Fiesta ETSIIT 2022

cursed_patrickstar Writeup

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Granada, 11 de mayo de 2022

En este reto se nos entrega una imagen llamada cursed_patrickstar.png la cual contiene la flag.

Para resolver este reto tenemos dos vías principales, una es aplicarle filtros a la imagen hasta que alguno de ellos muestre la flag, mientras que otro es intercambiar la posición de los bits menos significativos por la de los más significativos.

En este Writeup veremos ambos métodos.

Método 1: Empleando Stegsolve

Stegsolve es una sencilla herramienta escrita en java que nos permite, entre otras cosas, aplicar distintos filtros a una imagen. Si la usamos veremos que los planos Red plane 0, Blue plane 0 y Green plane 0 nos dan la flag:

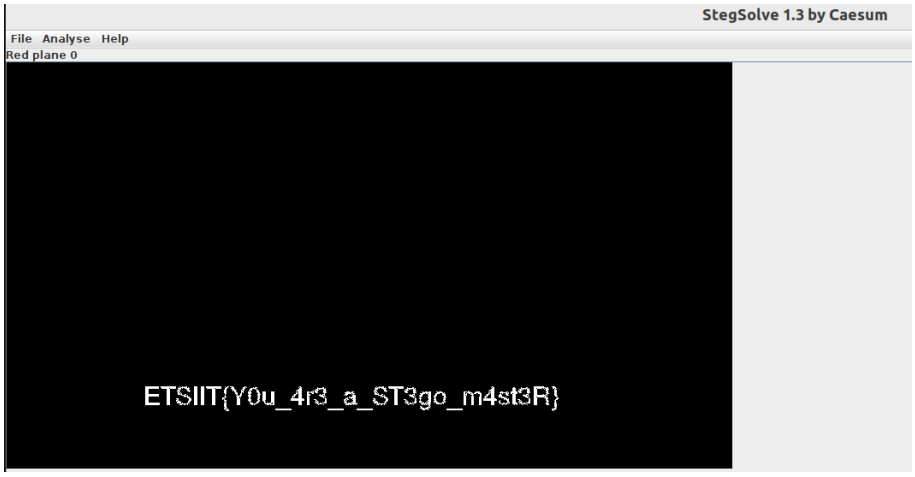

Figura 1: Stegsolve - Red plane 0

Método 2: Intercambiando LSB por MSB

Una de las formas más comunes de esconder una imagen dentro de otra es la siguiente.

Se tienen dos imágenes iniciales, una de "señuelo" y otra con la información a esconder. En este caso, para crear el reto se emplearon estas dos imágenes:

Figura 2: Imagen señuelo

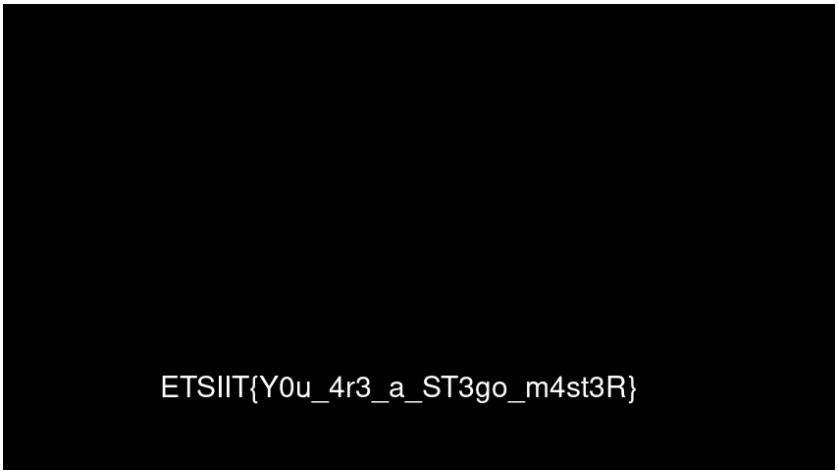

Figura 3: Flag

Partiendo de estas dos imágenes primero se toman los X MSB de cada color de cada pixel de la imagen señuelo, es decir, si el pixel ubicado en image[0][0] es igual a (129, 129, 129), que es igual a (10000001, 10000001, 10000001) y se toman los 7 MSB, estaríamos cogiendo los siguientes bits -> (1000000X, 1000000X, 1000000X), dejando 1 bit de información por cada canal libre. Después cogemos los 8-X MSB de la imagen con la información a ocultar, en este caso la flag, de forma que si flag[0][0] es igual a (0, 0, 0), es decir, (00000000, 00000000, 00000000), nos quedaríamos con los 8-7 MSB, es decir, estaríamos cogiendo un solo bit, y tendríamos esto -> (0XXXXXXX, 0XXXXXXX, 0XXXXXXX). Una vez tenemos esta separación hecha, se crea una imagen nueva cuyos MSB son los mismos que los que hemos cogido de la imagen señuelo, pero sus LSB son los MSB que hemos cogido de la flag, de esta forma, con nuestro ejemplo tendríamos que newImage[0][0] sería igual a (128, 128, 128), es decir, (10000000, 10000000, 10000000).

En este ejemplo que acabamos de ver, se ha codificado con 1 solo bit (al igual que se ha hecho con la imagen del reto), pero dependiendo del caso puede ser que se usen más bits de la imagen a esconder.

Una vez sabemos esto, el proceso para descifrar las imágenes ocultas consiste en sustituir los X LSB de la imagen por sus X MSB y aplicar un padding de 8-X 0's por la derecha, de esta forma, si image[0][0] es igual a (151, 151, 151), que es equivalente a (10010111, 10010111, 10010111), y elegimos descifrar con 1 bit, pasaremos a tener (10000000, 10000000, 10000000). Si hubiésemos elegido descifrar con 4 bits, el resultado habría sido (01110000, 01110000, 01110000).

Ahora que sabemos el proceso para esconder una imagen dentro de otra, así como el proceso inverso, solo nos queda crear un sencillo script en python que nos haga el trabajo:

```
1 def bin_format_decypher(channel_value, bits):
2     channel_value = channel_value[8-bits:]
3 
4     for i in range(8-bits):
5         channel_value += '0'
6 
7     return channel_value
```

Esta función habría que aplicársela a cada canal (RGB) de cada píxel de la imagen y obtendríamos la imagen descifrada.

A continuación se adjuntan las imágenes resultantes de descifrar con 3, 2 y 1 píxeles respectivamente (tener en cuenta que a la hora de cifrar se empleó 1 píxel):

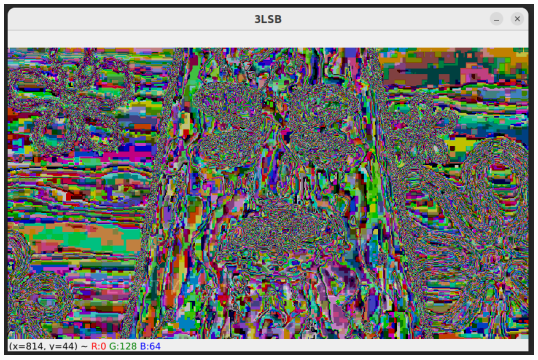

Figura 4: 3 LSB pasan a ser MSB

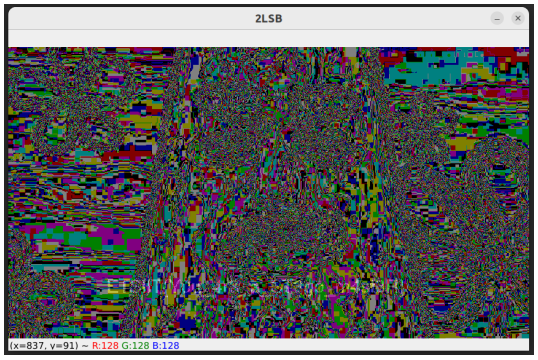

Figura 5: 2 LSB pasan a ser MSB

Figura 6: 1 LSB pasa a ser MSB

Referencias

[1] *Stegsolve tool*. https://github.com/zardus/ctf-tools/blob/master/stegsolve/install.

[2] *Hiding an image inside another*. https://towardsdatascience.com/steganography-hiding-an-image-inside-another-77ca66b2acb1.