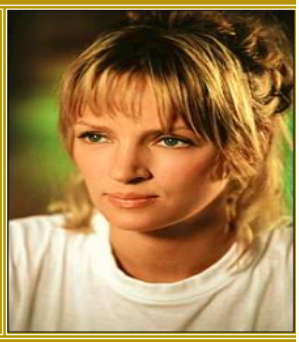
Una-al-mes: Misión#006 - Hispasec

Misión#006

Información personal:

Nombre: Beatrix Michelle Kiddo
Año de nacimiento: 1976
Trabajo: Ex Asesina
Afiliación: Antiguamente en 'Deadly Viper Assasination Squad'



Misión:

Nivel: Fácil

Introducción:

La flag escondida en esta prueba te va a dar a escoger entre dos opciones. Esperemos que escojas bien, sino vas a recibir las consecuencias...

Información adicional:

URL conseguida: goo.gl/YUNxSu

Accedemos a la URL que nos facilita el reto:
http://goo.gl/YUNxSu

Tenemos la imagen bill2.jpg para poder ser descargada :

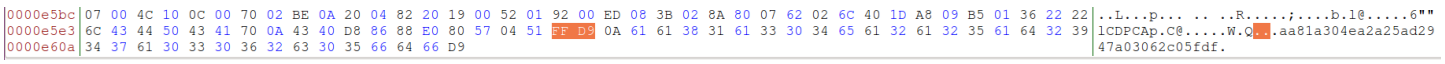
Al intentar abrir la imagen con un editor de imágenes, nos indica que la imagen está corrupta:

```
Corrupt JPEG data: 2 extraneous bytes before marker 0x0a
Unsupported marker type 0x0a
```

También ejecutamos strings sobre la imagen y podemos ver que oculta algo al final de la misma:

```
strings bill2.jpg | tail -n 1
aa81a304ea2a25ad2947a03062c05fdf
```

Con el editor hexadecimal bless analizamos la imagen para tratar de reconstruir un archivo JPG válido. Los archivos JPG finalizan con FF D9, por tanto añado D9 junto al último FF que encuentro en la imagen:



Guardo la imagen como "bill2_reparada.jpg" y ya podemos analizarla sin que nos de un error de JPG corrupto.

Para analizarla con diferentes herramientas de esteganografía hacemos uso de un contenedor de docker llamado DominicBreuker/stego-toolkit en el que vienen instalados gran número de herramientas para analizar imágenes de diferentes formatos.

Para ello lo primero que hacemos es bajarnos la imagen del contendor:
```
sudo docker pull dominicbreuker/stego-toolkit
```

Creamos una carpeta temporal donde guardar la imagen a ser analizada:
```
mkdir -p /tmp/stego/data
cp bill2_reparada.jpg /tmp/stego/data
cd /tmp/stego/
```

Y arrancamos el contenedor con la siguiente orden y poder acceder a él:
```
sudo docker run -it --rm -v $(pwd)/data:/data dominicbreuker/stego-toolkit
/bin/bash
```

Dentro del contenedor analizamos la imagen con un script que automatiza la ejecución de diferentes herramientas como stegoVeritas, strings, exiftool, binwalk, steghide, etc:
```
root@850bc1eb9fca:/data# check_jpg.sh bill2_reparada.jpg
```

Los resultados de analizar la imagen no arrojan nada, por tanto podemos descartar que haya algo más oculto en la imagen.

Centrándonos en la cadena encontrada al final del archivo, podemos deducir que se trata de un código hexadecimal o algún tipo de hash.

Si decodificamos el código hexadecimal a texto no obtenemos resultados positivos: `ª.£.ê*%.)G ObÀ ß`

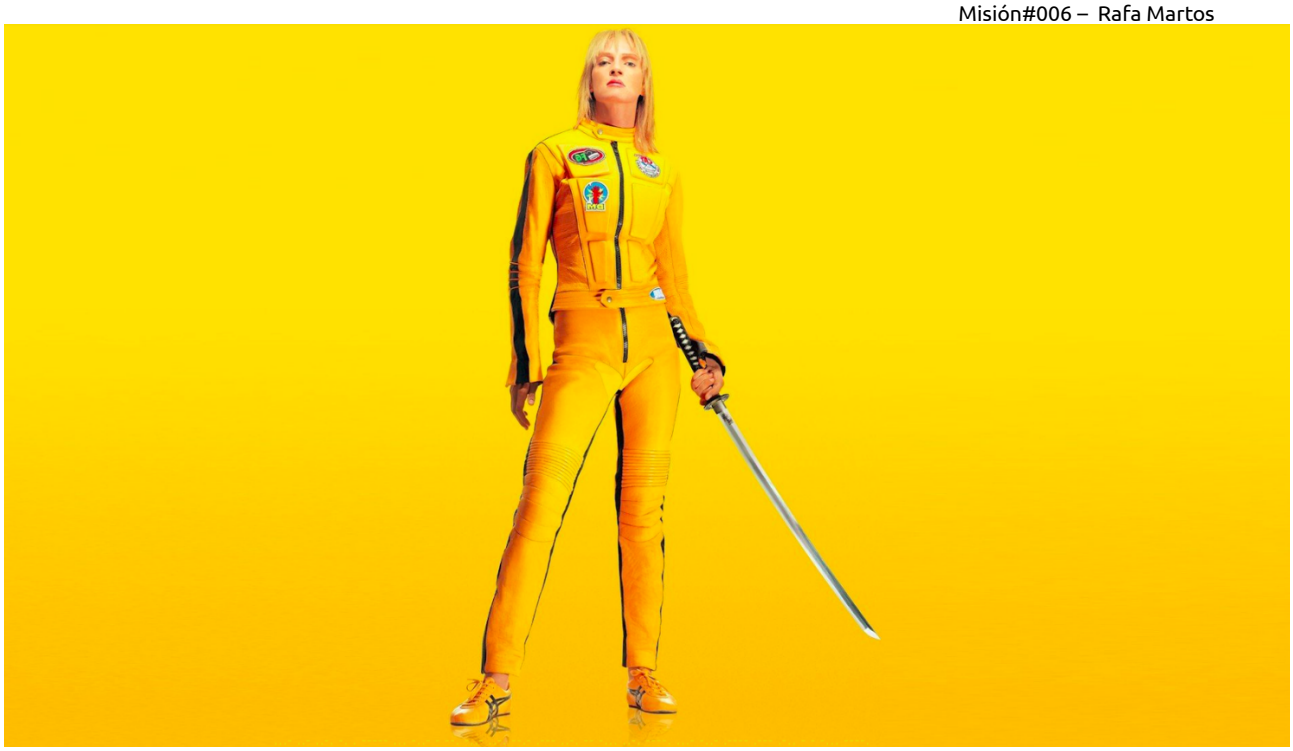
Aplicando algunas transformaciones como rot13, rot47, rot right o left tampoco obtenemos ningún resultado.

La cadena consta de 128 bits, por tanto podemos suponer que se trata de un hash MD5.

Haciendo uso del MD5 Decrypter online https://md5online.org// obtenemos un enlace en el que poder descargar otra imagen:

```
Found : goo.gl/4kxSs7
(hash = aa81a304ea2a25ad2947a03062c05fdf)
```

Es la imagen kill-bill-movie.png



Volvemos a analizar la imagen haciendo uso de las herramientas del contenedor de docker, pero en este caso usamos el script para los archivos .png

```
root@850bc1eb9fca:/data# check_png.sh kill-bill-movie.png
```

Los resultados del análisis tampoco dan resultados positivos.

A simple vista, la imagen tiene al final de la misma unos píxeles en un tono amarillo diferente al del fondo, por tanto con un editor de imágenes procedemos a resaltarlos:

Vemos que se trata de código morse:

..-- ..- ..-. -. . ----- ... -.-- - --. -.-- ..-. --.- .--- ..-. -- -... ..-. ----. ...- -.-- .---- .--- .-.. - - -... ----.

Usando el conversor de código morse a texto online de livephysics.com, obtenemos la siguiente cadena:
`VUFNE0SXTGXFQJFMBF9VX1JLTTB9`

Con un base64 decode, obtenemos lo que parece ser la flag, pero no está del todo correcta:
`UAM.D.LeÅ@.L._U_RKM0}`

Por tanto, cambiando mayúsculas por minúsculas de algunas de las letras conseguimos la combinación correcta para obtener el flag:
`VUFNe0sxTGxfQjFMbF9vX1JlTTB9`
`UAM{K1Ll_B1Ll_o_ReM0}`

El flag es: **UAM{K1Ll_B1Ll_o_ReM0}**

Rafa Martos
@elbuenodefali