

## Título: Análisis de estructuras de ficheros LNK (Análisis Forense) v1.2

Autor: Andrés Tarasco Acuña: atarasco@gmail.com

### Introducción:

Cada vez que un usuario instala o ejecuta una aplicación en el sistema se generan una serie de entradas de datos en el sistema de ficheros, en la carpeta *prefetch* y en el registro que nos permiten identificar que una aplicación ha estado instalada en nuestro sistema.

El sistema operativo también crea unos ficheros con extensión LNK (accesos directos) que se encargan de recopilar información de nuestro sistema. Entre esta información, que se actualizada cada vez que se accede a la aplicación o se modifica la ruta, se encuentra la siguiente:

- Ruta de la aplicación
- Ruta relativa
- Icono del fichero LNK
- Fecha de acceso (fichero de destino)
- Fecha de creación (fichero de destino)
- Fecha de modificación (fichero de destino)
- Tamaño (fichero de destino)
- Ubicación (red o local)
- Identificador de disco (numero de serie del disco duro en el que residía el fichero)

Un análisis exhaustivo sobre el contenido de una imagen o de un dispositivo de disco duro nos dará información sobre que ha estado ejecutando el usuario a lo largo del tiempo y a que documentos a accedido.

La herramienta que se presenta como prueba de concepto [1] con este *paper*, es capaz de analizar un dispositivo de disco y extraer un listado de todas las referencias a ficheros, incluso aquellas que se encuentren en clusters no asignados.

La indexación y tratamiento de toda esta información nos permitirá correlar de forma mas efectiva lo sucedido en el sistema sin la

necesidad de recurrir a otros productos de software como **Encase** [2].

```
D:\Programación\forensics>Ink -o
"Inks\WinHex.lnk"
(c) 2006 Andres Tarasco Acuña -
atarasco@gmail.com

[+] Block Device Inks\WinHex.lnk opened
(444 Bytes)
Analizando...

CTIME: 08/03/2006 16:30
MTIME: 08/03/2006 16:30
ATIME: 19/01/2006 11:07
File Length: 1380864
Volume Type: Fixed (Hard disk)
Serial Number: 00ed-b0f8
Path: D:\Winhex\WinHex.exe
WORKINGDIRECTORY: d:\Winhex
HOSTNAME: redbull
```

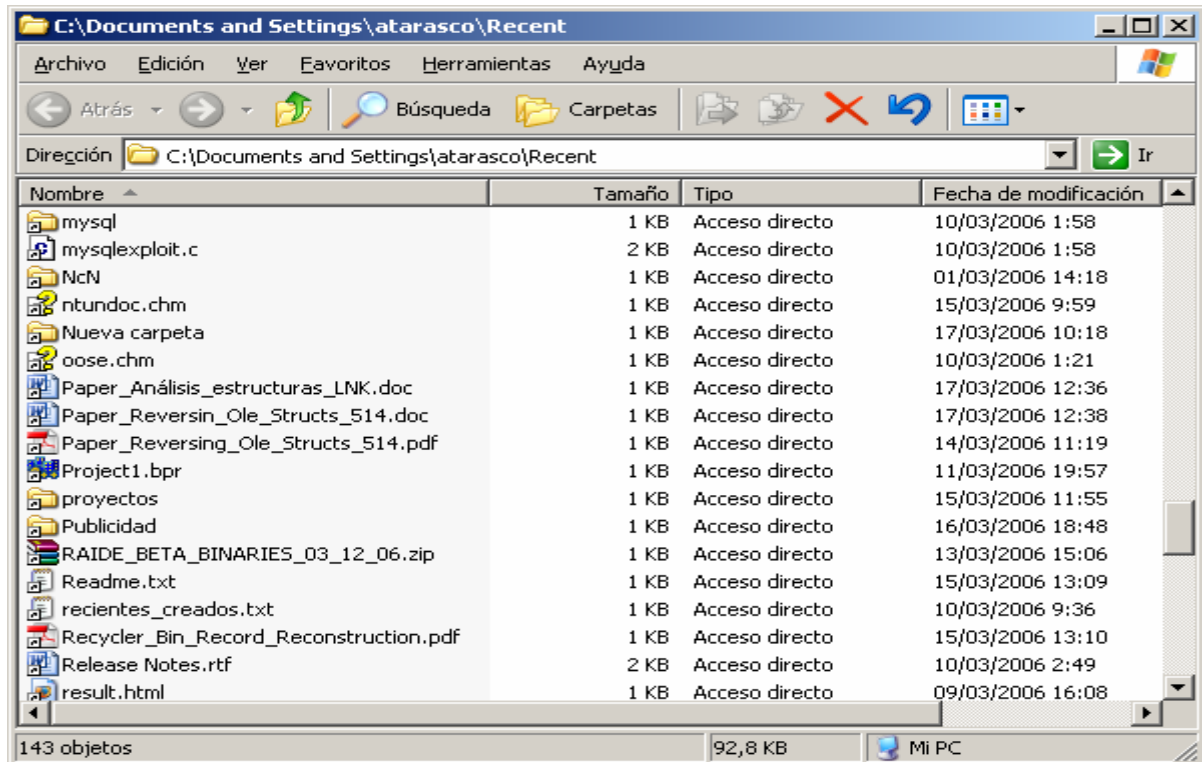
Todo fichero “.LNK” contiene una cabecera común de 76 bytes. El contenido de esta cabecera nos permitirá identificar que estructuras datos tendremos a lo largo del fichero.

```
4C 00 00 00 01 14 02 00 00 00 00 00 C0 00 00 00
00 00 00 46 93 00 00 00 20 00 00 00 7C 59 29 DE
8A 43 C6 01 7C 59 29 DE 8A 43 C6 01 40 5D 83 E4
8A 43 C6 01 69 17 00 00 00 00 00 00 01 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 01 01 14 00
1F 50 E0 4F D0 20 EA 3A 69 10 A2 D8 08 00 2B 30
```

```
struct _lnkHeader { //0x4c
    DWORD Header;
    LNKGUID guid;
    DWORD flags;
    DWORD FileAttributes;
    FILETIME CTIME;
    FILETIME MTIME;
    FILETIME ATIME;
    DWORD filelen;
    DWORD IconNumber;
    DWORD ShowWnd;
    DWORD HotKey;
    DWORD Unknowna;
    DWORD Unknownb;
};
```

## Análisis de la Cabecera de un fichero LNK:

Toda cabecera de ficheros LNK comienza con un DWORD que indica el tamaño de la cabecera. Este valor es siempre el mismo por lo que podremos utilizar este valor, 0x0000004c, para identificar ficheros .LNK en nuestro sistema.



Entradas de ficheros LNK creadas en la carpeta Recent

Al tamaño de la cabecera le sigue el parámetro uid, cuyo tamaño son 16 bytes y que se representa de la siguiente forma {00021401-0000-0000-00c0-000000000046}. Este uid hasta la fecha es único y es referenciado desde las siguientes claves del registro:

```

struct _guid {
    DWORD one;
    WORD two;
    WORD three;
    unsigned char four[2];
    unsigned char five[6];
};
    
```

- HKEY\_CLASSES\_ROOT\.lnk\ShellEx\\*
- HKEY\_CLASSES\_ROOT\CLSID\{00021401-0000-0000-C000-000000000046}

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	4C	00	00	00	01	14	02	00	00	00	00	00	C0	00	00	00	L... ..À...
00000010	00	00	00	46	93	00	00	00	20	00	00	00	7C	59	29	DE	...F ... .. Y)P
00000020	8A	43	C6	01	7C	59	29	DE	8A	43	C6	01	40	5D	83	E4	CÆ. Y)P CÆ.@]!ä
00000030	8A	43	C6	01	69	17	00	00	00	00	00	00	01	00	00	00	CÆ.i.....
00000040	00	00	00	00	00	00	00	00	00	00	00	00	01	01	14	00	.....
00000050	1F	50	E0	4F	D0	20	EA	3A	69	10	A2	D8	08	00	2B	30	.PàOè:!.cø...+0

UUID en la cabecera de un fichero .lnk

El parámetro, flags (0x00000093), es el que determinara el orden de los datos del resto de fichero. Dependiendo de los flags podremos encontrar diferentes estructuras y la forma de interpretarlas será diferente. A continuación se describe como identificar las siguientes estructuras:

- (Header.flags & 1): La estructura ShellIDItem se encuentra a continuación.
- (Header.flags & 2): El fichero LNK tiene una estructura FILELOCATION.
- (Header.flags & 4): El fichero LNK tiene la estructura con la descripción de la aplicación.
- (Header.flags & 8): El fichero LNK tiene una estructura con el path relativo al fichero
- (Header.flags & 16): Existe una estructura en la que está definido el directorio de trabajo
- (Header.flags & 32): Existe una estructura con parámetros pasados a la aplicación
- (Header.flags & 64): Existe una estructura en la que está definido el icono a mostrar.
- (Header.flags & 128): Flag UNICODE, indicando que los datos se almacenan en unicode.
- (Header.flags & 256): Estructura 256 (En Unicode?)
- (Header.flags & 512) : Estructura 512 (En Unicode?)
- (Header.flags & 1024) : Estructura 1024 (En Unicode?)
- (Header.flags & 2048) : Estructura con el GUID del producto de Software.
- (Header.flags & 4096) : Estructura con el GUID del paquete msi
- (Header.flags & 8192) : Estructura 8192 (En Unicode?)
- (Header.flags & 16384) : Estructura 16384 (En Unicode?)

Los flags 256 – 16384 han sido incorporados por Microsoft en las ultimas versiones de sus sistemas operativos y contienen información extra cuya estructura se detallará mas adelante.

Otro campo importante que se encuentra en la cabecera de un fichero lnk es la variable Fileattribute. Se trata de una variable de 32 bits (DWORD) que almacena información sobre el fichero de destino al que apunta el .lnk.

Esta información es proporcionada por el sistema de archivos (FAT/NTFS). La presencia de un flag de un sistema de archivos diferente al existente en la unidad en la que reside el archivo, es indicativo de que el fichero a sido movido desde una ubicación remota. El análisis de los flags de la variable fileattribute se detalla a continuación:

<b>1:</b>	READ_ONLY	<b>128:</b>	NORMAL
<b>2:</b>	HIDDEN	<b>256:</b>	TEMPORARY
<b>4:</b>	SYSTEM FILE	<b>512:</b>	SPARSE
<b>8:</b>	VOLUME_LABEL	<b>1024:</b>	REPARSE_POINT
<b>16:</b>	DIRECTORY	<b>2048:</b>	COMPRESSED
<b>32:</b>	MODIFIED_SINCE_BACKUP	<b>4096:</b>	OFFLINE
<b>64:</b>	ENCRYPTED (EFS)	<b>16384:</b>	UNUSED

Las siguientes 3 variables de la cabecera tienen 8 bytes, muestran las fechas de creación, modificación y acceso al fichero de destino en formato FILETIME. Estas fechas no siempre contiene valores válidos (en algunas ocasiones están inicializadas a 0, sobre todo cuando no existe la estructura FileLocation).

Dentro de las propiedades del fichero LNK también se especifica el número de icono que se utilizará en la aplicación de destino, utilizado cuando no existe la estructura de icono, y representado con un DWORD.

Las dos propiedades siguientes especifican de qué forma se mostrará la aplicación de destino al ejecutar el acceso directo (ShowWnd), permitiendo que la aplicación se ejecute oculta o minimizada, y una tecla de acceso rápido, que especifica una combinación de teclas válida para ejecutar la aplicación. Ambas variables ocupan 4 bytes (DWORD)

Análisis del parámetro de ventana y que influirá en la visualización de la aplicación de destino:

0: SW\_HIDE  
1: SW\_NORMAL  
2: SW\_SHOWMINIMIZED  
3: SW\_SHOWMAXIMIZED

Por ultimo tenemos dos variables inicializadas a 0 y que en algunas ocasiones, la última de ellas puede contener valores. Su funcionalidad es desconocida.

### **Estructura ShellID:**

---

Esta estructura contiene información, en ASCII y Unicode de la ruta de la aplicación u objeto de destino. En el caso de ficheros residentes en el disco local, esta ruta se encuentra dividida en fragmentos con nombres DOS (8.3) y NTFS.

Su existencia está supeditada a la presencia del flag 0x01 en la cabecera (Header.flags & 1).

El primer DWORD de la estructura ShellID determina el tamaño total de la estructura. Este offset será utilizado para llegar al principio de la siguiente estructura (según venga definida en los flags de la variable fileattribute de la cabecera.

El contenido de la estructura ShellID es una lista de elementos SHITEMID [3]. Su contenido se muestra a continuación:

```
typedef struct _SHITEMID {  
    USHORT cb;  
    BYTE abID[1];  
} SHITEMID;
```

A continuación se muestra un ejemplo de una estructura ShellID de 47 bytes (0x002f) que contiene dos elementos SHITEMID de tamaño 20 (0x14) y 25 (0x19) bytes.

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	4C	00	00	00	01	14	02	00	00	00	00	00	C0	00	00	00	I
00000010	00	00	00	46	83	00	00	00	10	00	00	00	00	18	11	80	ShellID Size (0x2f)
00000020	97	E7	A8	01	00	18	11	80	97	E7	A8	01	00	18	11	80	c''...  c''...
00000030	97	E7	A8	01	00	00	00	00	00	00	00	00	01	00	00	00	SHITEMID (len: 14bytes)
00000040	00	00	00	00	00	00	00	00	00	00	00	00	2F	00	14	00	...../...
00000050	1F	50	E0	4F	D0	20	EA	3A	69	10	A2	D8	08	00	2B	30	.PàOè é:i.cø...+0
00000060	30	9D	19	00	2F	46	3A	5C	00	00	00	00	00	00	00	00	0 .../F:\.....
00000070	00	00	00	00	00	00	00	00	00	00	00	00	00	32	00	00	.....2..
00000080	00	1C	00	00	00	01	00	00	00	1C	00	00	00	2D	00	00	.....-
00000090	00	00	00	00	00	31	00	00	00	11	00	00	00	02	00	00	.....1.....
000000A0	00	13	99	58	2C	10	00	00	00	00	46	3A	5C	00	00	00	.. X.....F:\...
000000B0	00	00	00														....

### Estructura ShellID

El ultimo elemento de la lista tendrá como longitud 0 (un USHORT con valor 0x0000).

En el análisis de un fichero LNK descartaremos la estructura SHELLID por no considerarla relevante para la indexación de los datos aunque, durante un análisis manual, debemos consultarla sobre todo cuando no se encuentra la estructura FileLocation.

### Estructura FileLocation:

La estructura Filelocation, presente cuando el flag 2 de la cabecera del fichero LNK está indicado. Esta estructura recopila información sobre el objeto a ejecutar, siempre y cuando este se trate de un fichero o directorio (por ejemplo no está presente cuando se trata de links http )

La disposición de los datos contenidos en la estructura FileLocation se muestra a continuación

```

struct _FileLocation {
    WORD size;
    WORD sizeHigh;
    DWORD FirstOffset;
    WORD Flags;
    WORD FlagHigh;
    DWORD OffsetLocalVolumeInfo;
    DWORD OffsetBasePathname;
    DWORD OffsetNetworkVolumeInfo;
    WORD OffsetRemainingPathname;
    WORD OffsetRemainingPathnameHigh;
};

```

La primera variable de 4 bytes indica el tamaño de la estructura (aparentemente los últimos 4 bytes no son utilizados) y a continuación le sigue la posición, relativa a esta estructura, en la que encontraremos la próxima estructura de datos (FirstOffset). El contenido de la próxima estructura vendrá determinado por la variable Flags de la estructura FileLocation. Esta variable indicará si el fichero o directorio al que apunta el fichero LNK está contenido en un dispositivo de almacenamiento local o si por el contrario se encuentra en un dispositivo de red.

```
#define LOCALFILE 1
```

```
#define NETWORKFILE 2
```

Las estructuras de datos, dependientes de la ubicación física del fichero referenciado en el LNK, y por tanto, del flag mencionado en el punto anterior, son:

```
struct _LocalVolumeTable {
    DWORD size;
    DWORD VolumeType;
    WORD VolumeSerialNumberLow;
    WORD VolumeSerialNumberHigh;
    DWORD VolumeNameOffset;
    unsigned char *name;
};
```

```
struct _NetworkVolumeTable {
    DWORD size
    DWORD NetworkType
    DWORD NetworkShareOffset
    DWORD NetworkLetterOffset
    DWORD Unknown
    unsigned char *name
}
```

**SubEstructura LocalVolume:**

En el caso de que la variable Flags valga 1, la estructura filelocation contendrá la subestructura LocalVolumeTable, con información del fichero almacenado en un disco local. Esta estructura se encuentra en la posición FileLocation.OffsetLocalVolumeInfo.

	LocalVolume Offset	Size: 4A	Pathname OFFSET	flags: LocalVolume	
00000130	00 16 00 00 00	4A 00 00 00	1C 00 00 00	01 00 00	J
00000140	00 1C 00 00 00	2D 00 00 00	00 00 00 00	49 00 00	- I
00000150	00 11 00 00 00	03 00 00 00	29 00 59 04	10 00 00	) Y
00000160	00 00 43 3A 5C 57 49	4E 44 4F 57	53 5C 73 79 73		C:\WINDOWS\sys
00000170	74 65 6D 33 32 5C 63	6D 64 2E 65	78 65 00 00	27	tem32\cmd.exe

Destination Flags HD Serial Number  
Estructura Filelocation con LocalVolumeTable

El primer DWORD (11 00 00 00) muestra el tamaño de la estructura (LocalVolumeTable->size). El segundo DWORD, LocalVolumeTable->VolumeType, indica el tipo de dispositivo al que apunta el fichero LNK (03 00 00 00). Sus posibles valores son:

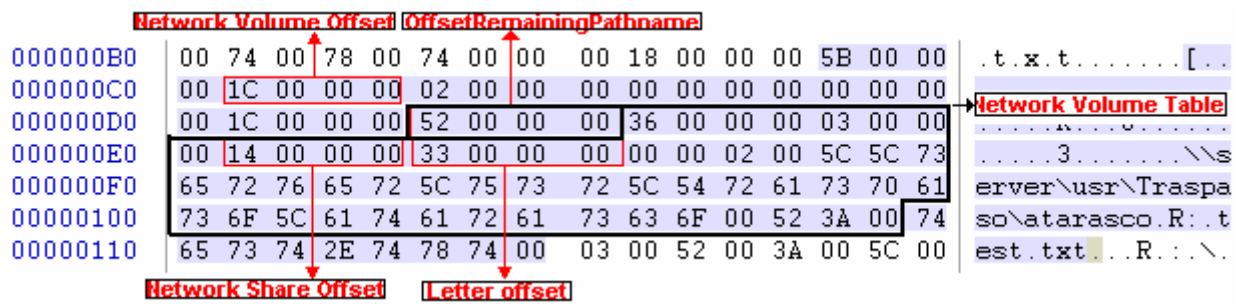
- 0 Desconocido
- 1 No existe el root directory
- 2 Dispositivo extraíble (USB, disco Zip, disquetera,...)
- 3 Dispositivo de disco (Disco duro)
- 4 Unidad de Red Remota
- 5 Unidad de DVD o CD-ROM
- 6 Unidad RAM DRIVE

El tercer DWORD de la estructura LocalVolume almacena el número de serie del disco duro de destino. Esta información puede ayudarnos a identificar ficheros accedidos a través de dispositivos USB o ficheros copiados de otros dispositivos de almacenamiento.

El último DWORD, habitualmente 0x10, indica el offset en el que nos encontraremos el path del fichero referenciado en este fichero lnk. Esta posición coincidirá con el offset almacenado en la variable OffsetBasePathname de la estructura Filelocation (en nuestro caso, 0x0000002d desde la cabecera). La longitud de este path estará determinada por OffsetRemainingPathname.

**SubEstructura NetworkVolume:**

La estructura **NetworkVolumeTable** está presente cuando el fichero referenciado está almacenado en un recurso compartido de la red y el flag Filelocation->flag contiene el valor 2.



Estructura Filelocation con NetworkVolumeTable

El primer DWORD, al igual que con LocalVolume, almacena el tamaño total de esta estructura (0x00000036). A este DWORD le sigue el tipo de red al que se ha accedido (por defecto es 0x03 para redes Microsoft)

La ruta al fichero, a través de la red, está almacenada al final de la estructura NetworkVolumeTable, en la posición referenciada por la variable NetworkShareOffset . En nuestro ejemplo vale 0x00000014 y apunta al string `\\server\usr\Traspaso\atarasco`

Si este fichero se encuentra en una unidad de red, entonces la variable NetworkLetterOffset tendrá un valor distinto de 0 y almacenará la posición en la que se encuentra la letra de unidad de red (por ejemplo z: ). En nuestro ejemplo este offset vale 0x00000033 y apunta a R:

Finalmente, el nombre del objeto accedido está en el offset referido por “NetworkVolumeOffset” de la estructura Filelocation (0x00000052 apunta a `test.txt`).

Resumiendo, con esta estructura obtendremos 3 strings, nombre del fichero, nombre de la ruta de red, y unidad de red en la que se ha mapeado la ruta.

### ASCII y Unicode Structs:

Dependiendo de los flags de la cabecera, es posible encontrarse con varias estructuras en el fichero LNK. Estas estructuras, cuyo tamaño máximo es de 259 caracteres (0x0103), deben ser interpretadas de forma diferente si el flan de UNICODE está activo o no.

Las estructuras ASCII o Unicode dependen de la presencia de flags en la cabecera del fichero LNK, Nos podemos encontrar de 0 a n estructuras cuyo significado es el siguiente:

- **DESCRIPTIONSTRING:** Descripción de la aplicación (Ejemplo: *Microsoft console*)
- **RELATIVEPATH:** Path relativo a la aplicación (Ejemplo: `..\..\windows\system32\cmd.exe`)
- **WORKINGDIRECTORY:** Directorio de ejecución de la aplicación (Ejemplo `c:\windows`)
- **COMMANDLINE:** Parámetros a enviar a la aplicación que se ejecute. (Ejemplo: `/c dir`)
- **CUSTOMICON:** Icono que se mostrará en el fichero LNK

A continuación se muestran dos estructuras ASCII presentes en un fichero LNK que pertenecen a las estructuras DescriptionString, que es la descripción que muestra el Explorer al seleccionar el fichero lnk, y el Working directorio, que es la ruta en la que se ejecutará la aplicación

00000000	4C 00 00 00 01 14 02 00 00 00 00 00 C0 00 00 00	L.....À...
00000010	00 00 00 46 0C 00 00 00 00 00 00 00 00 00 00	...F.....
00000020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00000030	00 00 00 00 00 00 00 00 00 00 00 00 01 00 00	.....
00000040	00 00 00 00 00 00 00 00 00 00 00 00 26 00 43 3A	.....&.C:
00000050	5C 57 49 4E 4E 54 5C 73 79 73 74 65 6D 33 32 5C	\WINNT\system32\
00000060	64 72 69 76 65 72 73 5C 65 74 63 5C 6E 65 74 77	drivers\etc\netw
00000070	6F 72 6B 73 26 00 43 3A 5C 57 49 4E 4E 54 5C 73	orks&.C:\WINNT\s
00000080	79 73 74 65 6D 33 32 5C 64 72 69 76 65 72 73 5C	ystem32\drivers\
00000090	65 74 63 5C 6E 65 74 77 6F 72 6B 7E	etc\networks

### ASCII Structs

Esta estructura comienza con un WORD (2 bytes) que especifica el tamaño del bloque de datos a continuación (0x0026) y a continuación un string en ASCII de esa longitud.

De la misma forma que sucede con las estructuras ASCII, en una estructura Unicode el primer DWORD indica el número de caracteres de la cadena salvo que, en este caso en concreto, el bloque de datos ocupará el doble debido a que esta cadena se almacena en wchar.

```

struct _UnicodeData {
WORD size;
wchar_t *datos; // size*2
};

```

El contenido de estas estructuras es muy similar al de las estructuras ASCII. El primer WORD indica la longitud de la cadena de texto que viene a continuación. Dado que esta cadena está en unicode, el tamaño del bloque de datos es size \* 2.

### Estructuras Mixtas:

---

Las últimas versiones de Windows almacenan, además de las estructuras ASCII y Unicode mencionadas en el apartado anterior, otra información identificada por los flags de la cabecera: **256, 512, 1024, 8192, 16384**. Para cada uno de estos flags existirá una estructura cuyo primer WORD indica el tamaño total de la estructura.

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000370	00	2E	00	69	00	63	00	6F	00	14	03	00	00	07	00	00	...i.c.o.....
00000380	A0	25	53	79	73	74	65	6D	52	6F	6F	74	25	5C	49	6E	%SystemRoot%\In
00000390	73	74	61	6C	6C	65	72	5C	7B	41	43	37	36	42	41	38	staller\{AC76BA8
000003A0	36	2D	31	30	33	33	2D	30	30	30	30	2D	42	41	37	45	6-1033-0000-BA7E
000003B0	2D	30	30	30	30	30	30	30	30	30	30	30	31	7D	5C	53	-000000000001}\S
000003C0	43	5F	41	63	72	6F	62	61	74	5F	53	74	61	6E	64	61	C_Acrobat_Standa
000003D0	72	64	5F	44	54	2E	69	63	6F	00	00	00	00	00	00	00	rd_DT.ico.....
000003E0	00	45	09	92	7C	4E	09	92	7C	3C	EB	8B	00	24	00	02	.E.' N.' <ë .\$.
000003F0	00	28	E9	8B	00	02	00	00	00	90	41	92	7C	00	80	FD	.(é ..... A' . ý
00000400	7F	05	10	91	7C	D0	E7	8B	00	00	00	00	00	A0	E8	8B	... Đç .....è
00000410	00	18	EE	91	7C	70	09	92	7C	C0	E4	98	7C	6F	3E	92	..i' p' Ää o>'
00000420	7C	62	3E	92	7C	08	02	00	00	FC	EB	8B	00	F0	EB	8B	b>' ...üë .ðë
00000430	00	5C	00	00	00	48	EC	8B	00	9B	39	5F	74	A6	90	5F	.\...Hi . 9_t  _
00000440	01	EC	44	7A	74	32	E9	8B	00	8C	EC	8B	00	2E	00	00	.iDzt2é .ii ....
00000450	00	02	00	00	00	5C	00	5E	00	4C	90	5F	01	00	00	00	.....\.^ L .....
00000460	00	08	00	00	00	4C	90	5F	01	00	00	00	00	5C	00	08	.....L .....\..
00000470	02	45	00	00	00	48	64	62	43	5C	00	00	00	00	00	00	.E...HdbC\.....
00000480	00	03	00	00	00	25	00	53	00	79	00	73	00	74	00	65	.....%.S.y.s.t.e
00000490	00	6D	00	52	00	6F	00	6F	00	74	00	25	00	5C	00	49	.m.R.o.o.t.%\.I
000004A0	00	6E	00	73	00	74	00	61	00	6C	00	6C	00	65	00	72	.n.s.t.a.l.l.e.r

Contenido de una estructura Mixta (flag 16384)

Esta estructura contiene varios strings en ASCII y UNICODE con información como iconos y rutas de la aplicación aunque, la interpretación interna de los datos, se desconoce.

Estas estructuras también contienen un array de estructuras SHELLITEM descritas en el apartado SHELLID

### Estructura GUID Information:

Existen dos estructuras, identificadas por los flags 2048 y 4096 cuya organización interna difiere totalmente de las estructuras analizadas hasta la fecha. Estas estructuras contienen información de los GUIDs del producto y de los componentes instalados en el sistema

000001A0	5C	00	61	00	63	00	63	00	69	00	63	00	6F	00	6E	00	\.a.c.c.i.c.o.n.
000001B0	73	00	2E	00	65	00	78	00	65	00	14	03	00	00	06	00	s...e.x.e.....
000001C0	00	A0	31	75	67	41	56	6E	2D	7D	66	28	5A	58	66	65	.lugAVn-}f(ZXfe
000001D0	41	52	36	2E	6A	69	41	43	43	45	53	53	46	69	6C	65	AR6.jiACCESSFile
000001E0	73	3E	6F	52	32	27	77	46	6A	67	5A	41	7B	4D	5A	62	s}oR2'wFjgZA{MZb
000001F0	45	71	7D	5F	36	5F	00	00	00	00	02	00	00	00	FC	E2	Eq}_6.....üä
00000200	90	00	4E	EA	90	00	88	E2	90	00	00	00	00	00	C4	E2	.Në . ä .....Ää
00000210	90	00	AC	E2	90	00	EC	E2	90	00	00	00	00	00	20	00	.~ä . ä .....
00000220	00	00	F0	2E	0A	00	FC	E2	90	00	00	00	00	00	A4	E2	..ð...üä .....Pä

Contenido de una estructura GUID

El primer DWORD de la estructura (0x00000314) Indica el tamaño total de los datos tras los cuales encontraremos la siguiente estructura a analizar.

El segundo DWORD, denominado MagicNumber, es un valor fijo que siempre vale 0xa0000006 y que es indicativo de este tipo de estructuras.

A continuación nos encontramos los datos realmente interesantes. Lo que a priori parece una cadena de texto sin sentido, **lugAVn-}f(ZXfeAR6.jiACCESSFiles>oR2'wFjgZA{MZbEq}\_6\_**, es una estructura de longitud variable cuyos primeros y últimos 20 bytes almacenan un GUID.

```

struct _guidInfo {
    DWORD size;
    DWORD magic;
    char guidText[20];
    char description[1];
};

```

Microsoft codifica estas estructuras en base85 por lo que la decodificación de los primeros 20 caracteres (1ugAVn-}f(ZXfeAR6.ji) devuelve una estructura GUID que podría representarse de la siguiente forma: **{90110c0a-6000-11d3-8CFE-0150048383C9}**

Tras este guid del producto (si es la estructura header & 1024) o del msi (estructura header & 2048) se encuentra una descripción opcional que siempre termina con el carácter '>'. En nuestro ejemplo la descripción es "ACCESSFiles". Tras esta cadena, se encuentra el GUID del componente (oR2'wFjgZA{MZbEq}\_6\_) que decodificado representa el GUID **{f2d782f8-6b14-4fa4-8FBA-565CDDDB9B2A8}**

El resto de los datos tiene un formato desconocido, en la que se vuelve a repetir estos guids almacenados en unicode. Para continuar con el procesado del resto de los datos haremos uso de la variable *size* de la estructura.

### Estructura Hostname:

---

La mayoría de los ficheros lnk contienen como última entrada del fichero una estructura que contiene información del equipo en el que se creó el fichero lnk. Su estructura es la siguiente:

```

struct _Hostname {
    DWORD size; //always 0x 00 00 00 60 ?
    DWORD flags; //always 0x A8 00 00 03
    DWORD OffsetToEnd; //always 0x 00 00 00 58
    DWORD zero; //always 0x00000000
    unsigned char hostname[0x10]; //padded with zeroes
    unsigned char unknown[0x40];
};

```

El tamaño de esta estructura es siempre 0x00000060 y contiene un string de 16 bytes, rellenado con 0's en el que se encuentra el nombre del equipo en el que se encuentra el fichero de destino.

00 65 00 72 00 63 00 61 00 70 00 4E 00 47 00 60	e r c a p N G
00 00 00 03 00 00 A0 58 00 00 00 00 00 00 00 72	X r
65 64 62 75 6C 6C 00 00 00 00 00 00 00 00 00 C2	edbull Å
8D 24 AF 94 0B 43 44 BD 8F B0 52 01 E1 28 F9 62	!\$   CD%   ° R á (ùb
EB 12 EB C6 29 DA 11 B0 18 A4 ED 3F EC 21 4D C2	ë ëÆ)Û ° ¢i?i!MÅ
8D 24 AF 94 0B 43 44 BD 8F B0 52 01 E1 28 F9 62	!\$   CD%   ° R á (ùb
EB 12 EB C6 29 DA 11 B0 18 A4 ED 3F EC 21 4D 10	ë ëÆ)Û ° ¢i?i!M
00 00 00 05 00 00 A0 25 00 00 00 A9 00 00 00 00	% ©

Nombre del equipo (redbull) en la estructura Hostname

Los últimos 40 bytes parece ser una sucesión de 2 estructuras de 32 bytes que contienen los mismos datos. La información almacenada en esta estructura es todavía desconocida.

### Ejemplo:

---

A continuación se muestra el resultado de la ejecución de la herramienta Lnk extractor sobre un fichero de disco. Prueba de concepto disponible en <http://www.514.es>

```
D:\Programación\forensics>lnk -o " Microsoft Office Word 2003.lnk"
(c) 2006 Andres Tarasco Acuña – atarasco @ gmail.com
[+] Url: http://www.rootkitdetector.com - http://www.514.es
[+] Block Device Microsoft Office Word 2003.lnk opened (2577 Bytes)

CTIME: 03/04/2006 22:41
MTIME: 13/11/2006 08:08
ATIME: 03/04/2006 22:41
File Length: 778240
Volume Type: Fixed (Hard disk)
Serial Number: 0459-0029
CommandLine:
RelativePath: ..\..\..\..\Archivos de programa\Microsoft Office\OFFICE11\WINWORD.EXE
LNKWORKINGDIRECTORY:
LNKDESCRIPTIONSTRING: Utilice Microsoft Office Word para crear y modificar texto y
gráficos en cartas, informes, páginas Web o mensajes de correo electrónico.
LNKCUSTOMICON: C:\WINDOWS\Installer\{90110C0A-6000-11D3-8CFE-
0150048383C9}\wordicon.exe
Struct 4096->GUID: {90110c0a-6000-11d3-8CFE-0150048383C9}
Struct 4096->string: WORDFiles
Struct 4096->GUID: {1ebde4bc-9a51-4630-B541-2561FA45CCC5}
LNKHOSTNAME: redbull
```

### Referencias:

---

- 1- Lnk Analyzer: <http://www.514.es>
- 2- Encase. Análisis forense: <http://www.encase.com>
- 3- Estructuras SHITEMID <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/shell/reference/structures/itemidlist.asp>