

Descripción del Reto:

Un agente de BPRD se enfrenta a nueva amenaza del Infierno
(<https://twitter.com/Hackers4F/status/1109010112172175360>)

Objetivo:

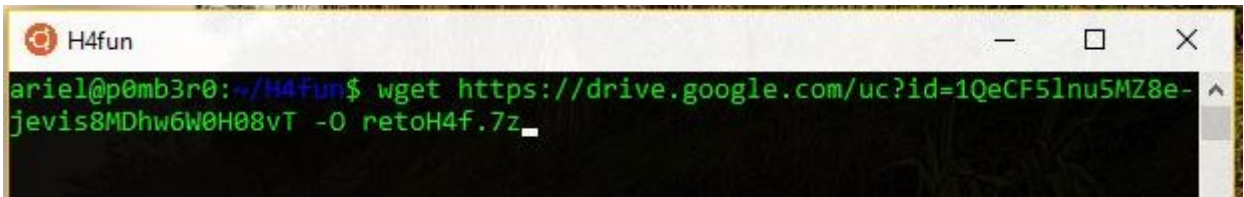
Encontrar la flag: H4F{string}

Herramientas utilizadas:

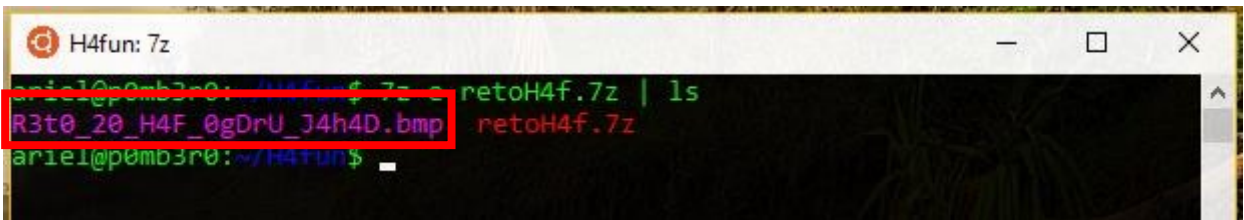
- EasyBmp - (<http://easybmp.sourceforge.net>)
- ❖ EasyBmp (Code Samples) “Steganography” - (<http://easybmp.sourceforge.net/steganography.html>)
- Steganography - (<https://github.com/GarrettBeatty/Steganography>)
- Malbolge - <http://malbolge.doleczek.pl/>
- Base64

Resolución del reto:

- 1- a) Descargamos el archivo “R3t0_20_H4F_0gDrU_J4h4D.7z” de la url
“<https://drive.google.com/uc?id=1QeCF5lnu5MZ8e-jevis8MDhw6W0H08vT>”.



- b) Luego lo descomprimos y el resultado es un archivo (.bmp) R3t0_20_H4F_0gDrU_J4h4D.bmp.



IMAGEN



(R3t0_20_H4F_0gDrU_J4h4D.bmp)

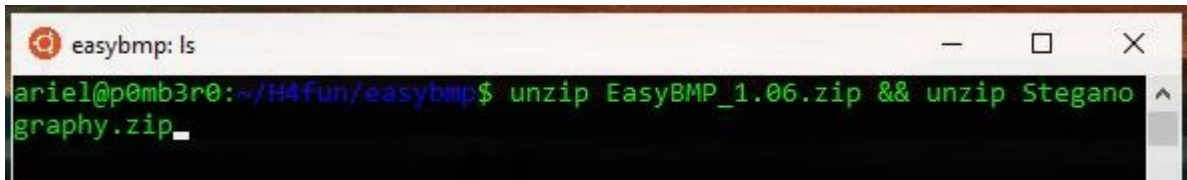
Sha1: c0c34210a3a725a0f9985867a7d2b51800536938

Md5: 3f3df69ea75e556b525dbc797cf1b78b

Cabecera del Archivo: R3t0_20_H4F_0gDrU_J4h4D.bmp: PC bitmap, Windows 3.x format, 1200x800x32

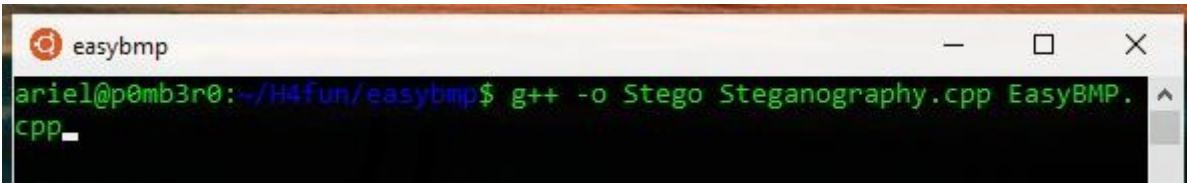
- 2- a) creamos una carpeta “easybmp” y descargamos la tool “EasyBmp”
(http://prdownloads.sourceforge.net/easybmp/EasyBMP_1.06.zip?download), y lo descomprimos.

b) Descargamos la segunda tool en la misma carpeta que creamos anteriormente, vamos a utilizar “Steganography” (<http://easybmp.sourceforge.net/downloads/samples/Steganography.zip>) que se encuentra en el menú **Code Samples** del sitio **EasyBmp**, lo descomprimos.



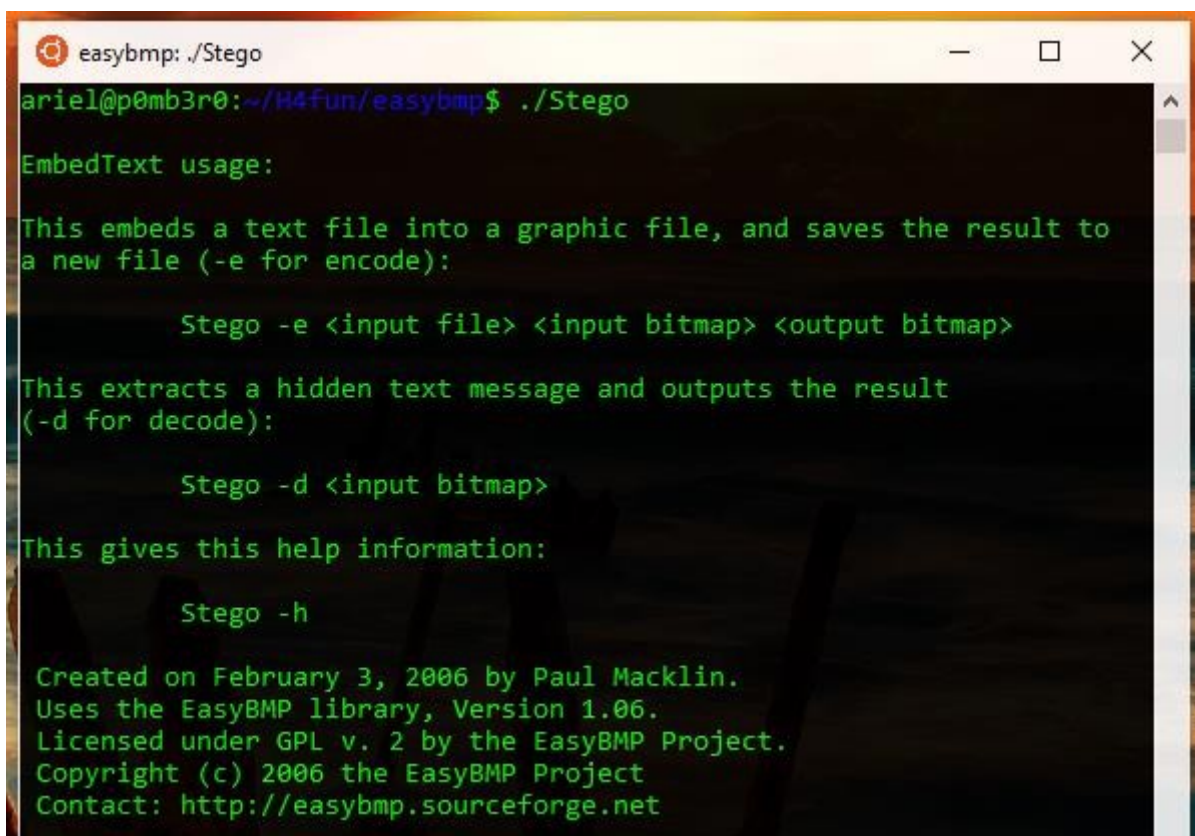
```
easybmp: ls
ariel@p0mb3r0:~/H4fun/easybmp$ unzip EasyBMP_1.06.zip && unzip Stegano
graphy.zip_
```

c) Por ultimo debemos compilar los archivos (“Steganography.cpp” y “EasyBMP.cpp”) para luego ejecutar “Steganography” sobre nuestra imagen “R3t0_20_H4F_0gDrU_J4h4D.bmp”. (~\$ g++ -o Stego Steganography.cpp EasyBMP.cpp)



```
easybmp
ariel@p0mb3r0:~/H4fun/easybmp$ g++ -o Stego Steganography.cpp EasyBMP.
cpp_
```

- 3- a) Copiamos la imagen del reto “R3t0_20_H4F_0gDrU_J4h4D.bmp” a la carpeta “easybmp”.
b) Ejecutamos el binario que creamos anteriormente “~\$./Stego” para ver las utilidades que tiene.



```
easybmp: ./Stego
ariel@p0mb3r0:~/H4fun/easybmp$ ./Stego

EmbedText usage:

This embeds a text file into a graphic file, and saves the result to
a new file (-e for encode):

    Stego -e <input file> <input bitmap> <output bitmap>

This extracts a hidden text message and outputs the result
(-d for decode):

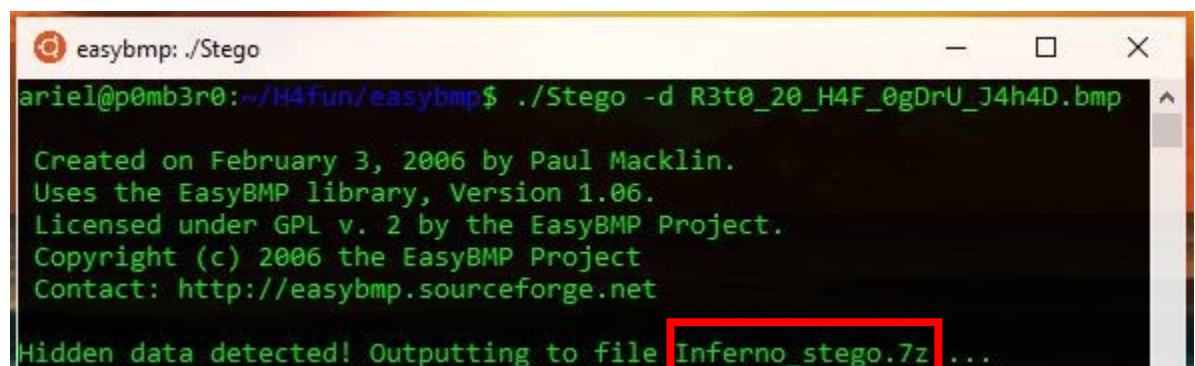
    Stego -d <input bitmap>

This gives this help information:

    Stego -h

Created on February 3, 2006 by Paul Macklin.
Uses the EasyBMP library, Version 1.06.
Licensed under GPL v. 2 by the EasyBMP Project.
Copyright (c) 2006 the EasyBMP Project
Contact: http://easybmp.sourceforge.net
```

c) Ahora utilizaremos la tool para extraer información que pueda llegar a tener nuestra imagen “R3t0_20_H4F_0gDrU_J4h4D.bmp”. (~\$./Stego -d R3t0_20_H4F_0gDrU_J4h4D.bmp)



```
easybmp: ./Stego
ariel@p0mb3r0:~/H4fun/easybmp$ ./Stego -d R3t0_20_H4F_0gDrU_J4h4D.bmp

Created on February 3, 2006 by Paul Macklin.
Uses the EasyBMP library, Version 1.06.
Licensed under GPL v. 2 by the EasyBMP Project.
Copyright (c) 2006 the EasyBMP Project
Contact: http://easybmp.sourceforge.net


Hidden data detected! Outputting to file Inferno_stego.7z ...
```


d) Nos encontramos con un archivo oculto comprimido “Inferno_stego.7z” al cual descomprimos y nos encontramos con otra imagen “Inferno_stego.jpg” al querer pasar nuevamente la tool de EasyBmp nos dice que solo acepta extensiones .bmp.

H4fun: 7z

```
ariel@p0mb3r0:~/H4fun$ 7z o Inferno_stego.7z | ls
easybmp
Inferno_stego.7z
Inferno_stego.7z  K3t0_20_H4F_0gDRO_J4h4D.bmp
retoH4f.7z
```

IMAGEN



(Inferno_stego.jpg)

Sha1: c0d6715450c0b8c7f7d9799b5dc85d67acc12

Md5: a5e585b9a651855ca6e506e95901c600

Cabecera del archivo:
Inferno_stego.jpg: PNG image data, 160x112, 8-bit/color RGB, non-interlaced

easybmp: ./Stego

```
ariel@p0mb3r0:~/H4fun/easybmp$ ./Stego -d Inferno_stego.jpg

Created on February 3, 2006 by Paul Macklin.
Uses the EasyBMP library, Version 1.06.
Licensed under GPL v. 2 by the EasyBMP Project.
Copyright (c) 2006 the EasyBMP Project
Contact: http://easybmp.sourceforge.net

EasyBMP Error: Inferno_stego.jpg is not a Windows BMP file!
Error: File Inferno_stego.jpg not encoded with this program.
```

4- a) Ya que no encontramos ningún otro dato nos descargamos otra tool también de esteganografía “<https://github.com/GarrettBeatty/Steganography>”.
(~\$ git clone <https://github.com/GarrettBeatty/Steganography> && cd Steganography/)

Steganography: git

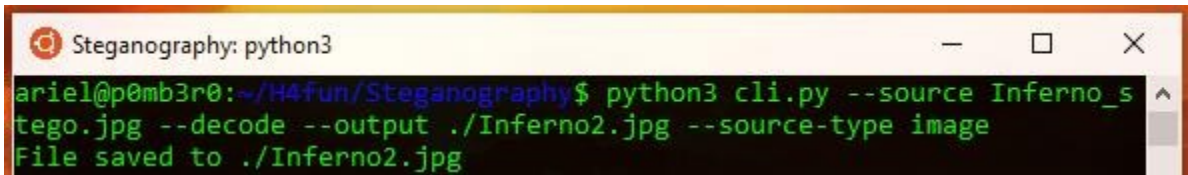
```
ariel@p0mb3r0:~/H4fun$ git clone https://github.com/GarrettBeatty/Steg
anography && cd Steganography/
Clonando en 'Steganography'...
remote: Enumerating objects: 157, done.
remote: Counting objects: 100% (157/157), done.
remote: Compressing objects: 100% (106/106), done.
remote: Total 157 (delta 67), reused 131 (delta 42), pack-reused 0
Recibiendo objetos: 100% (157/157), 8.08 MiB | 1.25 MiB/s, listo.
Resolviendo deltas: 100% (67/67), listo.
ariel@p0mb3r0:~/H4fun/Steganography$
```

b) Al instalar los requerimientos necesarios vemos las utilidades que contiene la tool.

Steganography: python3


```
ariel@p0mb3r0:~/H4fun/Steganography$ python3 cli.py
usage: cli.py [-h] --source SOURCE [--message MESSAGE]
              [--message-type MESSAGE_TYPE] --source-type SOURCE_TYPE
              [--bit-split BIT_SPLIT] [--encode] [--decode] --output O
UTPUT
cli.py: error: the following arguments are required: --source, --sourc
e-type, --output
```

c) Copiamos nuestra imagen “Inferno_stego.jpg” a la carpeta “Steganography” y ejecutamos el siguiente comando “~\$ python3 cli.py --source Inferno_stego.jpg --decode --output ./Inferno2.jpg --source-type image”.



d) Tenemos nuevamente otra imagen “Inferno2.jpg”, al cual pasaremos segunda vez nuestra herramienta para ver qué datos nos arroja. “~\$ python3 cli.py --source Inferno2.jpg --decode --output ./Inferno3.jpg --source-type image”.

IMAGEN



(Inferno2.jpg)


Sha1:
f096c593702d20f64ee24498c085c560f97806e3

Md5:
e050ab08b9745465db0e070daab417b5

Cabecera del archivo:
Inferno2.jpg: PNG image data, 150x89, 8-bit grayscale, non-interlaced

e) Tenemos nuevamente otra imagen “Inferno3.jpg”, pero al cual al analizar la cabecera del archivo nos arroja que es un archivo de texto plano. Si queremos podemos cambiar la extensión en mi caso lo dejo con la misma.

ARCHIVO



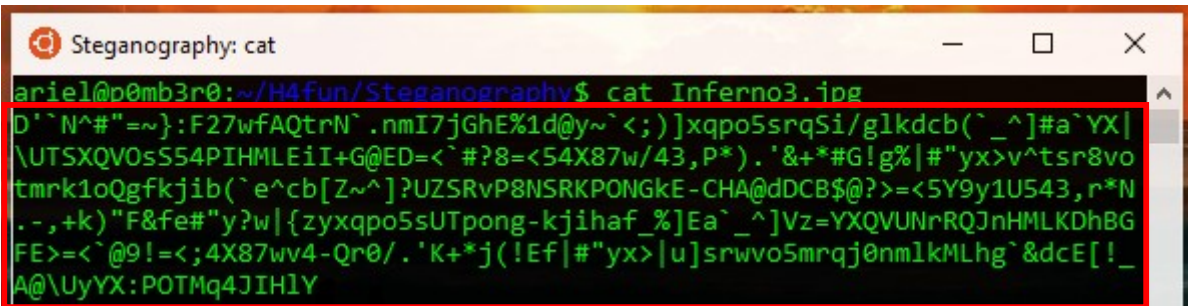
(Inferno3.jpg)

Sha1:
0d8fd0d1383fb9c67051154992bcf7f5931af85a

Md5:
591df15c0e2be3bbbd00007fd0b8a4a5

Cabecera del archivo:
Inferno3.jpg: ASCII text, with very long lines

5- a) Con el comando “cat” podemos ver el contenido de nuestro documento “Inferno3.jpg” al cual nos arroja el siguiente texto,
“D`N^#”=~}:F27wfAQtrN`.nml7jGhE%1d@y~`<;)xqpo5srqSi/glkdcB(`_^]#a`YX|\UTSXQVOsS54PI
HMLEil+G@ED=<`#?8=<54X87w/43,P*).'&+*#G!g%|#`yx>v^tsr8votmrk1oQgfkjib(`e^cb[Z~^]?UZS
RvP8NSRKPONGkE-CHA@dDCB\$@?>=<5Y9y1U543,r*N.-,+k)"F&fe#"y?w|{zyxqpo5sUTpong-
kjihaf_%]Ea`_^]Vz=YXQVUNrRQJnHMLKDhBGFE>=<`@9!<4X87wv4-
Qr0/.`K+*j(!Ef#`"yx>|u]srwvo5mrqj0nmlkMLhg`&dcE[!_A@UyYX:POTMq4JIHIY”.



b) Por lo que se ve a primera vista pensamos que es algún tipo de encriptación pero al leer y analizar el tweet “<https://twitter.com/Hackers4F/status/1109220570493845504>”, y luego buscar en google “lenguajes de programación del infierno” nos encontramos con este lenguaje “Malbolge” (<http://malbolge.doleczek.pl/>). Al abrir el sitio ingresamos el texto que conseguimos anteriormente al ejecutar nos arroja lo que

aparentemente es un texto cifrado en Base64
"U0RSR2UwZ3piR3hDTUhsZlZ6UnpYMGR5TkU1ME0wUjk=".



- 6- Abrimos nuestra terminal y utilizando "base64 --decode" ingresamos nuestro primer texto cifrado "U0RSR2UwZ3piR3hDTUhsZlZ6UnpYMGR5TkU1ME0wUjk=" los cual nos arroja otro texto "SDRGe0gzbgGxCMHlfVzRzX0dyNE50M0R9", pero como leímos anteriormente en el tweet nos dice que hay que pasar dos veces a la estego entonces lo decodificamos de nuevo y nos arroja la flag "H4F{H3l1B0y_W4s_Gr4Nt3D}".

