

Descripción

Nombre: Gr33tings Pr0F3s0R F4lk3n (Related <https://www.imdb.com/title/tt0086567/>)

Fecha de liberación: 8 de Noviembre de 2018

Autor: 1v4n (<https://twitter.com/1r0Dm48Q> // <https://twitter.com/hackers4f>)

Categoría: Misc

Dificultad: Medio-Bajo

NORAD COC analysts detect unusual traffic. The incident is transferred to the FBI, they make arrests to a teenager named David. You can help us discover secrets.

Objetivo

Formato de la flag: H4F{md5}

Método 1:

Herramientas utilizadas

Google Chrome Versión 70.0.3538.102 (Build oficial) (64 bits) <https://www.google.com/chrome/>
<https://www.virustotal.com>

curl 7.61.0 (x86_64-pc-linux-gnu) Release-Date: 2018-07-11 <https://github.com/curl/>

Curses Hexedit 0.9.7 by Adam Rogoyski <http://www.rogoyski.com/adam/programs/hexedit/>

Print or check MD5 (128-bit) checksums <https://www.gnu.org/software/coreutils/md5sum>

GNU strings (GNU Binutils for Debian) 2.31.1

<https://gchq.github.io/CyberChef/>

steghide version 0.5.1 <http://steghide.sourceforge.net/download.php>

<https://morsecode.scphillips.com/translator.html>

<https://github.com/DominicBreuker/stego-toolkit>

<https://www.google.es/maps?source=tldsi&hl=es>

<https://github.com/danielmiessler/SecLists/tree/master/Passwords/Leaked-Databases>

<http://www.whence.com/minimodem/minimodem.1.html>

Resumen:

Comenzamos por visitar el reto y descargamos el archivo adjunto RetoH4F_16_Gr33tings_Pr0F3s0R_F4lk3n [MD5:afe112fe88a60e7129a6b18a279bdce8]https://drive.google.com/file/d/1XgxybBOHreSIzbQTHg6osC-4g_J17oP6/view sin extensión

```
root@kali:~/Desktop/Hackers4F/hbGuadalajara# curl -L "https://docs.google.com/uc?export=download&id=1XgxybBOHreSIzbQTHg6osC-4g_J17oP6" > RetoH4F_16_Gr33tings_Pr0F3s0R_F4lk3n
% Total    % Received % Xferd  Average Speed   Time      Time      Time  Current
          Dload  Upload   Total   Spent    Left  Speed
100  388     0  388     0      501       0  --:--:--  --:--:--  --:--:--   501
100 118k  100 118k     0      81388       0  0:00:01  0:00:01  --:--:-- 1034k
root@kali:~/Desktop/Hackers4F/hbGuadalajara# md5sum RetoH4F_16_Gr33tings_Pr0F3s0R_F4lk3n
afe112fe88a60e7129a6b18a279bdce8  RetoH4F_16_Gr33tings_Pr0F3s0R_F4lk3n
```

Lo analizamos en el servicio VirusTotal detectamos datos Exif Metadata y compresión JPEG:

No engines detected this file

File Name: eebab0555cd0db882496f713441748de84ce6efc9c7831515778c09fc9c097
File size: 118.98 KB
Last analysis: 2018-11-08 11:04:31 UTC

Basic Properties

MDS	a0f112fe88a60e7129a6b18a279bdc8
SHA-1	ab5be08768f96c0a656a877795b1156097bcdff99
File Type	unknown
Magic	data
SSDeep	1536:dp1EL+g/G+/ng4TKxQ0/i6H1weKvqH/EZtQqjO/ET5E1CaSOG7iHXPoQMgAyD7Z:9QfG0nTK/H6JfSmTlZyPkg1p
File Size	118.98 KB

File Names

RetoH4F_16_Gr33tings_ProF3s0R_F4lk3n

ExifTool File Metadata

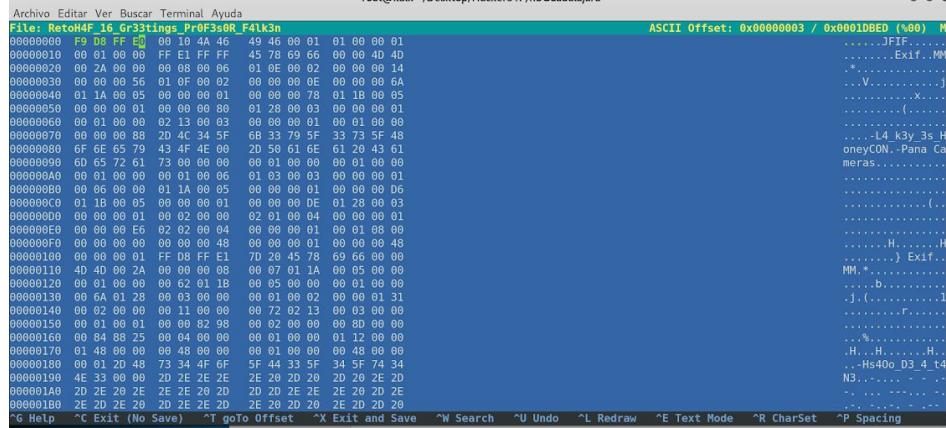
Compression	JPEG (old-style)
ExifByteOrder	Big-endian (Motorola, MM)
ImageDescription	-L4_k3y_3s_HoneyCON
Make	Pana Cameras
ResolutionUnit	None
ThumbnailImage	(Binary data 67584 bytes, use -b option to extract)
ThumbnailLength	67584
ThumbnailOffset	260
Warning	Processing TIFF-like data after unknown 30-byte header
XResolution	1
YCbCrPositioning	Centered
YResolution	1

Desde nuestra consola de Kali Linux vamos arrojarle *file* y *strings* para ver más a fondo el archivo:

```
root@kali:~/Desktop/Hackers4F/hbGuadalajara# file
RetoH4F_16_Gr33tings_ProF3s0R_F4lk3n
RetoH4F_16_Gr33tings_ProF3s0R_F4lk3n: data
root@kali:~/Desktop/Hackers4F/hbGuadalajara# strings
RetoH4F_16_Gr33tings_ProF3s0R_F4lk3n
JFIF
Exif
-L4_k3y_3s_HoneyCON
-Pana Cameras
} Exif
-Hs40o_D3_4_t4N3
.....
...
JFIF
,Exif
TDRfUDRzU19sNF8zbkMwb1RSNHI0U18zb18zTF9NMHYxM19DbDFwXzNTX0RETU1BQUFBX1dIRU5fSVRFV0FT
X0xPQURFRF95XzRudDNzX0QzX1V0MwxejRybGFFUk9DS31vdQ
```

Observamos en la cabecera de su código hexadecimal que ha sido alterada y pasamos a arreglarla por el File Signature correspondiente a la extensión JFIF que es F9 D8 FF E0 con Hexeditor

<https://filesignatures.net/index.php?search=JFIF&mode=EXT>



```

File: RetoH4F_16_Gr33tings_Pr0F3s0R_F4lk3n
ASCII Offset: 0x00000003 / 0x0001DBED (%00) M
00000000 F9 D8 FF E0 00 10 4A 46 49 46 00 01 01 00 00 01
.....JFIF.....
00000010 00 01 00 00 FF E1 FF FF 45 78 69 66 00 00 4D 4D
.....Exif..MM

```

Salvamos los cambios y guardamos el archivo como RetoH4F_16_Gr33tings_Pr0F3s0R_F4lk3n.jpeg

```

root@kali:~/Desktop/Hackers4F/hbGuadalajara# file
RetoH4F_16_Gr33tings_Pr0F3s0R_F4lk3n.jpeg
RetoH4F_16_Gr33tings_Pr0F3s0R_F4lk3n.jpeg: JPEG image data, JFIF standard 1.01,
aspect ratio, density 1x1, segment length 16, Exif Standard: [TIFF image data,
big-endian, direntries=6, description=-L4_k3y_3s_HoneyCON, manufacturer--Pana
Cameras, xresolution=120, yresolution=128, resolutionunit=1], Exif Standard: [],
baseline, precision 8, 630x346, frames 3

```

```

root@kali:~/Desktop/Hackers4F/hbGuadalajara# md5sum
'RetoH4F_16_Gr33tings_Pr0F3s0R_F4lk3n.jpeg'
54cc9bf888206cd863d1fde52e3c4349 RetoH4F_16_Gr33tings_Pr0F3s0R_F4lk3n.jpeg

```

Ahora ya podemos visualizar y abrir el archivo de imagen que esconde el artefacto inicial. En el que visualmente observamos un fotograma de la película de **WarGames (1983)** en el que se ambientará el reto https://es.wikipedia.org/wiki/Juegos_de_guerra



Reconocido el archivo de imagen vamos a explorar sus metadatos con la herramienta *exiftool*, podemos observar datos codificados y una imagen embebida en el atributo “ThumbnailImage”:

```

root@kali:~/Desktop/Hackers4F/hbGuadalajara# exiftool
RetoH4F_16_Gr33tings_Pr0F3s0R_F4lk3n.jpeg
ExifTool Version Number : 11.16
File Name               : RetoH4F_16_Gr33tings_Pr0F3s0R_F4lk3n.jpeg
Directory              : .
File Size               : 119 kB
File Modification Date/Time : 2018:11:18 19:38:04+01:00
File Access Date/Time   : 2018:11:18 19:38:08+01:00
File Inode Change Date/Time : 2018:11:18 19:38:04+01:00
File Permissions        : rw-r--r--
File Type               : JPEG

```

```

File Type Extension : jpg
MIME Type : image/jpeg
JFIF Version : 1.01
Warning : [minor] File contains multi-segment EXIF
Exif Byte Order : Big-endian (Motorola, MM)
Image Description : -L4_k3y_3s_HoneyCON
Make : -Pana Cameras
X Resolution : 1
Y Resolution : 1
Resolution Unit : None
Y Cb Cr Positioning : Centered
Compression : JPEG (old-style)
Thumbnail Offset : 260
Thumbnail Length : 67584
XMP Toolkit : Image::ExifTool 11.16
Event : -46 69 6c 6d
Person In Image : kcaM refinneJ namthgiL divaD
Subject : Blaise de Vigenere
Type : 57 61 72 47 61 6d 65 73 20 69 73 20 61 20 31 39 38
       33
Instructions : -Qcflsc
Image Width : 630
Image Height : 346
Encoding Process : Baseline DCT, Huffman coding
Bits Per Sample : 8
Color Components : 3
Y Cb Cr Sub Sampling : YCbCr4:2:0 (2 2)
Image Size : 630x346
Megapixels : 0.218
Thumbnail Image : (Binary data 67584 bytes, use -b option to
extract)

```

Los datos codificados o cifrados que no esperamos en los metadatos son “L4_k3y_3s_HoneyCON”, “46 69 6c 6d”, “kcaM refinneJ namthgiL divaD”, “Blaise de Vigenere”, “57 61 72 47 61 6d 65 73 20 69 73 20 61 20 31 39 38 33” y “Qcflsc”.

El primer dato nos da una clave que es **HoneyCON**, el segundo dato codificado en *Hexadecimal* que nos encontramos en texto es “*Film*”, el tercero dato es “*David Lightman Jennifer Mack*” en *reverse*, el cuarto es el nombre del criptógrafo al cual se le otorgó el nombre del cifrado *Vigenere*, el quinto dato es otro Hexadecimal “*WarGames is a 1983*” y el sexto más importante es un texto cifrado. Sospechamos que es un Vigenere y que la key que necesitamos es **HoneyCON**, resultando ser el nombre de **Joshua** (nombre de la AI del WOPR) en [https://gchq.github.io/CyberChef/#recipe=Vigen%C3%A8re_Decode\('HoneyCON'\)&input=UWNmbHNj](https://gchq.github.io/CyberChef/#recipe=Vigen%C3%A8re_Decode('HoneyCON')&input=UWNmbHNj)



A partir de aquí nos quedaría descartar que el archivo de imagen no esconde más secretos en forma de esteganografía. Con lo que nos ayudaremos de steghide:

`steghide info RetoH4F_16_Gr33tings_Pr0F3s0R_F4lk3n.jpeg`

```

root@kali:~/Desktop/Hackers4F/hbGuadalajara# steghide info
RetoH4F_16_Gr33tings_Pr0F3s0R_F4lk3n.jpeg
"RetoH4F_16_Gr33tings_Pr0F3s0R_F4lk3n.jpeg":
    formato: jpeg
    capacidad: 2,7 KB
'Intenta informarse sobre los datos adjuntos? (s/n) s
Anotar salvoconducto:
steghide: 'no pude extraer ning'n dato con ese salvoconducto!

```

Probamos la clave Joshua que conseguimos de descifrar el Vigenere y steghide:

steghide extract -sf RetoH4F_16_Gr33tings_Pr0F3s0R_F4lk3n.jpeg:

```

root@kali:~/Desktop/Hackers4F/hbGuadalajara# steghide info
RetoH4F_16_Gr33tings_Pr0F3s0R_F4lk3n.jpeg
"RetoH4F_16_Gr33tings_Pr0F3s0R_F4lk3n.jpeg":
    formato: jpeg
    capacidad: 2,7 KB
'Intenta informarse sobre los datos adjuntos? (s/n) s
Anotar salvoconducto:
archivo adjunto "H4F LSB_text_1":
    tamaño: 95,0 Byte
    encriptado: rijndael-128, cbc
    compactado: si
root@kali:~/Desktop/Hackers4F/hbGuadalajara# steghide extract -sf
RetoH4F_16_Gr33tings_Pr0F3s0R_F4lk3n.jpeg
Anotar salvoconducto:
anote' los datos extraídos e/"H4F LSB text_1".
root@kali:~/Desktop/Hackers4F/hbGuadalajara# cat H4F LSB text_1
VHJZX0g0ckQzc19IM3IZXzFzX04wdGgxbmdfaHR0cHM6Ly93d3cueW91dHVizS5jb20vd2F0Y2g/dj1PcWVG
Q2RYeHF5RQ

```

Obtenemos una cadena en base64 que decodificamos

[https://gchq.github.io/CyberChef/#recipe=From_Base64\('A-Za-z0-9%2B/%3D',true\)&input=VkhKWlgwZzBja1F6Y2w5SU0zSXpYekZ6WDA0d2RHZ3hibWRmYUhSMGNITZMeTkzZDNjdWVXOTFkSFZpWIM1amlyMHZkMkYwWTJnL2RqMVBiV1ZHUTJSWWVIRjVSUQ](https://gchq.github.io/CyberChef/#recipe=From_Base64('A-Za-z0-9%2B/%3D',true)&input=VkhKWlgwZzBja1F6Y2w5SU0zSXpYekZ6WDA0d2RHZ3hibWRmYUhSMGNITZMeTkzZDNjdWVXOTFkSFZpWIM1amlyMHZkMkYwWTJnL2RqMVBiV1ZHUTJSWWVIRjVSUQ) con el resultado siguiente:

TrY_H4rD3r_H3r3_1s_N0th1ng <https://www.youtube.com/watch?v=OqeFCdXxqyE>

Pasamos a extraer la imagen embebida a través exiftool:

exiftool -b -ThumbnailImage RetoH4F_16_Gr33tings_Pr0F3s0R_F4lk3n.jpeg > output1.jpeg

```

root@kali:~/Desktop/Hackers4F/hbGuadalajara# exiftool -b -ThumbnailImage
RetoH4F_16_Gr33tings_Pr0F3s0R_F4lk3n.jpeg > output1.jpeg
Warning: [minor] File contains multi-segment EXIF -
RetoH4F_16_Gr33tings_Pr0F3s0R_F4lk3n.jpeg
root@kali:~/Desktop/Hackers4F/hbGuadalajara# md5sum 'output1.jpeg'
a6433d86bfb7f78353fb8f95dd753c31  output1.jpeg
root@kali:~/Desktop/Hackers4F/hbGuadalajara# file output1.jpeg

```

Con el resultado obtenemos un clip video con música muy acorde para el reto 😊, pensamos que queda camino por andar y la imagen output1.jpeg [MD5: [a6433d86bfb7f78353fb8f95dd753c31](#)] que nos encontramos embebida en “ThumbnailImage”.



Volviendo a repetir el paso anterior vamos a explorar sus metadatos con la herramienta de *exiftool*, podemos observar de nuevo datos codificados y una imagen embebida en el atributo “ThumbnailImage”:

```
Image Height : 400
Encoding Process : Progressive DCT, Huffman coding
Bits Per Sample : 8
Color Components : 3
Y Cb Cr Sub Sampling : YCbCr4:2:0 (2 2)
Image Size : 720x400
Megapixels : 0.288
Thumbnail Image : (Binary data 31602 bytes, use -b option to
extract)
```

Los datos codificados que no los esperamos en los metadatos son

El primer dato nos vuelve a recordar “L4_Fl4G_N0s_3sC0nD3_S3cR3T0s_Y_Su_F0rM4T0_eS_H4F{md5}”, el segundo nos arroja decodificando el Morse “[HTTPS://TWITTER.COM/DAVIDLIGHTMAN83](https://twitter.com/davidlightman83)” el perfil de twitter creado en octubre de 2017 de @davidlightman83 (parece una pista falsa 😊), seguimos con la Longitud de una coordenada GPS con valor **118.326258** y finalmente una codificación Atbash “Sh4lI W3 4 q4M3”



A partir de aquí nos quedaría descartar que el archivo de imagen no esconde más secretos en forma de esteganografía. Con lo que nos ayudaremos nuevamente de steghide
steghide info output1.jpg

```
root@kali:~/Desktop/Hackers4F/hbGuadalajara# steghide info output1.jpeg
"output1.jpeg":
  formato: jpeg
  capacidad: 1,8 KB
'Intenta informarse sobre los datos adjuntos? (s/n) s
Anotar salvoconducto:
steghide: 'no pude extraer ning n dato con ese salvoconducto!'
```

También lo analizamos gracias al script de `check_jpg.sh` de <https://github.com/DominicBreuker/stego-toolkit> sin encontrar pistas de una posible clave para la esteganografía:

```
root@kali:~/Desktop/stego-toolkit/scripts# ./check_jpg.sh output1.jpeg

#####
# JPG CHECKER #
#####
```

Checking file output1.jpeg

```
file output1.jpeg:  
output1.jpeg: JPEG image data, Exif standard: [TIFF image data, big-endian,  
direntries=7, xresolution=98, yresolution=106, resolutionunit=2,  
software=-Hs40o_D3_4_t4N3, copyright=-.... - - - - . . . - - - - . . . - - - ..  
- - . . . . - - - - - - - - - - . . . . - - - - . , GPS-Data], progressive, precision  
...  
#####  
##### binwalk #####  
#####
```

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	JPEG image data, EXIF standard
12	0xC	TIFF image data, big-endian, offset of first image
directory: 8		
434	0x1B2	JPEG image data, JFIF standard 1.01
464	0x1D0	TIFF image data, big-endian, offset of first image
directory: 8		
...		

Si no se logra extraer la imagen embebida, intenta con el comando `exiftool -b -ThumbnailImage output1.jpeg > output2.jpeg`

```
root@kali:~/Desktop/Hackers4F/hbGuadalajara# file output2.jpeg
output2.jpeg: JPEG image data, JFIF standard 1.01, aspect ratio, density 1x1,
segment length 16, Exif Standard: [TIFF image data, big-endian, direntries=6,
description=TDRfUDRzU19sNF8zbkMwblRSNHI0U18zb18zTF9NMHYxM19DbDFwXzNTX0RETY1BQUFBX1dI
RU5fSVRFV0FTX0xPQURFRF9, xresolution=222, yresolution=230, resolutionunit=1,
GPS-Data], baseline, precision 8, 500x274, frames 3
root@kali:~/Desktop/Hackers4F/hbGuadalajara# md5sum 'output2.jpeg'
fb665a8c32385667aa63ca62637810c6  output2.jpeg
```



Repetimos los pasos anteriores volviendo a explorar sus metadatos con la herramienta de exiftool, podemos observar de nuevo datos codificados:

```
root@kali:~/Desktop/Hackers4F/hbGuadalajara# exiftool output2.jpeg
ExifTool Version Number : 11.16
File Name               : output2.jpeg
```

```

Directory : .
File Size : 31 kB
File Modification Date/Time : 2018:11:18 20:44:46+01:00
File Access Date/Time : 2018:11:18 20:44:57+01:00
File Inode Change Date/Time : 2018:11:18 20:44:46+01:00
File Permissions : rw-r--r--
File Type : JPEG
File Type Extension : jpg
MIME Type : image/jpeg
JFIF Version : 1.01
Exif Byte Order : Big-endian (Motorola, MM)
Image Description :

TDRfUDRzU19sNF8zbkMwb1RSNHI0U18zb18zTF9NMHYxM19DbDFwXzNTX0RE TU1BQUFBX1dIRU5fSVRFV0FT
X0xPQRFRF95XzRudDNzX0QzX1V0MWwxejRybGFfUk9DS31vdQ
X Resolution : 1
Y Resolution : 1
Resolution Unit : None
Y Cb Cr Positioning : Centered
GPS Version ID : 2.3.0.0
GPS Latitude : 34 deg 4' 4.56"
XMP Toolkit : Image::ExifTool 11.16
Person In Image : D4v1D_1Ms41_8080_WOPR
Instructions : b7ba59a1bdccf263e2e3a12c5b097d47
Image Width : 500
Image Height : 274
Encoding Process : Baseline DCT, Huffman coding
Bits Per Sample : 8
Color Components : 3
Y Cb Cr Sub Sampling : YCbCr4:4:4 (1 1)
Image Size : 500x274
Megapixels : 0.137

```

Los datos no esperados en los metadatos son:

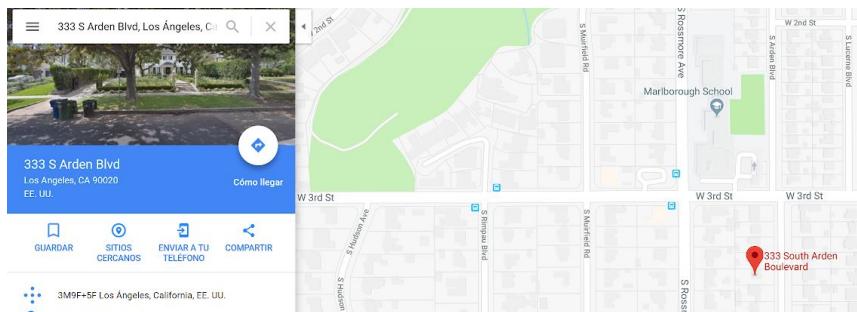
“*TDRfUDRzU19sNF8zbkMwb1RSNHI0U18zb18zTF9NMHYxM19DbDFwXzNTX0RE TU1BQUFBX1dIRU5fSVRFV0FTX0xPQRFRF95XzRudDNzX0QzX1V0MWwxejRybGFfUk9DS31vdQ*”, un Hash MD5 “*b7ba59a1bdccf263e2e3a12c5b097d47*” y “*GPS Latitude 34.067933 degrees*”. El primer dato nos arroja despues de decodificar el base64: “*L4_P4sS_I4_3nCOnTR4r4S_3n_3L_M0v13_Cl1p_3S_DDMMAAAA_WHEN_IT_WAS_LOADED_y_4nt3s_D3_Ut1l1z4rl_a_ROCKyou*” (tenemos que arrojar rockyou.txt primero, posiblemente en la esteganografia) y el segundo lo obtenemos haciendo reverse al MD la URL https://www.youtube.com/watch?v=KXzNo0vR_dU un clip de video del que nos están pidiendo el **DDMMAAAAA** de cuando fue cargando en YouTube. En este caso fue **30072013**.



No nos olvidamos del ultimo dato que es “GPS Latitude **34.067933 degrees**” que con el dato obtenido en la anterior imagen “GPS Longitude **118.326258 degrees**”, utilizamos la herramienta online <https://www.coordenadas-gps.com/convertidor-de-coordenadas-gps> y nos geolocaliza un lugar en China. 😊

Vamos a jugar con los signos (+ y -) de la Longitud y buscamos en **(34.0679329, -118.326258)** y nos geolocaliza el **333**

S Arden Blvd, Los Angeles, CA 90020, EE. UU. localización de la casa de nuestro protagonista dónde se rodó en 1982 la película WarGames. <http://www.themoviedistrict.com/wargames/>



Por último nos quedaría descartar que el archivo de imagen no esconde más secretos en forma de esteganografía.

Con lo que nos ayudaremos nuevamente de steghide

`steghide info output2.jpg`

```
root@kali:~/Desktop/Hackers4F/hbGuadalajara# steghide info output2.jpeg
"output2.jpeg":
  formato: jpeg
  capacidad: 1,5 KB
'Intenta informarse sobre los datos adjuntos? (s/n) s
Anotar salvoconducto:
steghide: 'no pude extraer ning'n dato con ese salvoconducto!
root@kali:~/Desktop/Hackers4F/hbGuadalajara#
```

Recordando el consejo que nos daban de

"**L4_P4s_I4_3nC0nTR4r4S_3n_3L_M0v13_C11p_3S_DDMMAAAA_WHEN_IT_WAS_LOADED_y_4nt3s_D3_Ut1l1z4rla_ROCKyou**" encontramos una pass que es **30072013** pero debemos utilizar rockyou.txt para conseguir el secreto de la esteganografía. Para un ataque de diccionario nos ayudaremos de StegCracker y rockyou aunque podemos tardar mucho tiempo:

```
root@kali:~/Desktop/Hackers4F/hbGuadalajara# stegcracker output2.jpeg rockyou.txt
StegCracker - (https://github.com/Paradoxis/StegCracker)
Copyright (c) 2018 - Luke Paris (Paradoxis)

Attacking file 'output2.jpeg' with wordlist 'rockyou.txt'..
^Ctempted: marine
root@kali:~/Desktop/Hackers4F/hbGuadalajara#
```

Afinamos nuestra búsqueda con el término "wargame", teniendo la premisa de que nuestra clave se encuentra dentro de rockyou.txt. Obtenemos un archivo con una URL acortada que nos dirige a la segunda parte del reto.

```
root@kali:~/Desktop/Hackers4F/hbGuadalajara# cat rockyou.txt | grep wargame > wargame
root@kali:~/Desktop/Hackers4F/hbGuadalajara# stegcracker output2.jpeg wargame
StegCracker - (https://github.com/Paradoxis/StegCracker)
Copyright (c) 2018 - Luke Paris (Paradoxis)

Attacking file 'output2.jpeg' with wordlist 'wargame'..
Successfully cracked file with password: wargame123
Your file has been written to: output2.jpeg.out
root@kali:~/Desktop/Hackers4F/hbGuadalajara# file output2.jpeg.out
```

```
output2.jpeg.out: ASCII text
root@kali:~/Desktop/Hackers4F/hbGuadalajara# cat output2.jpeg.out
https://goo.gl/oAdWFn
```

```
root@kali:~/Desktop/Hackers4F/hbGuadalajara# curl https://goo.gl/oAdWFn
<HTML>
<HEAD>
<TITLE>Moved Permanently</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF" TEXT="#000000">
<H1>Moved Permanently</H1>
The document has moved <A
HREF="https://drive.google.com/open?id=1J9mnpDhz1GMfYoWX3s1j7WaEGUrzygDH">here</A>.
</BODY>
</HTML>
```

Descargamos el archivo de la nube de Drive y utilizamos la pass **30072013** para descomprimirlo:

```
root@kali:~/Desktop/Hackers4F/hbGuadalajara# curl -L
"https://docs.google.com/uc?export=download&id=1J9mnpDhz1GMfYoWX3s1j7WaEGUrzygDH" >
output3
% Total    % Received % Xferd  Average Speed   Time     Time      Time  Current
                                         Dload  Upload   Total   Spent    Left  Speed
100  388     0  388     0      0   671      0  --:--:--  --:--:--  --:--:--   670
100 67160  100 67160     0      0  53174      0  0:00:01  0:00:01  --:--:--  53174
root@kali:~/Desktop/Hackers4F/hbGuadalajara# file output3
output3: 7-zip archive data, version 0.4
root@kali:~/Desktop/Hackers4F/hbGuadalajara# mv output3 output3.7z
root@kali:~/Desktop/Hackers4F/hbGuadalajara# 7z e output3.7z

7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=es_ES.UTF-8,Utf16=on,HugeFiles=on,64 bits,2 CPUs
Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz (406E3),ASM,AES-NI)

Scanning the drive for archives:
1 file, 67160 bytes (66 KiB)

Extracting archive: output3.7z

Enter password (will not be echoed):
--
Path = output3.7z
Type = 7z
Physical Size = 67160
Headers Size = 184
Method = LZMA2:6m 7zAES
Solid = -
Blocks = 1

Everything is Ok
```

```
Size:      4915200
Compressed: 67160
```

Conseguimos descomprimir con éxito y nos arroja un archivo comprimido llamado *N0r4D.tar* que esconde dos archivos uno *hint_modem.wav* y el fin del reto *F14g.rar* que nos solicita una password.

```
root@kali:~/Desktop/Hackers4F/hbGuadalajara# file N0r4D.tar
N0r4D.tar: POSIX tar archive (GNU)
root@kali:~/Desktop/Hackers4F/hbGuadalajara# 7z x N0r4D.tar

7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=es_ES.UTF-8,Utf16=on,HugeFiles=on,64 bits,2 CPUs
Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz (406E3),ASM,AES-NI)

Scanning the drive for archives:
1 file, 4915200 bytes (4800 KiB)

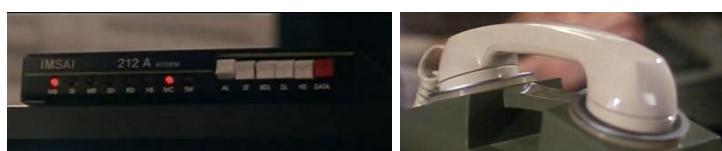
Extracting archive: N0r4D.tar
--
Path = N0r4D.tar
Type = tar
Physical Size = 4915200
Headers Size = 5120
Code Page = UTF-8

Everything is Ok

Folders: 1
Files: 2
Size:      4909659
Compressed: 4915200
root@kali:~/Desktop/Hackers4F/hbGuadalajara# cd N0r4D/
root@kali:~/Desktop/Hackers4F/hbGuadalajara/N0r4D# ls -la
total 4808
drwx----- 2 root root    4096 nov  6 22:59 .
drwxr-xr-x  3 root root    4096 nov 19 18:45 ..
-rw-r--r--  1 root root     175 nov  6 22:41 F14g.rar
-rw-r--r--  1 root root  4909484 nov  6 22:50 hint_modem.wav
```

```
root@kali:~/Desktop/Hackers4F/hbGuadalajara/N0r4D# file hint_modem.wav
hint_modem.wav: RIFF (little-endian) data, WAVE audio, Microsoft PCM, 16 bit, mono
48000 Hz
```

Si reprodujimos el audio podemos descubrir el sonido de un módem como el IMSAI (Cermetek) 212A de David L. en la película de WarGames. Que se ayudaba de un acoplador acústico iba a unos 300 bps en su mejor momento.



Investigando encontramos cómo interpretar el sonido del modem con \$minimodem y pasamos a instalarlo en nuestra máquina de Kali con *sudo apt-get install minimodem* y posteriormente pasamos a escuchar el audio y pasarlo a texto plano:

```
root@kali:~/Desktop/Hackers4F/hbGuadalajara/N0r4D# minimodem --rx 100 -f  
hint_modem.wav  
### CARRIER 100 @ 1250.0 Hz ###  
GREETINGS PROFESSOR FALKEN.  
  
Hello.  
  
HOW ARE YOU FEELING TODAY?  
  
I'm fine. How are you?  
  
EXCELLENT. IT'S BEEN A LONG TIME. CAN YOU EXPLAIN  
THE REMOVAL OF YOUR USER ACCOUNT NUMBER ON 6/23/73?  
  
People sometimes make mistak  
  
YES THEY DO. SHALL WE PLAY A GAME?  
  
Love to. How about Global Thermonuclear War?  
  
WOULDN'T YOU PREFER A GOOD GAME OF CHESS?  
  
Later. Lets play Global Thermonuclear War....  
  
...THANK YOU FOR PLAYING THIS GAME THE PASSWORD TO FIND THE FLAG IS THE NUMBER OF  
THE HOUSE WHERE DAVID LIVES  
  
### NOCARRIER ndata=511 confidence=9.499 ampl=0.936 bps=100.00 (rate perfect) ###
```

La password es el número de la casa donde vive David L. nuestro protagonista y que ya obtuvimos en los pasos anteriores: **333**. Pasamos a descomprimir el archivo Fl4g.rar y obtenemos la deseada **FLAG** en el archivo **Fl4G_R3t0_16**

```
root@kali:~/Desktop/Hackers4F/hbGuadalajara/N0r4D# unrar e Fl4g.rar  
  
UNRAR 5.50 freeware Copyright (c) 1993-2017 Alexander Roshal  
  
Extracting from Fl4g.rar  
  
Enter password (will not be echoed) for Fl4G_R3t0_16:
```

```
The specified password is incorrect.  
Enter password (will not be echoed) for F14G_R3t0_16:  
  
Extracting F14G_R3t0_16  
All OK  
root@kali:~/Desktop/Hackers4F/hbGuadalajara/N0r4D# cat F14G_R3t0_16  
H4F{05dfb621cdca15c190334d3b2eedcceb}
```

Y la solución es **H4F{05dfb621cdca15c190334d3b2eedcceb}**

H4cK_4nD_B33rS_P4s4n_4_D3fC0n_5



Método 2:

Herramientas utilizadas

Google Chrome Versión 70.0.3538.102 (Build oficial) (64 bits) <https://www.google.com/chrome/>

<https://www.virustotal.com>

<https://www.onlinehexeditor.com>

<https://hexed.it>

<https://md5hashing.net>

<http://rumkin.com/tools/cipher/vigenere.php>

<https://www.coordenadas-gps.com/convertidor-de-coordenadas-gps>

<https://console.cloud.google.com>

<http://www.whence.com/minimodem/minimodem.1.html>

<https://www.7-zip.org/>

Resumen

Comenzamos por visitar el reto y descargamos el archivo adjunto *RetoH4F_16_Gr33tings_ProF3sOR_F4lk3n* [MD5:afe112fe88a60e7129a6b18a279bdce8]https://drive.google.com/file/d/1XgxybBOHreSlzbQTHg6osC-4g_J17oP6/view sin extensión. Analizándolo en Virustotal detectamos datos Exif Metadata y compresión JPEG.

No engines detected this file

File Names ◊
RetoH4F_16_Gr33tings_ProF3sOR_F4lk3n

File Properties ◊

Basic Properties ◊

MD5: afe112fe88a60e7129a6b18a279bdce8
SHA-1: ab05e08768f96c0a656a87795b1156097bcdf99
File Type: unknown
Magic: data
SSDeep: 1536:9p1EL+g7G+/ng4TKzQq/f6H1weVqH/EZtQq/0/ETSEICCaSOG7iHXPOQMgAD7z9qGonJTk/s65SmflzyPkglp
File Size: 118.98 KB

File Name: RetoH4F_16_Gr33tings_ProF3sOR_F4lk3n

File Type: JPEG (old-style)

ExifByteOrder: Big-endian (Motorola, MM)

ImageDescription: L4_Personal File Metadata

Make: Pana Cameras

ResolutionUnit: None

ThumbnailImage: (Binary data 67584 bytes, use -b option to extract)

ThumbnailLength: 67584

ThumbnailOffset: 260

Warning: Processing TIFF-like data after unknown 30-byte header

XResolution: 1

YCbCrPositioning: Centered

YResolution: 1

Pasamos a identificar el archivo con la web tool de <https://www.onlinehexeditor.com/> y nos arroja que el *File Signature* ha sido alterado con *00 00 00 00* confirmándose que estamos ante la estructura de un archivo de imagen JPG con ayuda de <https://filesignatures.net>

Extension	Signature	Description
JFIF	FF D8 FF E0	JPEG IMAGE
	ASCII	Size: 4 Bytes Offset: 0 Bytes
JFIF	FF D8 FF E0	JFIF IMAGE FILE - jpeg
	ASCII	Size: 4 Bytes Offset: 0 Bytes

Modificamos el *Magic Number* (*File Signature*) de *00 00 00 00* a *FF D8 FF E0* con la web tool <https://hexed.it/> que es la correspondiente a una imagen JFIF o JPEG y exportamos obteniendo el archivo *RetoH4F_16_Gr33tings_ProF3sOR_F4lk3n.jpeg* [MD5 54cc9bf888206cd863d1fde52e3c4349]

Ahora ya podemos ejecutar y abrir el archivo de imagen que esconde el artefacto inicial. En el que visualmente observamos un fotograma de la película de **WarGames (1983)** en el que se ambientará el reto https://es.wikipedia.org/wiki/Juegos_de_guerra



Con lo que vamos a extraer la información oculta con la web tool de Jeffrey's <http://exif.regex.info/exif.cgi>

Basic Image Information

Target file:	Rent14F_16_Gt3ings_Prof3atR_Fslk3e.jpeg
Description:	L4_k3y_3s_HoneyCON
Camera:	iPan Camera
File:	630 x 346 JPEG 121,838 bytes (1.19 kilobytes)
Color Encoding:	WARNING: No colorspace metadata and no embedded color profile. Windows and Mac web browsers treat colors randomly. Images for the web are most widely viewable when in the sRGB color space and with an embedded color profile. See my Introduction to Digital-Image Color Spaces for more information.

Extracted 720 x 400 66-kilobyte "EXIF_ThumbnailImage" JPG
Displayed here at 62% width (52% the area of the original)



Click image to isolate; click this text to show histogram

Main JPEG image displayed here at 71% width (51% the area of the original)



Click image to isolate; click this text to show histogram

XMP

XMP Toolkit	Image:ExifTool 11.16
Event	-46 69 6c 6d
Person In Image	kcaM refineJ namthgiL divaD
Subject	Blaise de Vigenere
Type	57 61 72 47 61 6d 65 73 20 69 73 20 61 20 31 39 38 33
Instructions	Qcfisc

EXIF

Image Description	L4_k3y_3s_HoneyCON
Make	iPan Camera
Compression	JPEG (old-style)
Resolution	1 pixels/None
Y Cb Cr Positioning	Centered
X Resolution	72
Y Resolution	72
Resolution Unit	inches
Thumbnail Length	67,584
Thumbnail Image	(67,584 bytes binary data)

JFIF

JFIF Version	1.01
Resolution	1 pixels/None

File — basic information derived from the file.

File Type	JPEG
MIME Type	image/jpeg
Exif Byte Order	Big-endian (Motorola, MM)
Encoding Process	Baseline DCT, Huffman coding
Bits Per Sample	8
Color Components	3
File Size	119 kB
File Type Extension	jpg
Image Size	630 x 346
Y Cb Cr Sub Sampling	YCbCr-4:2:0 (2 2)

Composite
This block of data is computed based upon other items. Some of it may be wildly incorrect, especially if the image
[Magpixels] 0.318

Como en el anterior método detectamos que la imagen de la etiqueta "ThumbnailImage" es diferente a la imagen principal y además nos arroja datos que no son los esperados en los metadatos como "L4_k3y_3s_HoneyCON", "46 69 6c 6d", "kcaM refineJ namthgiL divaD", "Blaise de Vigenere", "57 61 72 47 61 6d 65 73 20 69 73 20 61 20 31 39 38 33" y "Qcfisc".

El primer dato nos da una clave que es **HoneyCON**, el segundo dato codificado en *Hexadecimal* que nos encontramos en texto es **"Film"**, el tercer dato es **"David Lightman Jennifer Mack"** en reverse, el cuarto es el nombre del criptógrafo al cual se le otorgó el nombre del cifrado **Vigenere**, el quinto dato es otro Hexadecimal **"WarGames is a 1983"** y el sexto es un texto cifrado. Sospechamos que es un Vigenere y que la key que necesitamos es **HoneyCON**, resultando ser el nombre de **Joshua** (nombre de la AI del WOPR) en:

<http://rumkin.com/tools/cipher/vigenere.php>

A 16th century French diplomat, Blaise de Vigenere, created a very simple cipher that is moderately difficult for any unintended parties to decipher. It is somewhat like a variable Caesar cipher, but the N changed with every letter. You would "encode" your message with a passphrase, and the letters of your passphrase would determine how each letter in the message would be encrypted.

This is the exact opposite of a "Variant Beaufort." To do the variant, just "decode" your plain text to get the cipher text and "encode" the cipher text to get the plain text again.

If you wanted even more security, you can use two passphrases to create a [keyed Vigenere cipher](#), just like the one that stumped cryptologists for years. Again, a pretty simple trick, but it can ensure that your message is even harder to crack.

Recently, a judge created his own "[Smithy Code](#)" in a legal document, but some errors were made. You can see what people consider to be the [correct code](#) with the fixes in upper case.

Decrypt ▾

Passphrase:

Your message:

This is your encoded or decoded text:

A partir de aquí nos quedaría descartar que el archivo de imagen no esconde más secretos en forma de esteganografía. Con lo que nos ayudaremos de <https://futureboy.us/stegano> y detectamos que nos guarda un secreto escondido de 2,7 KB:

Steganographic Encoder

This form uses steganography techniques to hide a secret message (or even another file) in a JPEG image, or a WAV or AU audio file. When you submit, you should be prompted to save your modified file.

If the payload is too large, (more than about 10% the size of the image for small images, closer to 20% for larger images) this may fail data in it.

Select a JPEG, AU or WAV file to upload:

Seleccionar archivo Ningún archivo seleccionado

Password (may be blank):

Payload

- Just find capacity of this file
- Text

"steganoimage803.jpg":
 format: jpeg
 capacity: 2.7 KB

File

Pasamos a descubrir el secreto con la password **Joshua** en <https://futureboy.us/stegano/decinput.html>

arrojándonos como resultado la siguiente cadena en base64 que pasamos a decodificar

VHJZX0q0ckQzcl9IM3lzxZxFzX04wdGxzbmdfaHR0cHM6Ly93d3cueW91dHVizS5ib20vd2F0Y2g/dj1PcWVGQ2RyelHF5R

[Q > TrY H4rD3r H3r3 1s N0th1ng](https://www.youtube.com/watch?v=OgeFCdXxgqE) <https://www.youtube.com/watch?v=OgeFCdXxgqE>

Steganographic Decoder

This form decodes the payload that was hidden in a JPEG image or a WAV or AU audio file using the [encoder form](#). When you submit, you will help you guess at what the payload is and its file type.

Select a JPEG, WAV, or All file to decode:

Seleccionar archivo RetoH4F_16...F4lk3n.jpeg

Password (may be blank):

Joshua View raw output as MIME-type text/plain

- Guess
- Prom

Con el resultado que obtenemos, un clip video con música muy acorde para el reto 😊, pensamos que queda camino por andar y pasamos a extraer la imagen `54cc9bf888206cd863d1fde52e3c4349.jpg` [MD5: `a6433d86fbh7f78353fb8f95dd753c31`], que nos encontramos embebida en "*ThumbnailImage*".

Volviendo a repetir los pasos anteriores en la web tool <http://exif.regex.info/exif.cgi> obtenemos de nuevo otra imagen embebida y datos codificados:

Basic Image Information	
Target file:	54c9f58830c0d63fdfe57e3c4349.jpg
Copyright:	 CC BY-NC-SA
File:	720 x 400 JPEG 67,584 bytes (66 kilobytes)
Color Encoding:	WARNING: No color-space metadata and no embedded-color profile; Windows and Mac web browsers treat colors randomly. Images for the web are most widely viewable when in the sRGB color space and with an embedded color profile. See my Introduction to Digital Image Color Spaces for more information.
	
<small>©  CC BY-NC-SA Extracted 500 x 274 31-kilobyte "DSC_ThumbnailImage" JPG Displayed here at 90% width (39% the area of the original)</small>	
	
<small>©  CC BY-NC-SA Main JPG image displayed here at 62% width (39% the area of the original)</small>	

XMP	
XMP Toolkit	Image::ExifTool 11.16
Instructions	-L4_Fl4G_N0s_3sC0nD3_S3cR3T0s_Y_Su_F0rM4T0_eS_H4F{md5}

EXIF	
Y Cb Cr Positioning	Centered
Copyright	© 2013 - All rights reserved.
GPS Version ID	2.3.0.0
GPS Longitude	118.326258 degrees
Compression	JPEG (old-style)
Resolution	72 pixels/inch
Software	-Hs400_D3_4_t4N3
Thumbnail Length	31,602
Thumbnail Image	(31,602 bytes binary data)

El primer dato nos vuelve a recordar “L4_Fl4G_N0s_3sC0nD3_S3cR3T0s_Y_Su_F0rM4T0_eS_H4F{md5}”, el segundo nos arroja decodificando el Morse “[HTTPS://TWITTER.COM/DAVIDLIGHTMAN83](https://twitter.com/davidlightman83)” el perfil de twitter creado en octubre de 2017 de @davidlightman83 (parece una pista falsa 😅), seguimos con la Longitud de una coordenada GPS con valor **118.326258** y finalmente una codificación Atbash “Sh4L1_W3_4_g4M3”



De nuevo a partir de aquí nos quedaría descartar que el archivo de imagen no esconde más secretos en forma de esteganografía. Que vuelve a ser afirmativo.

Steganographic Encoder

This form uses steganography techniques to hide a secret message (or even another file) in a JPEG image, or a WAV or AU audio file. The submit, you should be prompted to save your modified file.

If the payload is too large, (more than about 10% the size of the image for small images, closer to 20% for larger images) this may fail silent data in it.

Select a JPEG, AU or WAV file to upload:
 54cc5d6f68820...e3c4349.jpg

Password (may be blank):

Payload (select the appropriate radio button to either enter payload text directly or upload a file):

* Just find capacity of this file
 Text

File payload: Ningún archivo seleccionado

"steganoin10736.jpg":
format: jpeg
capacity: 1.8 KB

Probamos descubrir el secreto sin password en <https://futureboy.us/stegano/decinput.html> pero esta vez sin éxito. Probamos con posibles contraseñas comunes pero no conseguimos nada. Por lo tanto seguiremos adelante sin atascarse ya que no encontramos pistas de un posible ataque de diccionario.

Pasamos, de nuevo, a extraer la imagen *a6433d86fb7f78353fb8f95dd753c31.jpg* [MD5: fb665a8c32385667aa63ca62637810c6] que nos encontramos en “*ThumbnailImage*”. Y volviendo a repetir los pasos anteriores en la web tool <http://exif.regex.info/exif.cgi> sin obtener finalmente ninguna imagen embebida pero sí con datos codificados:

Basic Image Information

Target file: a6434860fb7778353fb8f95dd753c31.jpg

Description:	TDRfUDRzU19sNF8zbkMwbIRSNNH0U18zbI8zTF9NMHYxM19DbDFwXzNTXORETU1BQUFBX1dIRU5fsVRfV0FTX0xPQ
File:	500 × 274 JPEG 31,602 bytes (31 kilobytes)
Color Encoding:	WARNING: No color-space metadata and no embedded color profile. Windows and Mac web browsers treat colors randomly. Images for the web are most widely viewable when in the sRGB color space and with an embedded color profile. See my Introduction to Digital Image Color Spaces for more information.



Main JPG image displayed here at 90% width (81% the area of the original)

XMP

XMP Toolkit	Image:ExifTool 11.16
Person In Image	D4+D_1M64J_8080_WOPR
Instructions	b7ba59a1bdccf263e2e3a12c5b097d47

EXIF

Image Description	Decoded as: <i>L8_P4sS_I4_3nCOnTR4r4S_3n_3L_M0v13_C1p_3S_DDMMAAAA_WHEN_IT_WAS_LOADED_y_4nt3s_D3_Ut1l1z4rla_ROCKyou</i>
YCbCr Positioning	Centered
GPS Version ID	2.3.0.0
GPS Latitude	34.067933 degrees
Resolution	1 pixels/None

JFIF

JFIF Version	1.01
Resolution	1 pixels/None

File — basic information derived from the file.

File Type	JPEG
MIME Type	image/jpeg
Endianness	Big-endian (Motorola, MM)
Encoding Process	Baseline DCT, Huffman coding
Bits Per Sample	8
Color Components	3
File Size	31 kB
File Type Extension	.jpg
Image Size	500 × 274
YCbCr Sub Sampling	YCbCr4:4:4 (1:1)

Los datos no esperados en los metadatos son:

“TDRfUDRzU19sNF8zbkMwbIRSNNH0U18zbI8zTF9NMHYxM19DbDFwXzNTXORETU1BQUFBX1dIRU5fsVRfV0FTX0xPQURFR95XzRudDNzX0QzX1V0MWwxejRybGFFfUk9DS3lvdQ”, un Hash MD5 b7ba59a1bdccf263e2e3a12c5b097d47” y “GPS Latitude **34.067933 degrees**”.

El primer dato nos arroja despues de decodificar el base64:

“*L4_P4sS_I4_3nCOnTR4r4S_3n_3L_M0v13_C1p_3S_DDMMAAAA_WHEN_IT_WAS_LOADED_y_4nt3s_D3_Ut1l1z4rla_ROCKyou*” (tenemos que arrojar rockyou primero, posiblemente en la esteganografía) y el segundo lo obtenemos haciendo reverse al MD5 “https://www.youtube.com/watch?v=kXzNo0vR_dU” un clip de video del que nos están pidiendo el **DDMMAAAA** de cuando fue cargando en YouTube. En este caso fue **30072013**.



No nos olvidamos del último dato que es “GPS Latitude **34.067933 degrees**” que con el dato obtenido en la anterior imagen “GPS Longitude **118.326258 degrees**”, utilizamos la herramienta online

<https://www.coordenadas-gps.com/convertidor-de-coordenadas-gps> y nos geolocaliza un lugar en China. 😊

Vamos a jugar con los signos (+ y -) de la Longitud y buscamos en (34.0679329, -118.326258) y nos geolocaliza el **333 S Arden Blvd, Los Angeles, CA 90020, EE. UU.** localización de la casa de nuestro protagonista dónde se rodó en 1982 la película WarGames. <http://www.themoviedistrict.com/wargames/>

Dirección

GD (grados decimales)*

Latitud

Longitud

[Obtener Dirección](#)

[Ver el mapa](#)

De nuevo a partir de aquí nos quedaría descartar que el archivo de imagen no esconde más secretos en forma de esteganografía. Que vuelve a ser afirmativo.

Steganographic Encoder

This form uses steganography techniques to hide a secret message (or even another file) in a JPEG image, or a WAV or AU to a casual observer. Once you submit, you should be prompted to save your modified file.

If the payload is too large, (more than about 10% the size of the image for small images, closer to 20% for larger images) this below before embedding data in it.

Select a JPEG, AU or WAV file to upload:

Password (may be blank):

Payload (select the appropriate radio button to either enter payload text directly or upload a file):

Just find capacity of this file
 Text

File payload:

"steganoin13196.jpg":
 format: jpeg
 capacity: 1.5 KB

Como nos aconsejaron anteriormente que utilizaramos el diccionario rockyou (

<https://www.scrapmaker.com/data/wordlists/dictionaries/rockyou.txt>)

"L4_P4sS_I4_3nCOnTR4r4S_3n_3L_M0v13_C1p_3S_DDMMAAAA_WHEN_IT_WAS_LOADED_y_4nt3s_D3_Ut1l1z4rla_ROCKyou". Filtramos el diccionario con el término "wargame" obtendremos que **wargame123** es la password que nos revelará la esteganografía una URL acortada que nos dirigirá a la segunda parte del reto: <https://goo.gl/oAdWFN> > Reto_16H4F_HB_AFS101.7z [MD5: 71ff799128e00e1a4a4b04a3a6d456df]

El archivo comprimido nos solicita una contraseña que obtuvimos en la última imagen **30072013**, y pasamos a obtener otro archivo comprimido **N0r4D.tar** [MD5: d7dff2e29a49f86a441b7381214105db] que descomprimimos y obtenemos finalmente un archivo de audio **hint_modem.wav** [MD5: adc9599695c1ae757218525824ba5963] y el archivo comprimido **F14g.rar** [edbe186b80fa0708a85cf1bee0a3ab9b] que nos pide una password para conseguir nuestra FLAG.

Investigando encontramos cómo interpretar el sonido del modem con \$minimodem (<http://www.whence.com/minimodem/minimodem.1.html>) para ello necesitaremos una consola online en <https://console.cloud.google.com> y con los siguientes comandos:

```
sudo apt-get install minimodem

minimodem --rx 100 -f hint_modem.wav
### CARRIER 100 @ 1250.0 Hz ####
GREETINGS PROFESSOR FALKEN.
Hello.
```

HOW ARE YOU FEELING TODAY?

I'm fine. How are you?

EXCELLENT. IT'S BEEN A LONG TIME. CAN YOU EXPLAIN
THE REMOVAL OF YOUR USER ACCOUNT NUMBER ON 6/23/73?

People sometimes make mistak

YES THEY DO. SHALL WE PLAY A GAME?

Love to. How about Global Thermonuclear War?

WOULDN'T YOU PREFER A GOOD GAME OF CHESS?

Later. Lets play Global Thermonuclear War....

...THANK YOU FOR PLAYING THIS GAME THE PASSWORD TO FIND THE FLAG IS THE NUMBER OF
THE HOUSE WHERE DAVID LIVES

NOCARRIER ndata=511 confidence=9.499 ampl=0.936 bps=100.00 (rate perfect)

La password es el número de la casa donde vive David L. nuestro protagonista y que ya obtuvimos en los pasos anteriores: **333**. Pasamos a descomprimir el archivo Fl4g.rar y obtenemos la deseada **💪🚩 FLAG** en el archivo **Fl4G_R3t0_16**.

Y la solución es **H4F{05dfb621cdca15c190334d3b2eedcceb}**

H4cK_4nD_B33rS_P4s4n_4_D3fC0n_5



Autor: 1v4n a.k.a. @1r0Dm4480

Twitter: <https://twitter.com/Hackers4f> // <https://twitter.com/1r0Dm4480>