

Episodio 3

La bomba. Parte 1

Introducción	5
Desactivando la bomba.	6
Sistema de ficheros linux	7
Reversing del fichero .bomb	17

«Si el líder dice de tal evento esto no ocurrió, pues no ocurrió. Si dice que dos y dos son cinco, pues dos y dos son cinco. Esta perspectiva me preocupa mucho más que las bombas.»

George Orwell. 1984

“Sensual un movimiento sensual
sensual un movimiento muy sexi
sexi un movimiento muy sexi
y aquí se viene el africano con este baile que es una bomba”

King Africa

Challenge

4 Solves



EPISODIO 3

300

Con todo el dinero robado, necesitamos escapar dando una distracción a la policía. Para ello, hace falta encontrar la bomba programada en el firmware del sistema informático. Una vez resuelta, podremos acceder al servidor, donde tras buscar bien, conseguiremos la flag final y escaparemos con el premio.

Info: La flag tiene el formato UAM{md5}

TOP 3: 1. 2. 3.

[View Hint](#)

[View Hint](#)

[firmware.zip](#)

[Flag](#)

[Submit](#)

Introducción

El reto que se nos presenta esta vez está relacionado con la desactivación de una bomba que alguien ha insertado en el firmware, el acceso a un servidor y una última prueba que descubriremos al final. Empezemos.

Desactivando la bomba.

Se nos permite descargar desde la página del reto un fichero zip que , tras descomprimirlo, nos da una carpeta firmware con un único fichero en su interior, backup.raw

Lo primero que hacemos es ver que es ese fichero con la orden file:

```
linux@linux-desktop:~/RetosUAM/firmware$ ls -la
total 523276
drwxr-xr-x 2 linux linux      4096 jul 17 20:22 .
drwxr-xr-x 8 linux linux      4096 jul 17 20:22 ..
-rwxrwxrwx 1 linux linux 535822336 jul 13 21:12 backup.raw
linux@linux-desktop:~/RetosUAM/firmware$ file backup.raw
backup.raw: Linux rev 1.0 ext4 filesystem data,
UUID=046b9ae6-97df-49fd-8785-2c68de053b05 (needs journal recovery)
(extents) (large files) (huge files)
```

Bien, vemos que se trata de un fichero de sistema de ficheros ext4 de linux, procedemos a montarlo para poder acceder a los datos que tiene en su interior. Siempre que vamos a montar un fichero de este tipo es conveniente hacer una copia de él y no montar el fichero original para poder recuperar el estado inicial del mismo en caso de borrar o modificar algo del mismo, esto es muy importante cuando estamos haciendo un forense del mismo ya que requerirá que demostremos como estaba el fichero original. Además es muy conveniente extraer varios hash del mismo (MD5, SHA1,SHA256) que puedan demostrar y, sobre todo, permitir replicar a un perito de la otra parte los procedimientos que realicemos sobre él. En caso de tratarse de un disco duro y no de un fichero realizaremos una copia idéntica “bit a bit”, preferiblemente con una clonadora de discos, para trabajar siempre desde la copia y nunca sobre el original.

En este caso no es necesario ya que siempre podemos restaurar el fichero original desde el fichero zip y, espero, los organizadores del reto no nos llevaran a juicio (espero)

Sistema de ficheros linux

Un ingeniero de Microsoft, uno de Apple y un desarrollador de Linux se encuentran en el baño.

El de Microsoft acaba primero, y a la hora de lavarse las manos lo hace a conciencia, con mucho jabón y agua, y para secarse usa un montón de toallitas de papel, y dice:

- Los de Microsoft, lo que hacemos lo hacemos bien.

El ingeniero de Apple, también se lava a conciencia, pero utiliza muy poca agua y muy poco jabón, y para secarse utiliza una esquinita de una toallita de papel. Cuando acaba dice:

- Los de Apple, además de hacerlo bien, lo optimizamos al máximo.

El programador de Linux, sin lavarse, mira a los otros dos y dice:

- Los de Linux no nos meamos en las manos...

Montamos la imagen de disco con la orden mount creando primero una carpeta que nos servirá de punto de montaje, el comando es muy sencillo, Necesitamos ser root para utilizar el comando (asi lo tengo en mi equipo configurado) y después la orden mount, recibe como primer parámetro el nombre de la imagen raw a montar, backup.raw y como segundo parámetro el directorio donde lo queremos montar, en mi caso mount. La orden queda un poco liosa con el comando y la carpeta llamándose igual pero espero que se entienda bien:

```
linux@linux-desktop:~/RetosUAM/firmware$ mkdir mount  
linux@linux-desktop:~/RetosUAM/firmware$ sudo mount backup.raw mount
```

¡Que diablos, hemos venido a divertirnos, renombremos el fichero!

```
linux@linux-desktop:~/RetosUAM/firmware$ mv backup.raw mount.mount  
linux@linux-desktop:~/RetosUAM/firmware$ sudo mount mount.mount mount
```

Mucho mejor, sudo mount^4.

Entramos en la carpeta mount y curioseamos un poco a ver que contiene:

```
linux@linux-desktop:~/RetosUAM/firmware$ cd mount/  
linux@linux-desktop:~/RetosUAM/firmware/mount$ ls -la  
total 43  
drwxr-xr-x 21 root root 1024 jul 13 20:58 .  
drwxr-xr-x 3 linux linux 4096 jul 17 20:37 ..  
-rwxr-xr-x 1 root root 7548 jul 13 20:58 .bomb  
drwxr-xr-x 3 root root 1024 jul 13 21:02 boot  
drwxr-xr-x 6 root root 1024 jul 13 20:55 cdrom  
drwxr-xr-x 7 root root 1024 jul 13 20:55 dev  
drwxr-xr-x 2 root root 1024 jul 13 20:56 etc  
drwxr-xr-x 2 root root 1024 jul 13 20:54 home  
drwxr-xr-x 2 root root 1024 jul 13 21:07 lib  
drwxr-xr-x 2 root root 1024 jul 13 20:54 lib64  
drwx----- 2 root root 12288 jul 13 20:48 lost+found  
drwxr-xr-x 2 root root 1024 jul 13 20:56 media  
drwxr-xr-x 2 root root 1024 jul 13 20:54 opt  
drwxr-xr-x 2 root root 1024 jul 13 20:54 proc  
drwxr-xr-x 3 root root 1024 jul 13 20:56 root  
drwxr-xr-x 2 root root 1024 jul 13 20:54 run  
drwxr-xr-x 2 root root 1024 jul 13 21:01 sbin  
drwxr-xr-x 2 root root 1024 jul 13 20:54 srv  
drwxr-xr-x 2 root root 1024 jul 13 20:54 sys  
drwxr-xr-x 3 root root 1024 jul 13 21:00 tmp  
drwxr-xr-x 2 root root 1024 jul 13 20:54 usr  
drwxr-xr-x 2 root root 1024 jul 13 20:54 var
```

Ummmm interesante, vemos un fichero .bomb en la carpeta raíz y un esquema de directorios propio de un sistema *nix. Vamos a buscar todos los ficheros que contienen la palabra bomb en el sistema de archivos con la orden find:

Esta orden recibe como primer parámetro el punto a partir del cual empezamos a buscar los ficheros (en nuestro caso el directorio actual ./) y como segundo parámetro el comando -name seguido del texto que queremos buscar , en este caso utilizando los asteriscos como comodines:

```
linux@linux-desktop:~/RetosUAM/firmware/mount$ find ./ -name "*bomb*"
./lib/bomb.key
./lib/bomb.0
./lib/bomb.resolve
./root/firmware/src/executables/defusedbombInstruction.a
./.bomb
find: './lost+found': Permiso denegado
./boot/grub/locale/.bomb
```

Vaya , varias cosas interesantes en este comando. Primero, como hemos montado la imagen como root vemos un directorio al que no podemos acceder ./lost+found y por si tiene algo interesante repetimos la orden con sudo

```
linux@linux-desktop:~/RetosUAM/firmware/mount$ sudo find ./ -name
"*bomb*"
./lib/bomb.key
./lib/bomb.0
./lib/bomb.resolve
./root/firmware/src/executables/defusedbombInstruction.a
./.bomb
./boot/grub/locale/.bomb
```

Vemos que los ficheros que obtenemos son los que ya habíamos visto. Tambien podríamos haber cambiado los permisos del directorio ./lost+found con la orden chmod o haber cambiado el propietario del mismo con la orden chown pero queremos modificar lo menos posible la imagen original así que, de momento, no utilizaremos órdenes que nos modifiquen el sistema de ficheros. Una opción también válida para no modificar el sistema de ficheros original es montarlo en modo “solo lectura” para evitar modificaciones accidentales.

Miremos que son esos ficheros con la orden file

```

linux@linux-desktop:~/RetosUAM/firmware/mount$ file ./lib/bomb.key
./lib/bomb.key: ASCII text
linux@linux-desktop:~/RetosUAM/firmware/mount$ file ./lib/bomb.0
./lib/bomb.0: ASCII text, with very long lines
linux@linux-desktop:~/RetosUAM/firmware/mount$ file ./lib/bomb.resolve
./lib/bomb.resolve: ASCII text, with very long lines
linux@linux-desktop:~/RetosUAM/firmware/mount$ file ./root/firmware/src/executables/defusedbombInstruction.a
./root/firmware/src/executables/defusedbombInstruction.a: empty
linux@linux-desktop:~/RetosUAM/firmware/mount$ file ./boot/grub/locale/.bomb
./boot/grub/locale/.bomb: ELF 64-bit LSB executable, x86-64, version
1 (GNU/Linux), statically linked, stripped
linux@linux-desktop:~/RetosUAM/firmware/mount$ file ./.bomb
./.bomb: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux),
statically linked, stripped

```

Bien, ordenémoslos en un tabla para que se vea mejor el tipo

Fichero	Tipo
./lib/bomb.key	ASCII text
./lib/bomb.0	ASCII text, with very long lines
./lib/bomb.resolve	ASCII text, with very long lines
./root/firmware/src/executables/defusedbo mblInstruction.a	empty
./boot/grub/locale/.bomb	ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statically linked, stripped
./.bomb	ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statically linked, stripped

Descartamos el fichero vacío , aunque luego volveremos a ese directorio ya que el nombre es interesante.

Miramos primero que tienen los ficheros de texto con líneas largas con la orden cat que nos puede servir para ver el contenido de un fichero de texto (aunque su propósito original es otro)

```
linux@linux-desktop:~/RetosUAM/firmware/mount$ cat ./lib/bomb.0
IkxvcmVtIGlwc3VtIGRvbG9yIHNPdCBhbWV0LCBjb25zZWN0ZXR1ciBhZGlwaXNjaW5nI
GVsaXQsIHNLZCBkbyBlaXVzbW9kIHRlbXBvcIBpbmNpZG1kdW50IHV0IGxhYm9yZSB1dC
Bkb2xvcmUgbWFnbmEgYWxpcXvhLiBVdCB1bmltIGFkIG1pbmltIHZ1bmlhbSwgcXVpcyB
ub3N0cnVkJG4ZXJjaXRhdGlvbib1bGxhbWNvIGxhYm9yaXMgbmlzaSB1dcBhbGlxdWlw
IGV4IGVhIGNvbW1vZG8gY29uc2VxdWF0LiBEDWlzIGF1dGUgaXJ1cmUgZG9sb3IgaW4gc
mVwcmVoZW5kZXJpdCBpbIB2b2x1cHRhdGUgdmVsaXQgZXNzzSBjaWxsdW0gZG9sb3J1IG
V1IGZ1Z21hdCBudWxsYSBwYXJpYXR1ci4gRXhjZXB0ZXVYIHNPbnQgb2NjYWVjYXQgY3V
waWRhdGF0IG5vbiBwcm9pZGVudCwgc3VudCBpbIBjdWxwYSBxdWkgb2ZmaWNpYSBkZXN1
cnVudCBtb2xsaXQgYW5pbSBpZCB1c3QgbGFib3J1bs4iasdas
```

Parece un texto en base64 o algún tipo de cifrado, miremos el otro fichero

```
linux@linux-desktop:~/RetosUAM/firmware/mount$ cat ./lib/bomb.resolve
IkxvcmVtIGlwc3VtIGRvbG9yIHNPdCBhbWV0LCBjb25zZWN0ZXR1ciBhZGlwaXNjaW5nI
GVsaXQsIHNLZCBkbyBlaXVzbW9kIHRlbXBvcIBpbmNpZG1kdW50IHV0IGxhYm9yZSB1dC
Bkb2xvcmUgbWFnbmEgYWxpcXvhLiBVdCB1bmltIGFkIG1pbmltIHZ1bmlhbSwgcXVpcyB
ub3N0cnVkJG4ZXJjaXRhdGlvbib1bGxhbWNvIGxhYm9yaXMgbmlzaSB1dcBhbGlxdWlw
IGV4IGVhIGNvbW1vZG8gY29uc2VxdWF0LiBEDWlzIGF1dGUgaXJ1cmUgZG9sb3IgaW4gc
mVwcmVoZW5kZXJpdCBpbIB2b2x1cHRhdGUgdmVsaXQgZXNzzSBjaWxsdW0gZG9sb3J1IG
V1IGZ1Z21hdCBudWxsYSBwYXJpYXR1ci4gRXhjZXB0ZXVYIHNPbnQgb2NjYWVjYXQgY3V
waWRhdGF0IG5vbiBwcm9pZGVudCwgc3VudCBpbIBjdWxwYSBxdWkgb2ZmaWNpYSBkZXN1
cnVudCBtb2xsaXQgYW5pbSBpZCB1c3QgbGFib3J1bs4i
```

el segundo parece una parte del primero menos los últimos caracteres. Lo confirmaremos con la orden vimdiff que nos muestra las diferencias entre dos ficheros de texto en un editor de textos vim de forma visual

```
linux@linux-desktop:~/RetosUAM/firmware/mount$ vimdiff ./lib/bomb.0
./lib/bomb.resolve
```

y moviéndonos al final de la línea confirmamos lo que a simple vista nos parecía.

```
linux@linux-desktop: ~/RetosUAM/firmware/mount
```

```
linux@linux-desktop: ~/RetosUAM/reto3
```

```
linux@linux-desktop: ~/RetosUAM/firmware/mount 100x23
```

```
ZCBlc3QgbGFib3J1bS4iasdas
```

```
ZCBlc3QgbGFib3J1bS4i
```

```
./lib/bomb.0 [RO] 1,593 Todo ./lib/bomb.resolve [RO] 1,593 Todo
```

```
"./lib/bomb.resolve" [Sólo lectura] 2L, 598C
```

Ejecutamos dos veces :q! en el vimdiff para salir sin alterar los ficheros

Veamos si es un base64, decodificandolo, utilizaremos un comando de linux en lugar de una página web para ello.

```
linux@linux-desktop:~/RetosUAM/firmware/mount$ cat ./lib/bomb.0 | base64 --decode
"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."j♦Zbase64:
entrada inválida
```

Leemos el texto por si es algo, pero vemos que no, que parece una falsa pista. El mensaje en la parte final de la respuesta proviene del comando base64 y nos indica que es un mensaje mal formado. Ya nos lo esperábamos ya que la secuencia de caracteres asdas que aparece como distinta al final del fichero, como las secuencias qweqw o zxczx se suelen utilizar para poner caracteres al azar ya que son teclas que se encuentran juntas en un teclado qwerty. Ya vemos que los creadores del reto son algo vagos :)

Veamos el segundo fichero:

```
linux@linux-desktop:~/RetosUAM/firmware/mount$ cat ./lib/bomb.resolve  
| base64 --decode  
  
"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do  
eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim  
ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut  
aliquip ex ea commodo consequat. Duis aute irure dolor in  
reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla  
pariatur. Excepteur sint occaecat cupidatat non proident, sunt in  
culpa qui officia deserunt mollit anim id est laborum."
```

En este caso, un mensaje bien formado, contiene los mismos caracteres que el mensaje original.

Comprobemos el otro mensaje de texto en el fichero ./lib/bomb.key

```
linux@linux-desktop:~/RetosUAM/firmware/mount$ cat ./lib/bomb.key  
-nop-
```

!!!Vaya parece un texto cifrado en algún cifrado exótico!!! . No, mejor pensamos que es otra falsa pista.

Veamos los dos binarios que tenemos. Miraremos si son iguales con la orden cmp que nos permite comparar binarios:

```
linux@linux-desktop:~/RetosUAM/firmware/mount$ cmp ./lib/bomb  
./boot/grub/locale/.bomb  
linux@linux-desktop:~/RetosUAM/firmware/mount$
```

La orden no devuelve nada, porque los comandos de Linux / UNIX son muy respetuosos y si no tienen nada que decir, no dicen nada. Me recuerdan esta historia

Una pareja tiene un bebé y cuando cumple un año el niño aún no ha dicho ni una palabra, pasan dos, tres años y le llevan al médico. Este determina que el niño es mudo. A los siete años , un día comiendo, el niño se queda mirando a sus padres y les dice:

- Padres, la sopa está fría

Los padres muy emocionados, le dicen

-
- ¡Pero hijo mío! ¡Si sabes hablar! ¿Porque no nos has dicho nada hasta ahora?

A lo que el niño responde

- Padres, hasta ahora, todo perfecto.

Eso significa que los ficheros son iguales. Comprobémoslo calculando el hash sha1 de ambos

```
linux@linux-desktop:~/RetosUAM/firmware/mount$ shasum ./bomb  
7832735b7e68c771e844d811dd2407ebf31c602d ./bomb  
linux@linux-desktop:~/RetosUAM/firmware/mount$ shasum  
./boot/grub/locale/.bomb  
7832735b7e68c771e844d811dd2407ebf31c602d ./boot/grub/locale/.bomb
```

Efectivamente son el mismo fichero. vamos a ejecutarlo a ver que pasa:



ESTO NUNCA LO DEBEMOS HACER CON UN FICHERO BINARIO DESCONOCIDO,
PODRÍAMOS CAER EN LAS GARRAS DE CUALQUIER TIPO DE MALWARE, DEBERÍAMOS DE

EJECUTARLO EN UN ENTORNO CONTROLADO , por ejemplo en una máquina virtual. Pero nos fiamos de los organizadores del reto ¿o mejor no 😊?

```
linux@linux-desktop:~/RetosUAM/firmware/mount$ ./.bomb
|-----|
|-----|
|-----| 0:59 |-----|
|-----|
|-----|
```

¡¡¡Dios mio una bomba a punto de explotar, con una cuenta atrás en descenso , rapido, rapido Crtl-C Crtl-C Crtl-C !!!

Como no nos fiamos mucho de los organizadores abortamos la ejecución con control + la letra c y, por suerte, la bomba para su cuenta atrás saliendo del programa. Vamos a investigar un poco que es dicho fichero, haremos un poco de reversing sobre el.

Reversing del fichero .bomb

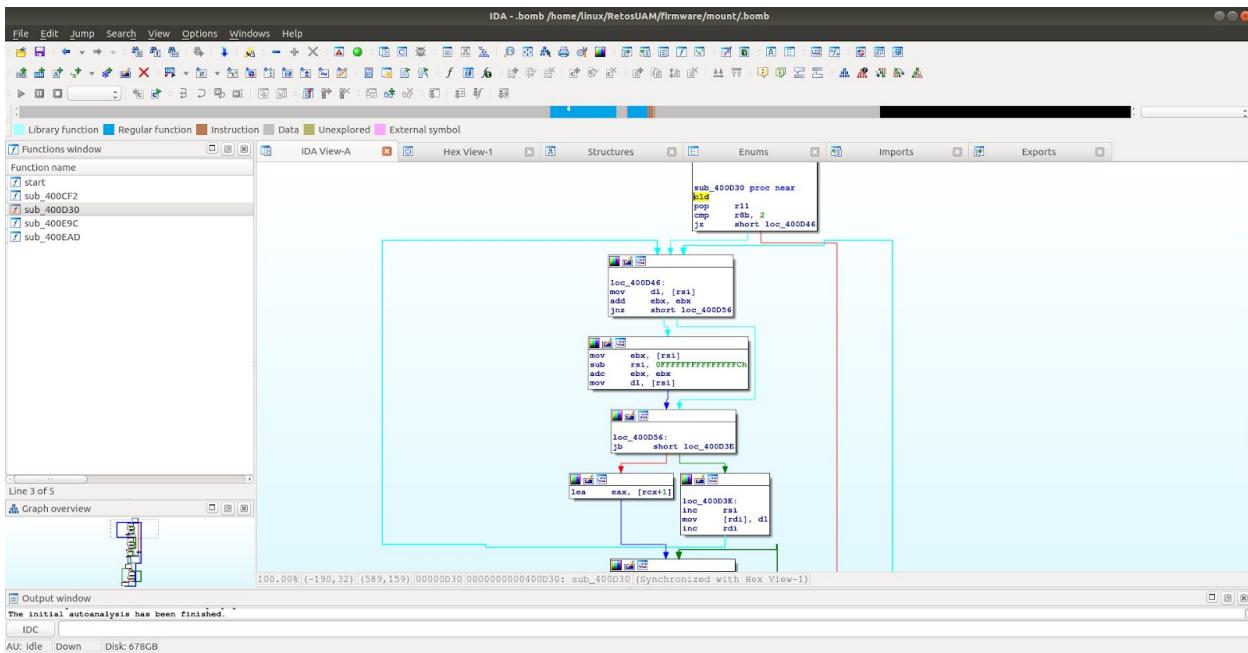
Lo primero que hacemos es abrir IDA en su versión gratuita que se puede descargar de https://www.hex-rays.com/products/ida/support/download_freeware.shtml

Como indican en su página

The freeware version of IDA v7.0 has the following limitations:

- no commercial use is allowed
- lacks all features introduced in IDA > v7.0
- lacks support for many processors, file formats, debugging etc...
- comes without technical support

A si que abrimos el desensamblador e indicamos el fichero



Miramos en el código , seguimos las funciones definidas, miramos en los diagramas de código que IDA nos va pintando y añadimos algunos comentarios al mismo para irnos orientando:

```

LOAD:0000000000400CE8      sub    rsi, OFFFFFFFFFFFFFFFCh
LOAD:0000000000400CEC      adc    ebx, ebx
LOAD:0000000000400CEE      mov    dl, [rsi]
LOAD:0000000000400CF0      rep    retn
LOAD:0000000000400CF0 start endp ; sp-analysis failed
LOAD:0000000000400CF0
LOAD:0000000000400CF2
LOAD:0000000000400CF2 ; ===== S U B R O U T I N E =====
LOAD:0000000000400CF2
LOAD:0000000000400CF2
LOAD:0000000000400CF2
LOAD:0000000000400CF2 sub_400CF2 proc near ; CODE XREF: sub_400D30+8C+j
LOAD:0000000000400CF2      lea    rax, [rdi+rpb] ; Bien pensado
LOAD:0000000000400CF6      cmp    ecx, 5
LOAD:0000000000400CF9      mov    dl, [rax]
LOAD:0000000000400CFB      jbe    short loc_400D1E ; Asi si, bien bien
LOAD:0000000000400CFB      cmp    rbp, OFFFFFFFFFFFFFFFCh
LOAD:0000000000400CFD      ja    short loc_400D1E ; JUGADA MAESTRA !!!!!
LOAD:0000000000400D01 ANALIZAMOS
LOAD:0000000000400D01      ja    short loc_400D1E ; JUGADA MAESTRA !!!!!
LOAD:0000000000400D03
LOAD:0000000000400D06
LOAD:0000000000400D06 loc_400D06:           sub    ecx, 4
LOAD:0000000000400D06      mov    edx, [rax]
LOAD:0000000000400D08      add    rax, 4
LOAD:0000000000400D0C      sub    ecx, 4
LOAD:0000000000400D0F      mov    [rdi], edx
LOAD:0000000000400D11      lea    rdi, [rdi+4]
LOAD:0000000000400D15      jnb    short loc_400D06
LOAD:0000000000400D17      add    ecx, 4

```

El libro de ensamblador me mira desde la estantería y me dice:

但你在做什麼！如果你從未打開我！

¡Osti tu nen!, voy rapido a google translate y me traduce correctamente

The screenshot shows the Google Translate interface. At the top, it says "Traductor de Google" and "Desactivar traducción instantánea". Below that, there are two input fields: the left one has "inglés" and "español" selected, and the right one has "español" and "chino (tradicional)" selected. A "Traducir" button is to the right of the second field. The main area shows a Chinese message "但你在做什麼！如果你從未打開我！" which is translated into Spanish as "Pero que estas haciendo! ¡Si nunca me abres!". Below the input text, there is a note in Chinese: "Dàn nǐ zài zuò shénme! Rúguō nǐ cóng wéi dǎkāi wǒ!" and a character count "17/5000". There are also some icons and a "Sugerir un cambio" link.

Asumo mis incapacidades y anoto en mi lista de TODO. "Aprender ensamblador, y que te deje de sonar a chino."

Cierro el IDA y decido que voy a jugármela y a ejecutar el fichero para ver qué hace cuando explote. Me fio, pero no mucho, así que monitorizaré lo que va haciendo con strace. Como dice la wikipedia

"strace es una utilidad de línea de comandos para comprobación de errores en el sistema operativo GNU/Linux. Permite monitorear las llamadas al sistema usadas por un determinado programa y todas las señales que éste recibe.¹ Su funcionamiento es posible por una característica del núcleo linux llamada ptrace. Es similar a la aplicación truss disponible en otros sistemas Unix" <https://es.wikipedia.org/wiki/Strace>

Por si las moscas, le cambiaré también el propietario al fichero, aunque sea modificar el sistema de ficheros original que teníamos montado, por un usuario con permisos restringidos en el sistema, en nuestro caso el usuario linux.

```
linux@linux-desktop:~/RetosUAM/firmware/mount$ sudo chown linux:linux .bomb
```

Ejecutamos la bomba y obtenemos el identificador del proceso con el comando ps para pasarselo al comando strace con posterioridad y ver que está haciendo.

```
linux@linux-desktop:~/RetosUAM/firmware/mount$ ./.bomb
```

y en otra consola

```
linux@linux-desktop:~$ ps uax
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME
COMMAND

...
linux      18377  0.0  0.0  18112  3296 pts/0      T     20:03   0:00
./.bomb

...
linux@linux-desktop:~$ sudo strace -p 18377
.....
rt_sigaction(SIGINT, {sa_handler=SIG_DFL, sa_mask=[],  
sa_flags=SA_RESTORER, sa_restorer=0x7f4dbca2bf20}, NULL, 8) = 0  
rt_sigaction(SIGQUIT, {sa_handler=SIG_DFL, sa_mask=[],  
sa_flags=SA_RESTORER, sa_restorer=0x7f4dbca2bf20}, NULL, 8) = 0  
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0  
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=18602,  
si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---  
write(1, "\7|-----|\n", 23) = 23  
write(1, "\n", 1) = 1  
write(1, "|-----|\n", 22) = 22  
write(1, "\n", 1) = 1  
write(1, "|-----|0:17|----|\n", 22) = 22  
write(1, "\n", 1) = 1  
write(1, "|-----|\n", 22) = 22  
write(1, "\n", 1) = 1  
write(1, "|-----|\n", 22) = 22  
write(1, "\n", 1) = 1  
nanosleep({tv_sec=1, tv_nsec=0}, 0x7ffea9677e20) = 0  
rt_sigaction(SIGINT, {sa_handler=SIG_IGN, sa_mask=[],  
sa_flags=SA_RESTORER, sa_restorer=0x7f4dbca2bf20},  
{sa_handler=SIG_DFL, sa_mask=[], sa_flags=SA_RESTORER,  
sa_restorer=0x7f4dbca2bf20}, 8) = 0  
.....
```

```

linux@linux-desktop: ~/RetosUAM/firmware/mount 81x32
[...]
[0:30] [...]
[...]
write(1, "\n", 1) = 1
write(1, "|-----|\n", 22) = 22
write(1, "\n", 1) = 1
nanosleep({tv_sec=1, tv_nsec=0}, 0x7ffea9677e20) = 0
rt_sigaction(SIGINT, {sa_handler=SIG_IGN, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7f4dbca2bf20}, {sa_handler=SIG_DFL, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7f4dbca2bf20}, 8) = 0
rt_sigaction(SIGQUIT, {sa_handler=SIG_IGN, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7f4dbca2bf20}, {sa_handler=SIG_DFL, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7f4dbca2bf20}, 8) = 0
clone(child_stack=NULL, flags=CLONE_PARENT_SETTID|SIGCHLD, parent_tidptr=0x7ffea9677cdc) = 18559
wait4(18559, [{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 18559
rt_sigaction(SIGINT, {sa_handler=SIG_DFL, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7f4dbca2bf20}, {sa_handler=SIG_DFL, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7f4dbca2bf20}, 8) = 0
rt_sigaction(SIGQUIT, {sa_handler=SIG_DFL, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7f4dbca2bf20}, {sa_handler=SIG_DFL, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7f4dbca2bf20}, 8) = 0
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=18559, si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---
write(1, "\n|-----|\n", 23) = 23
write(1, "\n", 1) = 1
write(1, "|-----|\n", 22) = 22
write(1, "\n", 1) = 1
write(1, "|-----|0:30|-----|\n", 22) = 22
write(1, "\n", 1) = 1
write(1, "|-----|\n", 22) = 22
write(1, "\n", 1) = 1
nanosleep({tv_sec=1, tv_nsec=0}, 0x7ffea9677e20) = 0

```

No vemos nada raro en la salida de strace, no parece que este borrando ficheros ni nada parecido, a si que esperamos con impaciencia a que acabe la cuenta atrás. Al finalizar, sorpresa, nos pide un codigo (para desactivar la bomba suponemos) Buscamos en internet como actuan los Tedax y encontramos esto

https://www.eldiario.es/canariasahora/sociedad/trabajan-especialistas-desactivacion-explosivos_0_568693305.html



Canarias Ahora / sociedad

Así trabajan los especialistas en desactivación de explosivos

- Su labor es discreta y poco conocida. Suelen actuar de paisano para que no se desate el pánico.

Saúl García Seguir a @saulgariacres - Las Palmas de Gran Canaria

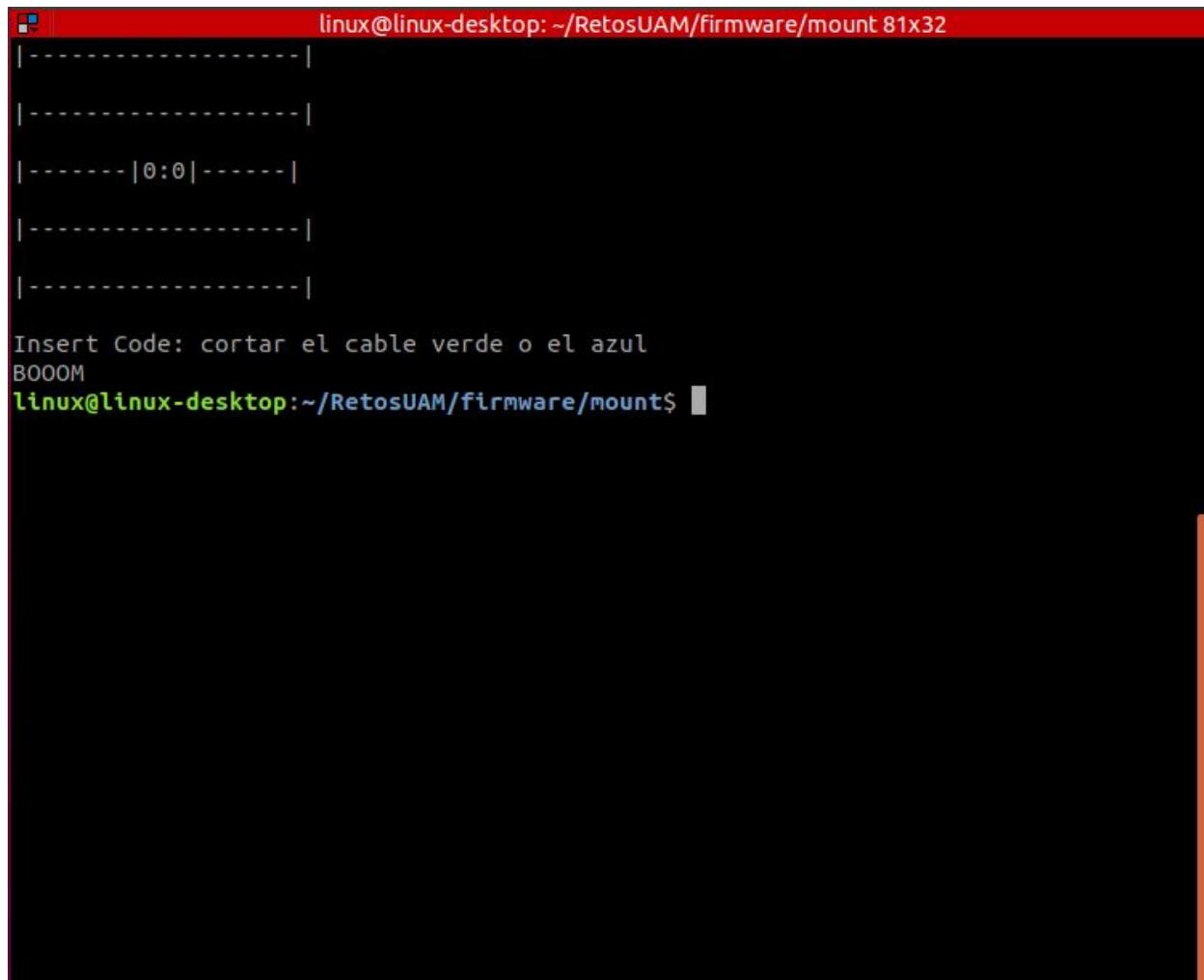
12/10/2016 - 12:09h



Fundación Anesvad



a sí que, tal y como comentan en el artículo, le ponemos “cortar el cable verde o el azul”.



```
linux@linux-desktop: ~/RetosUAM/firmware/mount 81x32
|-----|
|-----|
|-----| 0:0 |-----|
|-----|
|-----|
Insert Code: cortar el cable verde o el azul
BOOOM
linux@linux-desktop:~/RetosUAM/firmware/mount$
```

Zas, en toda la cara, tendría que haber puesto cortar el cable rojo, lo sabía. Tardamos un rato en recoger toda la habitación y en arreglar el estropicio que ha causado la bomba. Por suerte en el sistema no ha causado ningún daño, como nos dice strace

```
read(0, "cortar el cable verde o el azul\n", 1024) = 32
write(1, "BOOOM\n", 6)                      = 6
lseek(0, -26, SEEK_CUR)                     = -1 ESPIPE (Illegal seek)
exit_group(0)                                = ?
+++ exited with 0 +++
```

Pasamos a analizar el binario con strings, una orden “humana” no como lo que me decía IDA, que yo no entendía para nada, lo que nos permite ver las cadenas de texto de un fichero, como parámetro le pasamos -n8 que le indica que solamente nos muestre las palabras de 8 o más de 8 caracteres de longitud.

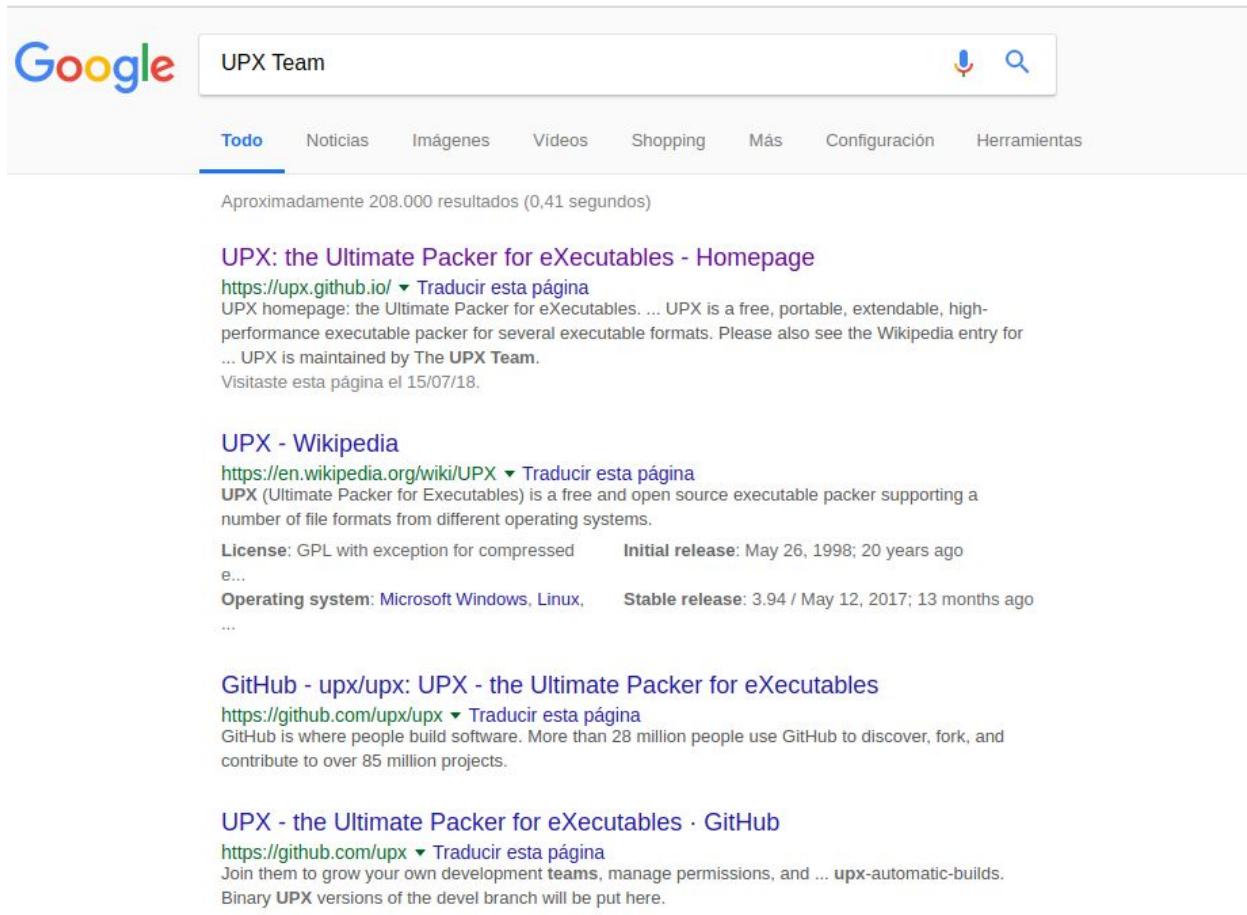
```

linux@linux-desktop:~/RetosUAM/firmware/mount$ strings -n8 .bomb
nux-x86-
stdc++s6
__gmon_o
11ch]_trao
o<BamJ_T0_
_cxx<12%-_
$1g1S4_S5(1E
ty_v0%Ks
-4endl.+]
";"D9ixEm
m+LGCC_3.[
/1GLIB      2.4
CXXABI_1.3
X;%w  " ||
_dbf7c981d7e_fe8
c462eab3c3~
PROT_EXEC|PROT_WRITE failed.
$000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000Id: UPX
Copyright (C) 1996-2013 the UPX Team. All Rights Reserved. $
/proc/self/exe
AZt?n<,5
GCC: (Ubuntu 5.4.0-6u
__JCR_LIST
_tm_clones
bal   tors_auxB
~omple)d.7594$_
me ummy2
'ZStL8-io#
atic5iriz
?ndN7ruc
i/GLOBAL2z_I_m
FRAME_END__
DRCOFFSE(TABLE(u
IBCXX_3.6ZNSolsEi

```

WrDITM_VEh
ES5_PKcAt7
F+=dsyc~1e7

La cadena de texto “Copyright (C) 1996-2013 the UPX Team” me suena de algo ... voy a ver que me dice google



The screenshot shows a Google search results page for the query "UPX Team". The top navigation bar includes the Google logo, a search bar containing "UPX Team", a microphone icon, and a magnifying glass icon. Below the search bar are tabs for "Todo", "Noticias", "Imágenes", "Videos", "Shopping", "Más", "Configuración", and "Herramientas", with "Todo" being the active tab. A status message indicates "Aproximadamente 208.000 resultados (0,41 segundos)". The first result is a link to the UPX homepage: "UPX: the Ultimate Packer for eXecutables - Homepage" (<https://upx.github.io/>). The snippet describes UPX as a free, portable, extendable, high-performance executable packer. The second result is a link to the UPX Wikipedia page: "UPX - Wikipedia" (<https://en.wikipedia.org/wiki/UPX>). The snippet provides details about its history, license (GPL), operating systems (Windows, Linux), and release dates. The third result is a link to the UPX GitHub repository: "GitHub - upx/upx: UPX - the Ultimate Packer for eXecutables" (<https://github.com/upx/upx>). The snippet explains GitHub's role in software development. The fourth result is a link to the UPX GitHub page: "UPX - the Ultimate Packer for eXecutables · GitHub" (<https://github.com/upx/upx>). The snippet encourages users to join the team and manage permissions.

Si que me sonaba si, es un “empaquetador”, una herramienta utilizada para reducir el tamaño de un ejecutable y para hacer más “difícil” su análisis en algunas ocasiones, ya que transforma el contenido, es como hacer un zip de un fichero de texto e intentar analizar el resultado del zip para descubrir que había en el fichero original. Recuerdo que estos empaquetados se podían desempaquetar a si que el desempaquetador que los desempaque buen desempaquetador será. Vamos a buscar por Google

A Google search results page for the query "unpack UPX". The top result is a link to "Descomprimir UPX con UPX. Unpack UPX. - karmany NET" from www.karmany.net. Below the link are three video thumbnails:

- UPX Easily Unpacked All Versions** by System_Override (YouTube) - 30 nov. 2009 (Duration: 4:25)
- Unpack UPX file exe** by Moon Dark (YouTube) - 20 abr. 2016 (Duration: 2:46)
- [Tutorial] Unpack UPX by BlackSource** by Ayuth Mangmesap (YouTube) - 22 ago. 2014 (Duration: 13:20)

Unpacking UPX packed (possibly scrambled) executable - Reverse ...

La primera entrada parece prometedora, vamos a ella, la leemos, aprendemos alguna cosa más de UPX y vemos que existe el parametro -d para ello. Antes de nada instalamos el paquete con

```
sudo apt-get install upx
```

y luego ejecutamos la orden

```
linux@linux-desktop:~/RetosUAM/firmware/mount$ sudo upx -d .bomb
[sudo] contraseña para linux:
          Ultimate Packer for eXecutables
          Copyright (C) 1996 - 2017
UPX 3.94           Markus Oberhumer, Laszlo Molnar & John Reiser   May
12th 2017

      File size        Ratio       Format       Name
-----  -----  -----
  15280 <-    7548    49.40%  linux/amd64  .bomb

Unpacked 1 file.
```

Bien, parece que ha funcionado, antes teníamos

```
-rwxr-xr-x 1 linux linux 7548 jul 13 20:58 .bomb
```

y ahora tenemos

```
-rwxr-xr-x 1 linux linux 15280 jul 13 20:58 .bomb
```

El fichero ha doblado su tamaño, vamos a ver que contiene el despaquetado con la orden anterior strings, eliminó las partes no interesantes en este documento con ... Morralla ...

```
linux@linux-desktop:~/RetosUAM/firmware/mount$ strings -n8 .bomb
/lib64/ld-linux-x86-64.so.2
libstdc++.so.6
... Morralla ...
CXXABI_1.3
GLIBCXX_3.4.21
GLIBCXX_3.4
[]A\A]A^A_
_dbf7c981d7e_fe8
_c462eab3c39
_f2b06_fd
Tienes 1 minuto
| -----
| -----
| -----
Insert Code:
GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.10) 5.4.0 20160609
... Morralla ...
frame_dummy
__frame_dummy_init_array_entry
test.cpp
... Morralla ...
_Jv_RegisterClasses
... Morralla ...
__data_start
... Morralla ...
sleep@@GLIBC_2.2.5
__TMC_END__
_ZSt4cout@@GLIBCXX_3.4
__dso_handle
... Morralla ...
segundos
... Morralla ...
.shstrtab
.note.ABI-tag
```

```
.note.gnu.build-id  
.gnu.hash  
.gnu.version  
.gnu.version_r  
.rela.dyn  
.rela.plt  
.plt.got  
.eh_frame_hdr  
.eh_frame  
.gcc_except_table  
.init_array  
.fini_array  
.dynamic  
.got.plt  
.comment
```

bueno vemos varios mensajes de los que veíamos por pantalla y algunas cadenas que parecen llamadas a funciones en c , aun así las probamos todas, una por una.

Nada, ninguna funciona. Vaya por dios, no parece que este en los strings la cadena de desactivación. Después de un rato pensando, como siempre que parece que estoy en un callejón sin salida, vuelvo sobre mis pasos y repaso el comando que utilice. ¿Y sacó cadenas de texto más pequeñas, no sea que el código de desactivación tenga menos de 8 caracteres?, vamos a intentarlo. Antes que volver al ensamblador, lo que sea. (Aunque oigo que mi libro de ensamblador de la estantería me dice 懶夫, 懶夫, le ignoro). Este vez le paso la orden sin parámetros (4 caracteres por defecto)

```
linux@linux-desktop:~/RetosUAM/firmware/mount$ strings .bomb
... Morralla ...
_dbf7c981d7e_fe8
_c462eab3c39
_f2b06_fd
Tienes 1 minuto
clear
| -----
| -----
| -----
Insert Code:
italy
BOOOM
... Morralla ...
```

Ummmm aparece la palabra italy justo entre insert code y booom, probemos a ver, lo volvemos a ejecutar y vemos que el desempaquetado ha funcionado, ya que la aplicación se comporta, al menos aparentemente, igual.

```
| -----
| -----
| -----| 0:0 |-----|
| -----
| -----
Insert Code: italy
_dbf7c981d7e_fe8_c462eab3c39_f2b06_fdlinux@linux-desktop:~/RetosUAM/f
irmware/moulinux@linux-desktop:~/RetosUAM/firmware/mount$
```

¡Toma! los caracteres que veíamos antes con la orden string y que ya nos llamaron la atención aparecen aquí con un determinado orden formando la cadena

_dbf7c981d7e_fe8_c462eab3c39_f2b06_fd

¿Que sera eso?

Resolviendo la dirección del servidor

La verdad es que esta cadena nos volvió locos durante muchas horas, analizamos los caracteres que aparecen en ella, aunque a primera vista parece hexadecimal, los subrayados son raros. utilizamos para el análisis de frecuencias el recurso online <https://www.dcode.fr/frequency-analysis>

Search for a tool

★ SEARCH A TOOL ON DCODE BY KEYWORDS:
e.g. type scrabble

Results

	5×	13.51%
F	4×	10.81%
C	3×	8.11%
B	3×	8.11%
E	3×	8.11%
D	3×	8.11%
6	2×	5.41%
3	2×	5.41%
2	2×	5.41%
8	2×	5.41%
7	2×	5.41%
9	2×	5.41%
0	1×	2.7%
1	1×	2.7%
A	1×	2.7%
4	1×	2.7%
#N : 16 Σ = 37.000 Σ = 100.02		

FREQUENCY ANALYSIS

Cryptography > Cryptanalysis > Frequency Analysis

Sponsored ads

JFrog ARTIFACTORY Universal Artifact Management for DevOps Acceleration | FREE 30-DAY TRIAL LEARN MORE

Frequency Analysis (advanced)

★ TEXT TO ANALYZE
_dbf7c981d7e_fe8_c462eab3c39_f2b06_fd

Target characters for frequency analysis

LETTERS (A-Z) ONLY
 LETTERS (A-Z) AND DIGITS (0-9) ONLY
 DIGITS (0-9) ONLY
 ONLY THESE CHARACTERS: αβγδε
 ALL CHARACTERS (INCLUDING PUNCTUATION AND SYMBOLS)
★ STANDARDIZE LETTERS (IGNORE UPPER-LOWER CASE AND DIACRITICS)

Items to Analyze

EACH CHARACTER SEPARATELY
 BIGRAMS (COUPLES OF 2 CHARACTERS)

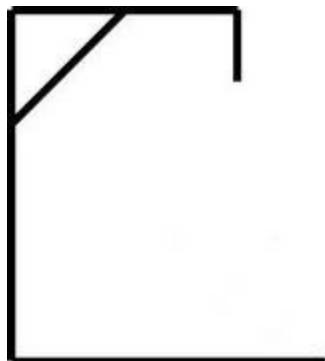
Pues si, parecen caracteres del código hexadecimal. pero hay varias cosas raras:

- Con la orden string aparecian en tres grupos _dbf7c981d7e_fe8 , _c462eab3c39 y _f2b06_fd

-
- Ninguno de los grupos sacado con string parece significar nada en hexadecimal
 - Tenemos 5 caracteres _, no pueden ser los puntos de una ip ya que solo tenemos tres puntos en una ip
 - Ninguno de los grupos separados por _ tiene un número par de caracteres, excepto el último, “necesario” para ser hexadecimal dbf7c981d7e (11 caracteres), fe8 (3), c462eab3c39 (11) , f2b06 (5), fd(2)

Aunque no tiene pinta, por utilizar solamente 17 caracteres y por la frecuencia de aparición de los mismos (se puede leer sobre esto en https://en.wikipedia.org/wiki/Letter_frequency sobre el idioma inglés Ingles, la página en español es bastante peor aunque nos da la frecuencia en nuestro idioma) probamos algunos métodos de cifrado clásicos sin resultado.

De pronto me viene una idea ¡Ya está! ¡¡Es un juego del ahorcado!! cinco posiciones , cinco letras de la clave de desactivación!! Juguemos a ello

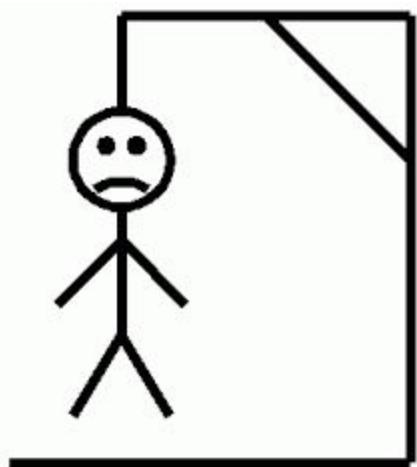


idbf7c981d7etfe8ac462eab3c39lf
2b06yfd

La cadena que sale es muy rara pero la buscamos en google

The screenshot shows a Google search interface. The search bar contains the query "idbf7c981d7ef8ac462eab3c39f2b06yfd". Below the search bar are navigation links: Todo (highlighted in blue), Maps, Imágenes, Videos, Noticias, Más, Configuración, and Herramientas. To the right of the search bar are a microphone icon and a magnifying glass icon. The main content area displays the message: "La búsqueda de idbf7c981d7ef8ac462eab3c39f2b06yfd no obtuvo ningún resultado." followed by "Sugerencias:" and a bulleted list: "• Comprueba que todas las palabras están escritas correctamente.", "• Intenta usar otras palabras.", and "• Intenta usar palabras más generales."

No ha habido suerte. Probamos de nuevo con esta cadena en varios sistemas de cifrado clásico en la pagina anterior <https://www.dcode.fr/tools-list> pero sin suerte. Nos quedamos así:



Bueno de esta no hemos sacado nada en claro. Quitaremos los caracteres subrralados a ver que tenemos:

dbf7c981d7ef8ac462eab3c39f2b06fd

Interesante 32 caracteres, 16 Bytes, esto nos da una idea de que se puede tratar de un hash MD5 que tiene esa longitud. Probamos en varias páginas de reversing de MD5 que conocemos, a ver qué encontramos

<https://isc.sans.edu/tools/reversehash.html>

The screenshot shows a web browser window for the ISC SANS Reverse Hash Calculator. The URL is https://isc.sans.edu/tools/reversehash.html. The page has a green header bar with "Threat Level: GREEN" and "Handler on Duty: Kevin Liston". On the left, there's a sidebar with links like Contact Us, Diary, Podcasts, Jobs, News, and TOOLS (DShield Sensor, Honeypot (RPI/AWS), InfoSec Glossary). The main content area has a "Reverse Hash Calculator" section with a search bar containing "Keyword, Domain, Port, IP or He:" and a "Search" button. Below the search bar, there are links for "Back to Tools", "Background", "Search Form", and "Last 20 Hashes". A note says: "This page doesn't use rainbow tables (yet), but a similar, simpler approach. It uses a database of a couple million pre-compiled hash values. The strings used come from various password databases, and should have a pretty good chance of "hitting" your value. There is an intentional delay in the response to limit the load on our database." Another note says: "Please be patient." A "Search Form" note states: "NOTE: This page is limited to 20 queries per one(1) hour time period." The search results for the hash "md5 hash dbf7c981d7efe8c462eab3c39f2b06fd" show the message "Sorry, no solution found." To the right, there's a sidebar with "Follow the Internet Storm Center on Twitter" and a SANS EMEA logo.

Nada

<https://md5hashing.net/hash/md5/>

The screenshot shows a web browser window for the md5hashing.net tool. The URL is https://md5hashing.net/hash/md5/. The page has a sidebar with "Tools" and options like "[ENIDE]CRYPT HASH", "RECENT HASHES LIST", "HASH TYPE CHECKER", "CRYPTOGRAPHY Q&A", "ANONYMOUS EMAIL", and "ANONYMOUS CRYPTO CHAT". The main content area has two sections: "Calculate a Hash" and "Decrypt (search for a match)". In the "Calculate a Hash" section, there's a "Text" input field with "Plain string (text)" and a "Hash it!" button. In the "Decrypt" section, there's a "Hash String" input field containing "dbf7c981d7efe8c462eab3c39f2b06fd" and a "Decode!" button. A message in the "Decrypt" section says: "Reverse decryption is failed. No match found. Try to search via "by all hash types" option. or try later. Sorry...:(

Nada

<http://reversemd5.com/>

md5 and reverse md5 calculator

A small web project by Dany Shaanan. v0.1.0

String to md5:

md5("d41d8cd98f00b204e9800998ecf8427e") = "d41d8cd98f00b204e9800998ecf8427e"

md5 to string:

reverse_md5("dbf7c981d7efe8c462eab3c39f2b06fd") = Sorry, not found.

Fork me on GitHub

Nada

<http://hashtoolkit.com>

Hash Toolkit Home Decrypt MD5 Hash Decrypt SHA1 Hash Generate Hash About

Search in 11,919,818,431 decrypted md5 / sha1 hashes.

Hash: dbf7c981d7efe8c462eab3c39f2b06fd

No hashes found for dbf7c981d7efe8c462eab3c39f2b06fd

Decrypt Hash
Hash Toolkit Hash Decrypter enables you to decrypt / reverse a hash in various formats into their original text. Hashes are often used to store passwords securely in a database. With hash toolkit you could find the original password for a hash.

Supported hashes for decryption:
Usually it's not possible to decrypt a hash, but with hash toolkit you can!

- reverse / decrypt md5 hash
- reverse / decrypt sha1 hash
- reverse / decrypt sha256 hash
- reverse / decrypt sha356 hash
- reverse / decrypt sha512 hash

Generate Hash
Hash Toolkit Hash Generator enables you to generate a

GAMA JEEP DESCUENTOS DE HASTA 30% EN TODA LA GAMA Y 4 AÑOS DE GARANTÍA

Nada

<http://reverse-hash-lookup.online-domain-tools.com/>

The screenshot shows a web browser window with the URL reverse-hash-lookup.online-domain-tools.com. The page is titled "Online Reverse Hash Lookup". On the left, there's a sidebar for "Nessus Vulnerability Scanner" with the text: "Discover why 1M+ IT professionals trust Nessus for vulnerability scanning". The main content area has fields for "Hash values" containing "dbf7c981d7ef:e8c4:62ea:b3c3:9f2b:6fd" and "Hash function" set to "MD5". A green button labeled "Reverse!" is present. To the right, there's a sidebar titled "TOP 10 Tools" listing various security tools, and an advertisement for shutterstock featuring a cooking scene.

Nada

Pues parece que no hemos tenido mucha suerte en reversear el MD5. Bueno, es una IP, ¿no? Y si fuera IPv6, estan en hexadecimal, vamos a probar

[http://\[dbf7:c981:d7ef:e8c4:62ea:b3c3:9f2b:6fd\]/](http://[dbf7:c981:d7ef:e8c4:62ea:b3c3:9f2b:6fd]/)

The screenshot shows a web browser with the URL [http://\[dbf7:c981:d7ef:e8c4:62ea:b3c3:9f2b:6fd\]/](http://[dbf7:c981:d7ef:e8c4:62ea:b3c3:9f2b:6fd]/) in the address bar. The page displays an error message: "No se puede acceder a este sitio web" (The site cannot be reached). Below the message, it says "No se puede acceder a [http://\[dbf7:c981:d7ef:e8c4:62ea:b3c3:9f2b:6fd\]/](http://[dbf7:c981:d7ef:e8c4:62ea:b3c3:9f2b:6fd]/)". A link to search Google for the address is provided, and the error code "ERR_ADDRESS_UNREACHABLE" is shown at the bottom.

Nada, host inalcanzable. Bueno volvamos sobre la idea original, es un hash. Vamos a probar un poco de fuerza bruta con python, así que nos creamos un programa que calcule los hash de todas las direcciones IP a ver si alguno coincide, total solamente son 4.294.967.296 direcciones únicas, vamos, un ratito

El programa que utilizamos es

```
import hashlib
import time

for n1 in range(0,255):
    start_time = time.time()
    for n2 in range(0,255):
        for n3 in range(0,255):
            for n4 in range(0,255):
                m = hashlib.md5()
                ip = str(n1) + "." + str(n2) + "." + str(n3) + "."
str(n4)
                m.update(ip.encode('utf-8'))
                respon = m.hexdigest()
                if m.hexdigest() ==
"dbf7c981d7efe8c462eab3c39f2b06fd":
                    print(ip)
                    print(m.hexdigest())
    elapsed_time = time.time() - start_time
    print(time.strftime("%H:%M:%S", time.gmtime(elapsed_time)))
    print(n1)
```

Y el resultado que nos da en iPython Notebook es

The screenshot shows a Jupyter Notebook interface. At the top, there's a toolbar with File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, Python 3, and Logout buttons. Below the toolbar is a menu bar with File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Run, Cell, Code, and Cell Kernel buttons.

In the main area, there's a code cell labeled "In [*]:" containing the following Python script:

```
import hashlib
import time

for n1 in range(0,255):
    start_time = time.time()
    for n2 in range(0,255):
        for n3 in range(0,255):
            for n4 in range(0,255):
                m = hashlib.md5()
                ip = str(n1) + "." + str(n2) + "." + str(n3) + "." + str(n4)
                m.update(ip.encode('utf-8'))
                respon = m.hexdigest()
                if respon == "dbf7c981d7efe8c462eab3c39f2b06fd":
                    print(ip)
                    print(m.hexdigest())
    elapsed_time = time.time() - start_time
    print(time.strftime("%H:%M:%S", time.gmtime(elapsed_time)))
    print(n1)
```

The output of the script is displayed below the code cell:

```
00:00:41
0
00:00:40
1
```

Unos 40 segundos por cada 16.777.216 de hashes calculados, no está mal, en el peor de los casos, que la ip este al final, unos 170 minutos. A ver una película, que tiempo me da.

Cuando acaba la película vuelvo a mi python y las nubes más negras del averno se ciernen sobre mí, no tengo nada como resultado de mi script excepto una bonita lista de 255 números con sus tiempos. El script no ha funcionado.

N.A. Si, se que podría haber quitado del script todas las IPs de los rangos reservados, que no están en internet, pero no me merecía la pena, el tiempo de espera no era muy grande :). https://en.wikipedia.org/wiki/IPv4#Special-use_addresses

Sabemos que hay varios algoritmos de hash con 16 bytes de salida, o 128 bites, que son

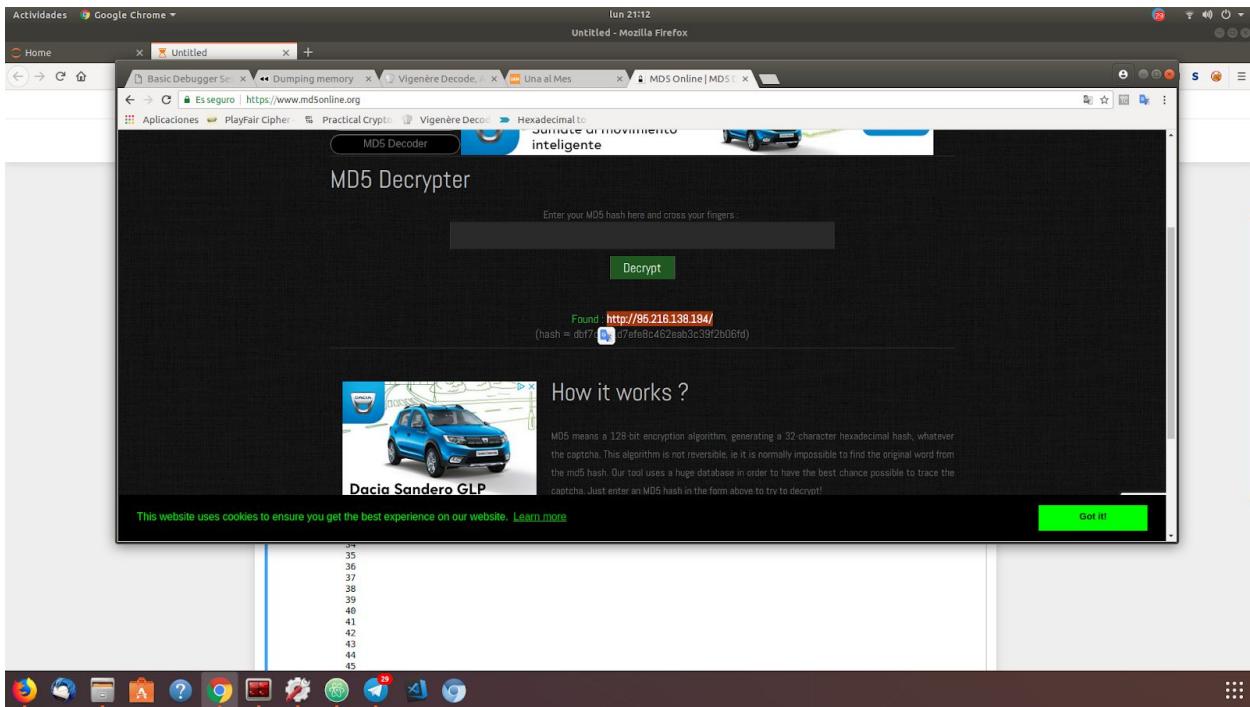
https://en.wikipedia.org/wiki/List_of_hash_functions

Name	Length	Type
<u>MD2</u>	128 bits	hash
<u>MD4</u>	128 bits	hash
<u>MD5</u>	128 bits	<u>Merkle–Damgård construction</u>
<u>RIPEMD</u>	128 bits	hash
<u>RIPEMD-128</u>	128 bits	hash
<u>Snefru</u>	128 or 256 bits	hash
<u>HAVAL</u>	128 to 256 bits	hash

7 algoritmos, algunos más lentos que MD5 en su cálculo, son unas cuantas horas.

Pensamos y pensamos y pensamos en todo lo que nos hemos encontrado en esta parte del reto pero no damos con la solución. El hash no es MD5 de una dirección IP, será alguna otra cosa.

Después de probar de nuevo cifrados clásicos, volvemos a la idea del hash. ¿Y si probamos en algunas páginas más de Internet de reversing de MD5? Parece raro que se haya utilizado otro algoritmo mas exótico. Por último llegamos a <https://www.md5online.org/>



BINGO !!!! Tenemos la dirección IP , al final nuestra intuición era correcta. El pobre Script de python me mira desde Firefox diciendome:

anda, no me dejes con este mal sabor de boca de no haber encontrado la IP,
modificame para que la encuentre, porfa, porfa, pooooooooorrrrrfaaaaaaaaa

a si que asi lo hago:

```
import hashlib
import time

for n1 in range(95,255):
    start_time = time.time()
    for n2 in range(0,255):
        for n3 in range(0,255):
            for n4 in range(0,255):
                m = hashlib.md5()
                ip = str(n1) + "." + str(n2) + "." + str(n3) + "."
                str(n4)
                m.update("http://".encode('utf-8') +
ip.encode('utf-8') + "/".encode('utf-8'))
                respon = m.hexdigest()
                if m.hexdigest() ==
"dbf7c981d7efe8c462eab3c39f2b06fd":
                    print(ip)
                    print(m.hexdigest())
    elapsed_time = time.time() - start_time
    print(time.strftime("%H:%M:%S", time.gmtime(elapsed_time)))
    print(n1)
```

jupyter Untitled Last Checkpoint: 25 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [7]:

```
import hashlib
import time

for n1 in range(95,255):
    start_time = time.time()
    for n2 in range(0,255):
        for n3 in range(0,255):
            for n4 in range(0,255):
                m = hashlib.md5()
                ip = str(n1) + "." + str(n2) + "." + str(n3) + "." + str(n4)
                m.update("http://".encode('utf-8') + ip.encode('utf-8') + "/".encode('utf-8'))
                respon = m.hexdigest()
                if m.hexdigest() == "dbf7c981d7efe8c462eab3c39f2b06fd":
                    print(ip)
                    print(m.hexdigest())
    elapsed_time = time.time() - start_time
    print(time.strftime("%H:%M:%S", time.gmtime(elapsed_time)))
    print(n1)
```

95.216.138.194
dbf7c981d7efe8c462eab3c39f2b06fd

```
KeyboardInterrupt                                     Traceback (most recent call last)
<ipython-input-7-1b0d7338ad42> in <module>()
      8         for n4 in range(0,255):
      9             m = hashlib.md5()
--> 10             ip = str(n1) + "." + str(n2) + "." + str(n3) + "." + str(n4)
     11             m.update("http://".encode('utf-8') + ip.encode('utf-8') + "/".encode('ut
     12             respon = m.hexdigest()

KeyboardInterrupt:
```

In []:

Ya la tenemos y recordar amigos, **“Un script contento es un script que te funcionara mejor la siguiente vez que lo utilices”**

<http://95.216.138.194> Voy a por tiiiiiii !!!!!!!!!!!!!!!

CONTINUARA