB: I will be using 010 Editor to demonstrate the structure.

GIF file always begins with a Header and a Logical Screen Descriptor. A Global Color Table may optionally appear after the Logical Screen Descriptor. Each of these three sections is always found at the same offset from the start of the file. Simply, a GIF is a collection of multiple images. Each image stored in the file contains a Local Image Descriptor, an optional Local Color Table, and a block of image data. The last field in every GIF file is a Terminator character, which indicates the end of the GIF data stream.

What's interesting for us here is the The Logical Screen Descriptor. It contains information describing the screen and color information used to create and display the GIF file image.

For our case we got the Width x Height of Display Screen in Pixels is **769x144** :

Name	Value	Start	Size
struct GIFHEADER GifHeader		0h	6h
struct LOGICALSCREENDESCRIPTOR LogicalScreenDescriptor		6h	7h
ushort Width	769	6h	2h
ushort Height	144	8h	2h
struct LOGICALSCREENDESCRIPTOR_PACKEDFIELDS PackedFields		Ah	1h
UBYTE BackgroundColorIndex	0	Bh	1h
UBYTE PixelAspectRatio	0	Ch	1h
struct GLOBALCOLORTABLE GlobalColorTable		Dh	300h
struct DATA Data		30Dh	8D92h
struct TRAILER Trailer		909Fh	1h

Then if we check the DATA field of our GIF, we will find 3 images :

struct DATA Data		30Dh	8D92h
struct GRAPHICCONTROLEXTENSION GraphicControlExtension[0]		30Dh	8h
struct IMAGEDESCRIPTOR ImageDescriptor[0]		315h	Ah
struct IMAGEDATA ImageData[0]		31Fh	468Fh
struct GRAPHICCONTROLEXTENSION GraphicControlExtension[1]		49AEh	8h
struct IMAGEDESCRIPTOR ImageDescriptor[1]		49B6h	Ah
struct LOCALCOLORTABLE LocalColorTable[0]		49C0h	60h
struct IMAGEDATA ImageData[1]		4A20h	1050h
struct GRAPHICCONTROLEXTENSION GraphicControlExtension[2]		5A70h	8h
struct IMAGEDESCRIPTOR ImageDescriptor[2]		5A78h	Ah
struct LOCALCOLORTABLE LocalColorTable[1]		5A82h	300h
struct IMAGEDATA ImageData[2]		5D82h	331Dh
struct TRAILER Trailer		909Fh	1h

Here, we can 2 paths either we extract the 3 images and start comparing between them or we can keep comparing between them using an editor like 010 Editor.

Each Image will contain mostly 3 big sections : Local Image Descriptor, Local COlor Table and Image Data.

After checking those 3 images, we will notice that there is 1 difference in the ImageDescriptor section, exactly in the fields ImageWidth x ImageHeight. The 2nd image has a Width x Height of **144x769** instead if **769x144** :

struct DATA Data		30Dh	8D92h
▸ struct GRAPHICCONTROLEXTENSION GraphicControlExtension[0]		30Dh	8h
▾ struct IMAGEDESCRIPTOR ImageDescriptor[0]		315h	Ah
UBYTE ImageSeperator	44	315h	1h
ushort ImageLeftPosition	0	316h	2h
ushort ImageTopPosition	0	318h	2h
ushort ImageWidth	769	31Ah	2h
ushort ImageHeight	144	31Ch	2h
▸ struct IMAGEDESCRIPTOR_PACKEDFIELDS PackedFields		31Eh	1h
▸ struct IMAGEDATA ImageData[0]		31Fh	468Fh
▸ struct GRAPHICCONTROLEXTENSION GraphicControlExtension[1]		49AEh	8h
▾ struct IMAGEDESCRIPTOR ImageDescriptor[1]		49B6h	Ah
UBYTE ImageSeperator	44	49B6h	1h
ushort ImageLeftPosition	0	49B7h	2h
ushort ImageTopPosition	0	49B9h	2h
ushort ImageWidth	144	49BBh	2h
ushort ImageHeight	769	49BDh	2h
▸ struct IMAGEDESCRIPTOR_PACKEDFIELDS PackedFields		49BFh	1h
▸ struct LOCALCOLORTABLE LocalColorTable[0]		49C0h	60h
▸ struct IMAGEDATA ImageData[1]		4A20h	1050h
▸ struct GRAPHICCONTROLEXTENSION GraphicControlExtension[2]		5A70h	8h
▾ struct IMAGEDESCRIPTOR ImageDescriptor[2]		5A78h	Ah
UBYTE ImageSeperator	44	5A78h	1h
ushort ImageLeftPosition	0	5A79h	2h
ushort ImageTopPosition	0	5A7Bh	2h
ushort ImageWidth	769	5A7Dh	2h
ushort ImageHeight	144	5A7Fh	2h
▸ struct IMAGEDESCRIPTOR_PACKEDFIELDS PackedFields		5A81h	1h
▸ struct LOCALCOLORTABLE LocalColorTable[1]		5A82h	300h
▸ struct IMAGEDATA ImageData[2]		5D82h	331Dh

Now, it is clear that we need to fix the size of the 2nd Image. We can do that directly with an editor :

struct DATA Data		30
struct GRAPHICCONTROLEXTENSION GraphicControlExtension[0]		30
struct IMAGEDESCRIPTOR ImageDescriptor[0]		31
UBYTE ImageSeperator	44	31
ushort ImageLeftPosition	0	31
ushort ImageTopPosition	0	31
ushort ImageWidth	769	31
ushort ImageHeight	144	31
struct IMAGEDESCRIPTOR_PACKEDFIELDS PackedFields		31
struct IMAGEDATA ImageData[0]		31
struct GRAPHICCONTROLEXTENSION GraphicControlExtension[1]		49
struct IMAGEDESCRIPTOR ImageDescriptor[1]		49
UBYTE ImageSeperator	44	49
ushort ImageLeftPosition	0	49
ushort ImageTopPosition	0	49
ushort ImageWidth	769	49
ushort ImageHeight	144	49
struct IMAGEDESCRIPTOR_PACKEDFIELDS PackedFields		49
struct LOCALCOLORTABLE LocalColorTable[0]		49
struct IMAGEDATA ImageData[1]		4A
struct GRAPHICCONTROLEXTENSION GraphicControlExtension[2]		5A
struct IMAGEDESCRIPTOR ImageDescriptor[2]		5A
UBYTE ImageSeperator	44	5A
ushort ImageLeftPosition	0	5A
ushort ImageTopPosition	0	5A
ushort ImageWidth	769	5A
ushort ImageHeight	144	5A
struct IMAGEDESCRIPTOR_PACKEDFIELDS PackedFields		5A
struct LOCALCOLORTABLE LocalColorTable[1]		5A
struct IMAGEDATA ImageData[2]		5D

If now we save the file and view it, we see fast glimpse of the second image. What we can do is extract the images from GIF using the convert command :

```
convert chall-fixed.gif flag.png
```

This will result in the creation of 3 files : flag-0.png , flag-1.png and flag-2.png.

flag-1.png contains the real flag :

What did we learn from this ? a GIF contains Logical Screen Descriptor which only represents the logical screen (width, height, colors) of what is shown on the whole GIF. Then each Image inside a GIF will have its own Local Image Descriptor which represents the true physical data of the image shown. In our case the 2nd image was inverted and that's why we couldn't see the flag.

Flag

`shellmates{G1F_57RUCTUR3}`