Propuesta de solución del reto Una-al-mes de septiembre de 2018 de Hispasec por Ramón Sola (@asterixco)

Este desafío supone el comienzo una nueva etapa en UAM tras el cierre de la trilogía de *La casa de papel* y el descanso veraniego de agosto. Entre septiembre y noviembre, los retos estarán inspirados en la serie *Silicon Valley*.

La investigación de la prueba se ha llevado a cabo en una máquina con Windows.

Enunciado

El primer episodio de esta aventura se plantea de la siguiente manera:

EPISODIO 1

200

Alguien ha denunciado a "El Flautista" por hacer actividades empresariales en una vivienda personal. Necesitamos encontrar a la persona en cuestión para convencerlo de que retire la denuncia o se nos caerá el pelo. El problema es que ha habido un apagón en la incubadora de Erlich y todos los discos duros han muerto menos el de Gilfoyle. En ellos estaban las credenciales de acceso (encriptadas) a la plataforma de la empresa y la única pista del denunciante. Debes conseguir las credenciales de alguno de los archivos de Gilfoyle para entrar y poder encontrar la dirección de la persona que ha montado todo este lío.

- Disco duro de Gilfoyle (escoged el enlace que mejor os venga):

http://www.mediafire.com/file/31pj2a5umpfm345/GILFOYLE-HELLDD.zip

https://mega.nz/#!3IkWlSiK!MkrFIvvt7JBWm-_vrhIv-JFLoNFVh8_dDvFCE-qjKuc

- Login: http://34.247.69.86/siliconvalley/episodio1/login.php

Info: La flag es el número de la casa en formato UAM{md5}

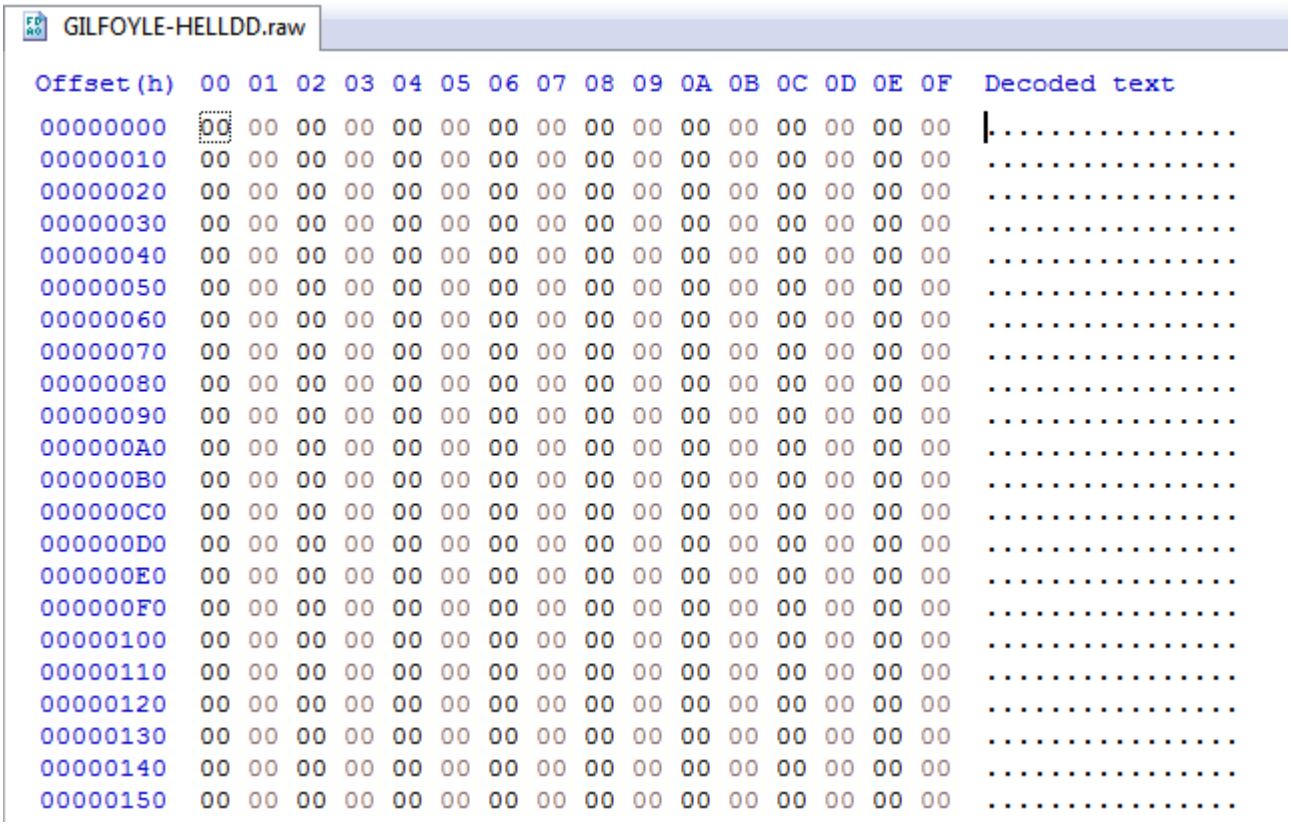
Los enlaces descargan el archivo comprimido `GILFOYLE-HELLDD.zip`, que contiene una supuesta imagen de disco de casi dos gigabytes de tamaño con el nombre `GILFOYLE-HELLDD.raw`, exactamente 2147418112 bytes (2 GB menos 65536).

Primera fase: análisis del fichero RAW

El primer paso consiste en intentar identificar el contenido del fichero RAW obtenido al descomprimir el ZIP. Una aproximación razonable es el uso de la herramienta `file` de línea de comandos existente en muchos sistemas basados en Unix (Linux, BSD, Mac). Si por el contrario se emplea Windows, se puede recurrir al subsistema de Linux que ofrece Windows 10, un entorno tipo Cygwin o el *shell* Bash incluido en la distribución de Git. También, en caso de disponer del entorno de ejecución de Python, el *script* file-magic.py de Didier Stevens es otra opción. Se requiere tener instalado el paquete *python-magic-bin* para Windows a través de `pip install python-magic-bin`.

En este caso, `file GILFOYLE-HELLDD.raw` no consigue reconocer nada y solamente muestra un genérico *data*. Sin embargo, no todo está perdido.

Un visor de ficheros binarios o un editor hexadecimal podrían ayudar a desentrañar el misterio. Se ha usado HxD por ser gratuito, ligero y sencillo pero versátil. Recientemente apareció una versión nueva. Al cargar el fichero RAW, se observa que no hay ninguna cabecera distintiva; hasta el desplazamiento 1000 hexadecimal (4096 decimal), todos los bytes son ceros.

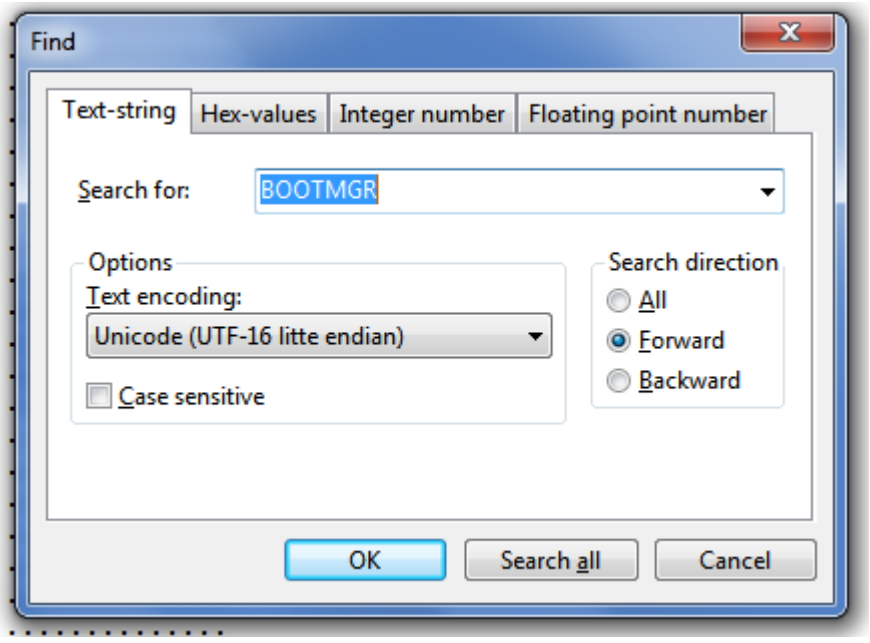


A continuación se ven algunas secuencias de letras mezcladas con otros valores. Parece tratarse de una estructura de datos. Se reconocen las siglas **VBOX** (quizá el fichero se generó en una máquina virtual sobre VirtualBox), **FACP** y **ASL**. Estos dos últimos términos están relacionados con la especificación ACPI, que gobierna la configuración básica y la gestión de energía en los equipos PC actuales.

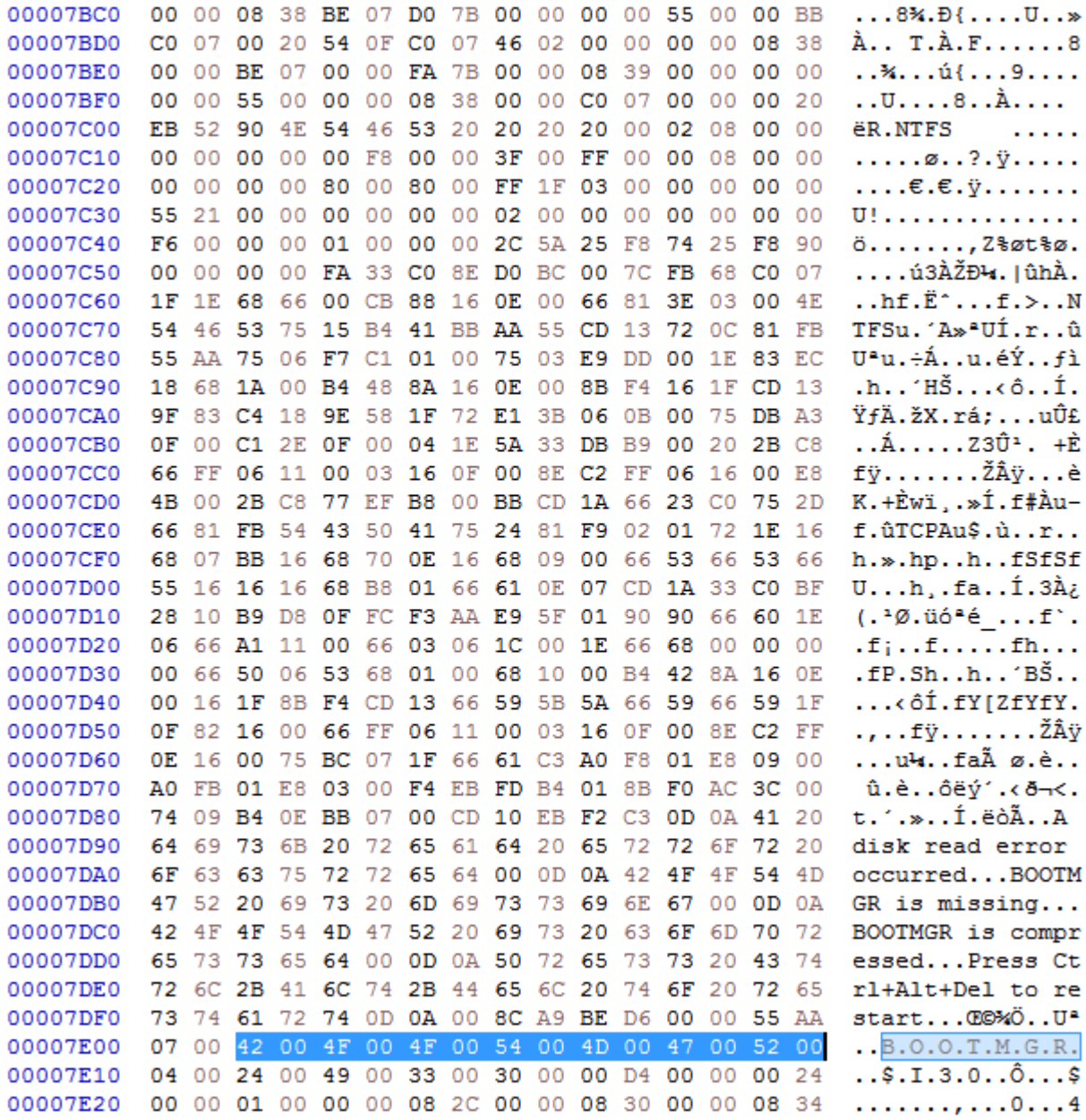
```
00001000  00 C0 D0 FF FF FF FF FF 00 30 D0 FF FF FF FF FF  .ÀÐÿÿÿÿÿ.0Ðÿÿÿÿÿ
00001010  F0 00 FF 7F 00 00 00 00 46 41 43 50 F4 00 00 00  ð.ÿ.....FACPô...
00001020  04 00 56 42 4F 58 20 20 56 42 4F 58 46 41 43 50  ..VBOX  VBOXFACP
00001030  01 00 00 00 41 53 4C 20 61 00 00 00 00 02 FF 7F  ....ASL a.....ÿ.
00001040  70 04 FF 7F 00 00 09 00 2E 44 00 00 A1 A0 00 00  p.ÿ......D..¡ ..
00001050  00 40 00 00 00 00 00 00 04 40 00 00 00 00 00 00  .@.......@......
00001060  00 00 00 00 08 40 00 00 20 40 00 00 00 00 00 00  .....@.. @......
00001070  04 02 00 04 02 00 00 00 65 00 E9 03 00 00 00 00  ........e.é.....
00001080  00 00 00 00 00 03 00 00 41 05 00 00 01 08 00 01  ........A.......
00001090  50 40 00 00 00 00 00 00 10 00 00 00 00 02 FF 7F  P@............ÿ.
000010A0  00 00 00 00 70 04 FF 7F 00 00 00 00 01 20 00 02  ....p.ÿ..... ..
000010B0  00 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .@..............
000010C0  00 00 00 00 01 10 00 02 04 40 00 00 00 00 00 00  .........@......
000010D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000010E0  00 00 00 00 00 00 00 00 01 20 00 03 08 40 00 00  ......... ...@..
000010F0  00 00 00 00 01 10 00 01 20 40 00 00 00 00 00 00  ........ @......
00001100  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001110  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001120  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001130  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Después de todo, no da la impresión de ser una imagen de disco. ¿Dónde está el código de arranque? ¿Dónde está el sistema de archivos? Además, las tablas ACPI solamente residen en la memoria. ¿Y si no es sino un volcado de RAM?

El sector de arranque de partición de un Windows contemporáneo, desde Windows Vista, contiene la cadena **BOOTMGR** (*boot manager*) en codificación UTF-16. También se puede buscar en ASCII al formar parte de dos mensajes de error.



Hay un resultado cercano al inicio. La hipótesis de un sistema Windows era correcta. La ubicación no se debe a la casualidad.



Resulta muy significativo encontrar en el desplazamiento 7C00 hexadecimal la secuencia EB 52 90 (una instrucción de salto corto y un NOP en el conjunto de instrucciones x86) seguida de **NTFS**. El método de arranque basado en la BIOS tradicional carga el MBR (*Master Boot Record*) del disco duro a partir de la dirección física de memoria 7C00 hexadecimal. El código del MBR se copia a sí mismo en otra zona, carga el primer sector de la partición declarada como activa en el mismo desplazamiento 7C00 y le cede el control.

No se trata, pues, de una imagen de disco sino de memoria.

Segunda fase: extracción de información

Existe una herramienta muy versátil para el análisis forense de volcados de memoria: el *framework* Volatility. Se ejecuta sobre Python 2.7, pero no las versiones 3.x. El proyecto está en GitHub y cuenta con una wiki.

El uso más básico de Volatility es `vol.py imageinfo` junto con la opción `-f` para indicar la ubicación del fichero que se va a analizar. De este modo, `vol.py imageinfo -f GILFOYLE-HELLDD.raw` arroja un resultado como el siguiente:

```
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : Win7SP1x64, Win7SP0x64, Win2008R2SP0x64,
Win2008R2SP1x64_23418, Win2008R2SP1x64, Win7SP1x64_23418
                     AS Layer1 : WindowsAMD64PagedMemory (Kernel AS)
                     AS Layer2 : FileAddressSpace (F:\GILFOYLE-HELLDD.raw)
                      PAE type : No PAE
                           DTB : 0x187000L
                          KDBG : 0xf800029f00a0L
          Number of Processors : 1
     Image Type (Service Pack) : 1
                KPCR for CPU 0 : 0xfffff800029f1d00L
             KUSER_SHARED_DATA : 0xfffff78000000000L
           Image date and time : 2018-09-15 09:56:27 UTC+0000
     Image local date and time : 2018-09-15 11:56:27 +0200
```

La imagen de memoria corresponde a un sistema operativo Windows 7 o Windows Server 2008 R2 de 64 bits (x64), pues comparten el mismo kernel y la mayoría de ficheros básicos.

Otra función interesante es `kdbgscan`, ya sabiendo que probablemente parece tratarse de un Windows 7 x64, tal vez con Service Pack 1, lo que se especifica mediante `--profile=Win7SP1x64`. En verdad, se podría probar con cualquier otro perfil indicado en *Suggested Profile(s)*, o incluso omitirlo y confiar en la heurística.

```
Volatility Foundation Volatility Framework 2.6
**************************************************
Instantiating KDBG using: Kernel AS Win7SP1x64 (6.1.7601 64bit)
Offset (V)                    : 0xf800029f00a0
Offset (P)                    : 0x29f00a0
KDBG owner tag check          : True
Profile suggestion (KDBGHeader): Win7SP1x64
Version64                     : 0xf800029f0068 (Major: 15, Minor: 7601)
Service Pack (CmNtCSDVersion) : 1
Build string (NtBuildLab)     : 7601.18247.amd64fre.win7sp1_gdr.
PsActiveProcessHead           : 0xfffff80002a263d0 (40 processes)
PsLoadedModuleList            : 0xfffff80002a446d0 (138 modules)
KernelBase                    : 0xfffff80002801000 (Matches MZ: True)
Major (OptionalHeader)        : 6
```

```
Minor (OptionalHeader)        : 1
KPCR                          : 0xfffff800029f1d00 (CPU 0)

**************************************************
Instantiating KDBG using: Kernel AS Win7SP1x64 (6.1.7601 64bit)
Offset (V)                    : 0xf800029f00a0
Offset (P)                    : 0x29f00a0
KDBG owner tag check          : True
Profile suggestion (KDBGHeader): Win2008R2SP1x64
Version64                     : 0xf800029f0068 (Major: 15, Minor: 7601)
Service Pack (CmNtCSDVersion) : 1
Build string (NtBuildLab)     : 7601.18247.amd64fre.win7sp1_gdr.
PsActiveProcessHead           : 0xfffff80002a263d0 (40 processes)
PsLoadedModuleList            : 0xfffff80002a446d0 (138 modules)
KernelBase                    : 0xfffff80002801000 (Matches MZ: True)
Major (OptionalHeader)        : 6
Minor (OptionalHeader)        : 1
KPCR                          : 0xfffff800029f1d00 (CPU 0)

**************************************************
Instantiating KDBG using: Kernel AS Win7SP1x64 (6.1.7601 64bit)
Offset (V)                    : 0xf800029f00a0
Offset (P)                    : 0x29f00a0
KDBG owner tag check          : True
Profile suggestion (KDBGHeader): Win7SP1x64_23418
Version64                     : 0xf800029f0068 (Major: 15, Minor: 7601)
Service Pack (CmNtCSDVersion) : 1
Build string (NtBuildLab)     : 7601.18247.amd64fre.win7sp1_gdr.
PsActiveProcessHead           : 0xfffff80002a263d0 (40 processes)
PsLoadedModuleList            : 0xfffff80002a446d0 (138 modules)
KernelBase                    : 0xfffff80002801000 (Matches MZ: True)
Major (OptionalHeader)        : 6
Minor (OptionalHeader)        : 1
KPCR                          : 0xfffff800029f1d00 (CPU 0)

**************************************************
Instantiating KDBG using: Kernel AS Win7SP1x64 (6.1.7601 64bit)
Offset (V)                    : 0xf800029f00a0
Offset (P)                    : 0x29f00a0
KDBG owner tag check          : True
Profile suggestion (KDBGHeader): Win2008R2SP0x64
Version64                     : 0xf800029f0068 (Major: 15, Minor: 7601)
Service Pack (CmNtCSDVersion) : 1
Build string (NtBuildLab)     : 7601.18247.amd64fre.win7sp1_gdr.
PsActiveProcessHead           : 0xfffff80002a263d0 (40 processes)
PsLoadedModuleList            : 0xfffff80002a446d0 (138 modules)
KernelBase                    : 0xfffff80002801000 (Matches MZ: True)
Major (OptionalHeader)        : 6
Minor (OptionalHeader)        : 1
KPCR                          : 0xfffff800029f1d00 (CPU 0)

**************************************************
Instantiating KDBG using: Kernel AS Win7SP1x64 (6.1.7601 64bit)
Offset (V)                    : 0xf800029f00a0
```

```
Offset (P)                    : 0x29f00a0
KDBG owner tag check          : True
Profile suggestion (KDBGHeader): Win2008R2SP1x64_23418
Version64                     : 0xf800029f0068 (Major: 15, Minor: 7601)
Service Pack (CmNtCSDVersion) : 1
Build string (NtBuildLab)     : 7601.18247.amd64fre.win7sp1_gdr.
PsActiveProcessHead           : 0xfffff80002a263d0 (40 processes)
PsLoadedModuleList            : 0xfffff80002a446d0 (138 modules)
KernelBase                    : 0xfffff80002801000 (Matches MZ: True)
Major (OptionalHeader)        : 6
Minor (OptionalHeader)        : 1
KPCR                          : 0xfffff800029f1d00 (CPU 0)

**************************************************
Instantiating KDBG using: Kernel AS Win7SP1x64 (6.1.7601 64bit)
Offset (V)                    : 0xf800029f00a0
Offset (P)                    : 0x29f00a0
KDBG owner tag check          : True
Profile suggestion (KDBGHeader): Win7SP0x64
Version64                     : 0xf800029f0068 (Major: 15, Minor: 7601)
Service Pack (CmNtCSDVersion) : 1
Build string (NtBuildLab)     : 7601.18247.amd64fre.win7sp1_gdr.
PsActiveProcessHead           : 0xfffff80002a263d0 (40 processes)
PsLoadedModuleList            : 0xfffff80002a446d0 (138 modules)
KernelBase                    : 0xfffff80002801000 (Matches MZ: True)
Major (OptionalHeader)        : 6
Minor (OptionalHeader)        : 1
KPCR                          : 0xfffff800029f1d00 (CPU 0)
```

Pero lo verdaderamente útil para el propósito de este reto es la búsqueda y extracción de ficheros mapeados en memoria y almacenados en caché. La función `filescan` explora los objetos de ficheros (*file objects*) de la memoria del kernel. Como la lista puede ser muy larga, se emplea también la opción `--output-file` para escribirla en un fichero de texto. No hay que olvidar `--profile`. Por ejemplo: `vol.py filescan -f GILFOYLE-HELLDD.raw --profile=Win7SP1x64 --output-file=ficheros.txt`.

```
Volatility Foundation Volatility Framework 2.6
Outputting to: ficheros.txt
```

Estas son las primeras líneas del fichero de texto:

```
Offset(P)           #Ptr   #Hnd Access Name
------------------ ------ ------ ------ ----
0x000000007e6015e0     12      0 R--r-d \Device\HarddiskVolume2\Program Files
(x86)\OpenOffice 4\program\lng.dll
0x000000007e603340      2      1 ------ \Device\Afd\Endpoint
0x000000007e603f20      1      1 R--rw-
\Device\HarddiskVolume2\Windows\winsxs\x86_microsoft.vc90.crt_1fc8b3b9a1e18e3b_9.0
.30729.6161_none_50934f2ebcb7eb57
```

```
0x000000007e604740      1      1 R--rw-
\Device\HarddiskVolume2\Windows\winsxs\amd64_microsoft.windows.common-
controls_6595b64144ccf1df_6.0.7601.17514_none_fa396087175ac9ac
0x000000007e604890      2      0 R--r--
\Device\HarddiskVolume2\Windows\System32\Microsoft\Protect\S-1-5-19\Preferred
0x000000007e604c80      2      1 ------ \Device\Afd\Endpoint
0x000000007e6083b0      2      0 R--r--
\Device\HarddiskVolume2\ProgramData\Microsoft\Windows\DRM\blackbox.bin
0x000000007e608530      2      1 ------ \Device\Afd\Endpoint
0x000000007e609870      1      1 R--rw-
\Device\HarddiskVolume2\Windows\winsxs\x86_microsoft.vc90.crt_1fc8b3b9a1e18e3b_9.0
.30729.6161_none_50934f2ebcb7eb57
0x000000007e609f20      2      0 R--rwd
\Device\HarddiskVolume2\Users\Public\Pictures\Sample Pictures\desktop.ini
```

Para una visualización más cómoda, conviene ordenar la tabla por la ruta. En Windows es posible mediante `sort` y su parámetro `/+N`, donde *N* es el número de columna a partir de la cual se ordenará. La ruta empieza en la columna 41: `sort /+41 ficheros.txt > ordenados.txt`.

Una inspección minuciosa no pasará por alto el fichero `info.odt` del escritorio.

```
x000000007fb0e170      2      1 R--rwd
\Device\HarddiskVolume2\Users\unaalmes\Desktop
0x000000007e9dd520      2      1 R--rwd
\Device\HarddiskVolume2\Users\unaalmes\Desktop
0x000000007fca5580      2      0 RW-rw-
\Device\HarddiskVolume2\Users\unaalmes\Desktop\.~lock.info.odt#
0x000000007ebfbbe0     16      0 R--rwd
\Device\HarddiskVolume2\Users\unaalmes\Desktop\desktop.ini
0x000000007e9d8af0     12      0 R--r-d
\Device\HarddiskVolume2\Users\unaalmes\Desktop\DumpIt_1734677328.exe
0x000000007e82ca10     16      0 R--rw-
\Device\HarddiskVolume2\Users\unaalmes\Desktop\IDA Pro (32-bit).lnk
0x000000007e82b180      2      0 R--rw-
\Device\HarddiskVolume2\Users\unaalmes\Desktop\IDA Pro (64-bit).lnk
0x000000007fcabd50      1      1 RW-r--
\Device\HarddiskVolume2\Users\unaalmes\Desktop\info.odt
0x000000007fafb440      2      1 R--rwd
\Device\HarddiskVolume2\Users\unaalmes\Desktop\RamCapturer
0x000000007fb02430      2      1 R--rwd
\Device\HarddiskVolume2\Users\unaalmes\Desktop\RamCapturer
```

Evidentemente, en un caso real, el fichero con más posibilidades no destacará tanto y habrá que hacer conjeturas y varios intentos.

¿Podría esconder información útil? Para extraerlo de la imagen de memoria, si está disponible, se emplea la opción `-Q` con el valor de la primera columna: 0x000000007fcabd50. También es obligatorio especificar el directorio de destino mediante `--dump-dir`. Por último, `-n` es opcional pero sirve para distinguir mejor los archivos extraídos si son varios, pues añade el nombre original al

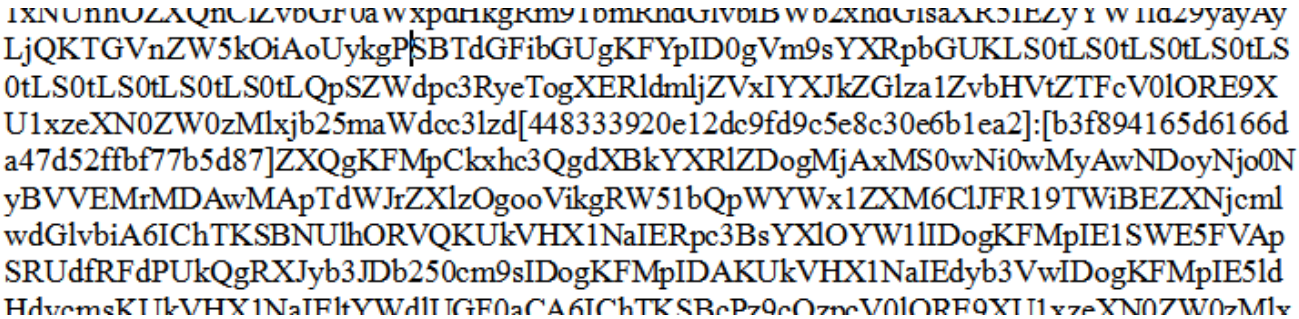
nombre del fichero de destino, que es un poco críptico. Así, `vol.py dumpfiles -f GILFOYLE-HELLDD.raw --profile=Win7SP1x64 -Q 0x000000007fcabd50 --dump-dir=. -n` generará un fichero `file.None.0xfffffa8001acdf10.info.odt.dat`.

```
Volatility Foundation Volatility Framework 2.6
DataSectionObject 0x7fcabd50   None
\Device\HarddiskVolume2\Users\unaalmes\Desktop\info.odt
```

El fichero obtenido tiene un tamaño de 40960 bytes. Si se abre con un descompresor, se comprueba que se trata un archivo ZIP con la estructura de ficheros esperada para un documento ODT (OpenDocument Text). La herramienta `file` así lo identifica también. Se puede intentar abrir con el procesador de textos OpenOffice Writer (también vale Microsoft Word), una acción no exenta de ciertos riesgos, enviarlo a un servicio *online* de edición de documentos como los de Google o Microsoft, o examinar directamente su estructura interna. El contenido más apetecible quizá se encuentre en `content.xml`.

Bien en el procesador de textos o despojado de su envoltura XML, el contenido del documento es en apariencia un galimatías codificado en Base64, realmente un fragmento de un documento técnico sobre el *malware* Stuxnet probablemente para despistar. Sin embargo, una investigación atenta revela el mensaje misterioso `[448333920e12dc9fd9c5e8c30e6b1ea2]: [b3f894165d6166da47d52ffbf77b5d87]`.

Content.xml en el bloc de notas



Documento abierto en el procesador de textos

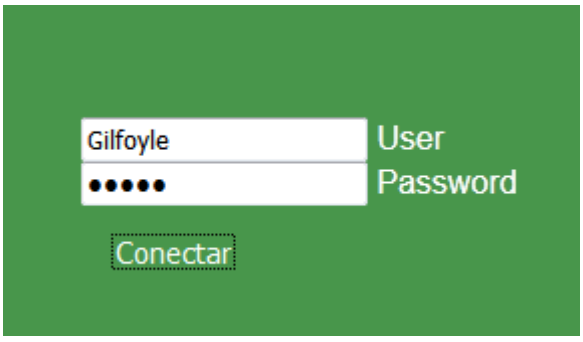
Se sospecha que sean *hashes*. Sitios web como https://hashkiller.co.uk/md5-decrypter.aspx, https://crackstation.net y http://md5decrypt.net/en/ confirman que corresponden a *Gilfoyle* y *Satan* respectivamente.

```
448333920e12dc9fd9c5e8c30e6b1ea2 MD5 : Gilfoyle
b3f894165d6166da47d52ffbf77b5d87 MD5 : Satan
```

Tercera fase: login y búsqueda de la dirección

El enunciado del reto señalaba una página para hacer *login* en un servidor web con un sencillo formulario de conexión.

Parece sensato introducir *Gilfoyle* como nombre de usuario y *Satan* como contraseña.



Efectivamente, son correctos.

```
Denuncia recibida: https://drive.google.com/open?id=10iguWjRmx3mB0Y4g9iRrJOIXZ1HIJ_zC
```

El enlace dirige a un fichero de imagen `denuncia.jpeg` que se ve así:

JUZGADO DE INSTRUCCION Nº 2

PLAZA CASTILLA,1
Teléfono: Fax:
Número de Identificación Único:

DILIGENCIAS PREVIAS PROC. ABREVIADO

Procurador/a: SIN PROFESIONAL ASIGNADO
Representado:

PROVIDENCIA DEL MAGISTRADO-JUEZ

SR.

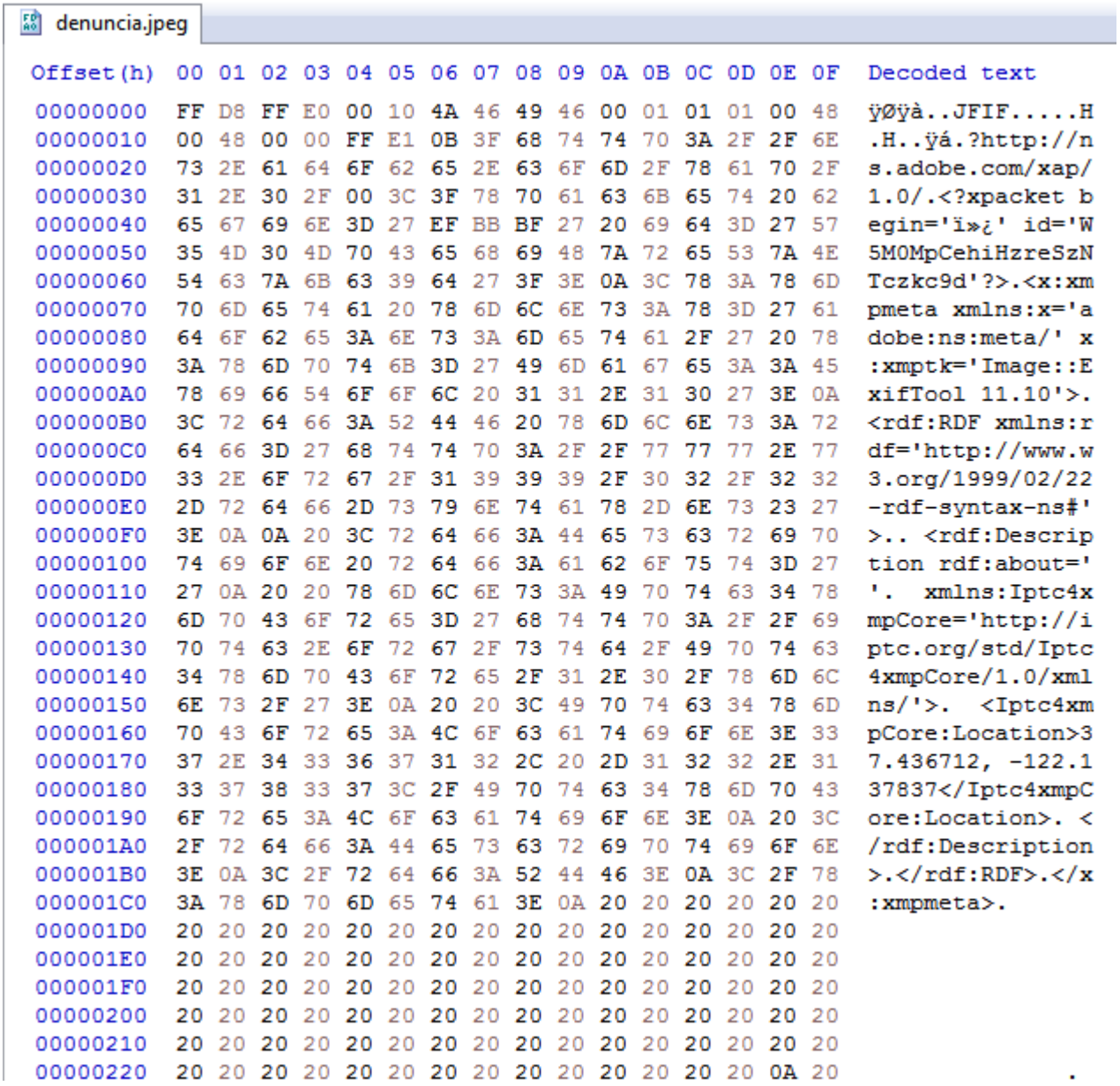
En , a

Vista la anterior diligencia se tiene por personado y parte en las mismas al bajo la dirección letrada de D. en nombre y representación de y al propio tiempo, dese traslado de las actuaciones al Procurador por medio de copia de las mismas, para que, conforme a lo dispuesto en el artículo 784, 1º de la Ley de Enjuiciamiento Criminal, presente escrito de defensa en el plazo de **diez días** frente a las acusaciones formuladas, con la prevención de que en caso de no verificarlo se entenderá que se opone a las actuaciones y seguirá su curso el procedimiento sin perjuicio de la responsabilidad en que pueda incurrir.

MODO DE IMPUGNACION: mediante interposición de recurso de reforma en el plazo de tres días ante este órgano judicial.

Lo mandó y firma S.Sª. Doy fe.-

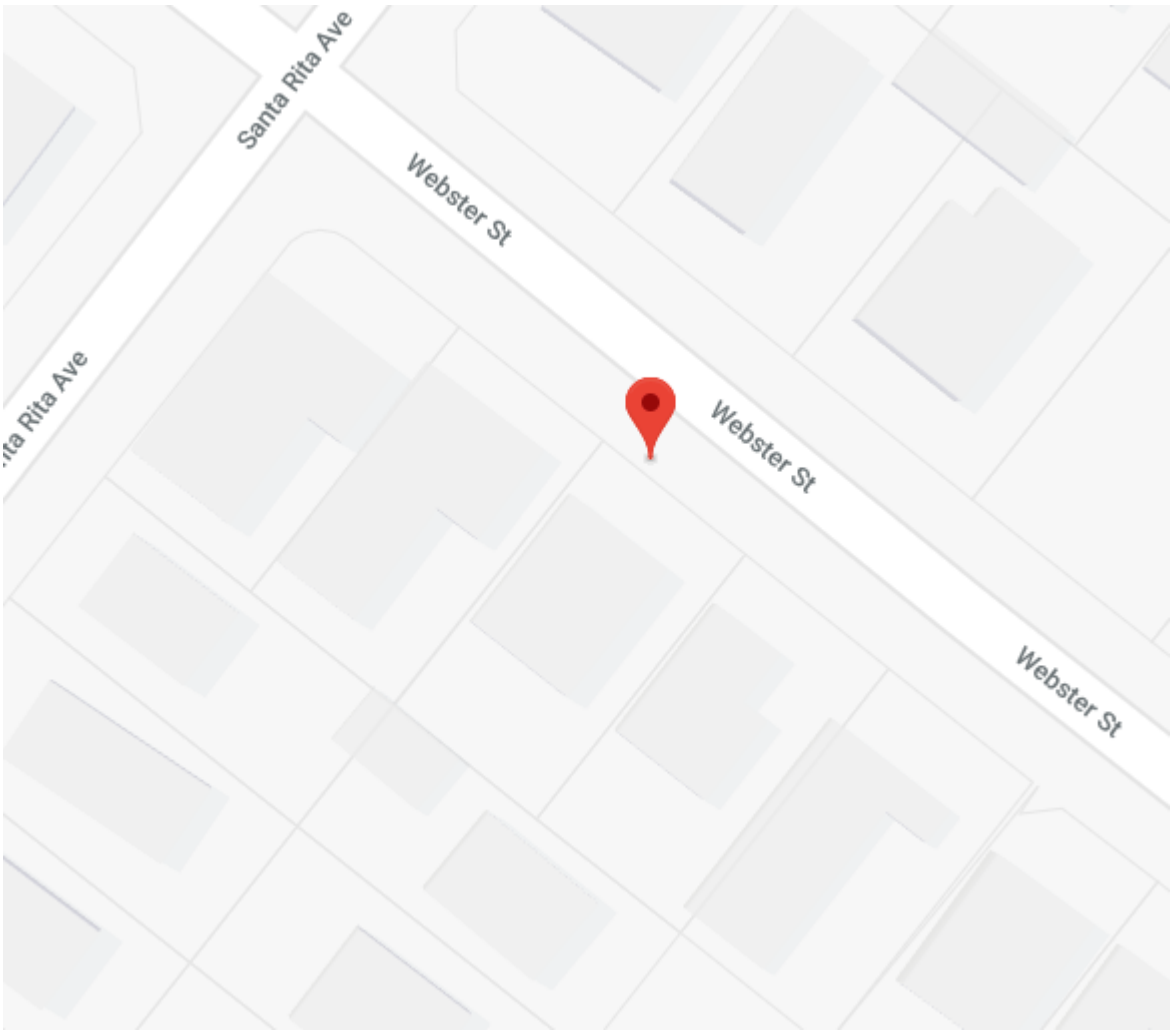
Un documento legal con unos datos censurados y contenido irrelevante. ¿Dónde se puede esconder la información requerida? La hipótesis de la esteganografía no es descabellada, pero hay una aproximación más simple: metadatos. En consecuencia, editor hexadecimal e inspección visual.



denuncia.jpeg

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	FF	D8	FF	E0	00	10	4A	46	49	46	00	01	01	01	00	48	ÿØÿà..JFIF.....H
00000010	00	48	00	00	FF	E1	0B	3F	68	74	74	70	3A	2F	2F	6E	.H..ÿá.?http://n
00000020	73	2E	61	64	6F	62	65	2E	63	6F	6D	2F	78	61	70	2F	s.adobe.com/xap/
00000030	31	2E	30	2F	00	3C	3F	78	70	61	63	6B	65	74	20	62	1.0/.<?xpacket b
00000040	65	67	69	6E	3D	27	EF	BB	BF	27	20	69	64	3D	27	57	egin='ï»¿' id='W
00000050	35	4D	30	4D	70	43	65	68	69	48	7A	72	65	53	7A	4E	5M0MpCehiHzreSzN
00000060	54	63	7A	6B	63	39	64	27	3F	3E	0A	3C	78	3A	78	6D	Tczkc9d'?>.<x:xm
00000070	70	6D	65	74	61	20	78	6D	6C	6E	73	3A	78	3D	27	61	pmeta xmlns:x='a
00000080	64	6F	62	65	3A	6E	73	3A	6D	65	74	61	2F	27	20	78	dobe:ns:meta/' x
00000090	3A	78	6D	70	74	6B	3D	27	49	6D	61	67	65	3A	3A	45	:xmptk='Image::E
000000A0	78	69	66	54	6F	6F	6C	20	31	31	2E	31	30	27	3E	0A	xifTool 11.10'>.
000000B0	3C	72	64	66	3A	52	44	46	20	78	6D	6C	6E	73	3A	72	<rdf:RDF xmlns:r
000000C0	64	66	3D	27	68	74	74	70	3A	2F	2F	77	77	77	2E	77	df='http://www.w
000000D0	33	2E	6F	72	67	2F	31	39	39	39	2F	30	32	2F	32	32	3.org/1999/02/22
000000E0	2D	72	64	66	2D	73	79	6E	74	61	78	2D	6E	73	23	27	-rdf-syntax-ns#'
000000F0	3E	0A	0A	20	3C	72	64	66	3A	44	65	73	63	72	69	70	>.. <rdf:Descrip
00000100	74	69	6F	6E	20	72	64	66	3A	61	62	6F	75	74	3D	27	tion rdf:about='
00000110	27	0A	20	20	78	6D	6C	6E	73	3A	49	70	74	63	34	78	'. xmlns:Iptc4x
00000120	6D	70	43	6F	72	65	3D	27	68	74	74	70	3A	2F	2F	69	mpCore='http://i
00000130	70	74	63	2E	6F	72	67	2F	73	74	64	2F	49	70	74	63	ptc.org/std/Iptc
00000140	34	78	6D	70	43	6F	72	65	2F	31	2E	30	2F	78	6D	6C	4xmpCore/1.0/xml
00000150	6E	73	2F	27	3E	0A	20	20	3C	49	70	74	63	34	78	6D	ns/'>. <Iptc4xm
00000160	70	43	6F	72	65	3A	4C	6F	63	61	74	69	6F	6E	3E	33	pCore:Location>3
00000170	37	2E	34	33	36	37	31	32	2C	20	2D	31	32	32	2E	31	7.436712, -122.1
00000180	33	37	38	33	37	3C	2F	49	70	74	63	34	78	6D	70	43	37837</Iptc4xmpC
00000190	6F	72	65	3A	4C	6F	63	61	74	69	6F	6E	3E	0A	20	3C	ore:Location>. <
000001A0	2F	72	64	66	3A	44	65	73	63	72	69	70	74	69	6F	6E	/rdf:Description
000001B0	3E	0A	3C	2F	72	64	66	3A	52	44	46	3E	0A	3C	2F	78	>.</rdf:RDF>.</x
000001C0	3A	78	6D	70	6D	65	74	61	3E	0A	20	20	20	20	20	20	:xmpmeta>.
000001D0	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	
000001E0	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	
000001F0	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	
00000200	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	
00000210	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	
00000220	20	20	20	20	20	20	20	20	20	20	20	20	20	20	0A	20	.

Hay unas coordenadas camufladas en el campo `Iptc4xmpCore:Location`. Si se consulta *37.436712, -122.137837* en Google Maps, se llega a una calle de la localidad californiana de Palo Alto, en pleno Silicon Valley. El marcador de posición señala una casa concreta.



El siguiente paso es conmutar a Street View en dicho punto.

Al ampliar la imagen de la puerta se ve el número 2126, como corrobora la ficha.

De este modo, tan solo hay que calcular el *hash* MD5 de la cadena “2126”. Por ejemplo, el buscador DuckDuckGo da la respuesta de forma inmediata con la simple consulta md5 2126: *3b92d18aa7a6176dd37d372bc2f1eb71*. Por tanto, la *flag* y solución del reto es **UAM{3b92d18aa7a6176dd37d372bc2f1eb71}**.