Título: Extracción de estructuras OLE (*Reversing structured storage serialized*)
Autor: Andrés Tarascó Acuña: atarasco@gmail.com

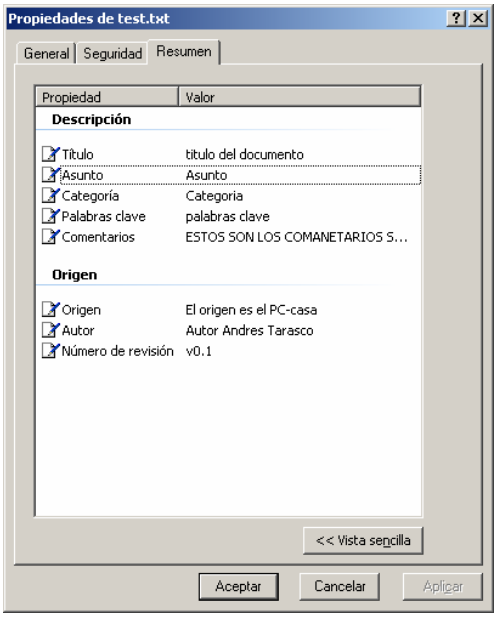
Introducción:

La mayoría de los productos de software actual manejan y guardan más información en los documentos de la que a priori es visible para los usuarios. Estos rastros digitales son los que nos pueden permitir, a la hora de realizar un análisis forense sobre un sistema, identificar de forma clara que ha sucedido a lo largo de un determinado período de tiempo.

En este documento nos centraremos en analizar como están almacenadas las estructuras OLE en el disco y como estas estructuras son utilizadas, por los productos de la familia Microsoft Office, para guardar información extra dentro de ficheros (.doc, .xls, .mdb, .dot, .ppt,..).

Esta información oculta puede contener, en algunas ocasiones, datos especialmente sensibles para la organización como puede ser direccionamiento ip, identificador del usuario que ha modificado un documento o rutas de equipos de la red. Será precisamente esta información, que atenta de forma expresa contra nuestra privacidad y de la que el usuario no tiene constancia, la que será utilizada en muchas ocasiones durante análisis forenses. Interpretarla correctamente puede suponer la diferencia entre realizar con éxito un análisis, por encontrar por ejemplo referencias a un usuario en la *metadata*, o descartar información vital como si se tratase de basura.

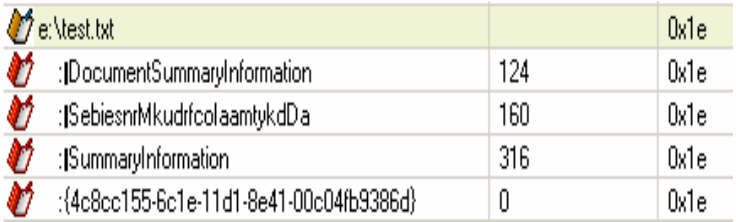
Las estructuras OLE también se pueden encontrar dentro de entradas ocultas de ficheros denominadas (*ADS*: *Alternate Data Streams*) y serán analizadas a lo largo del documento. Entre los ejemplos de ADS que nos podemos encontrar con estructuras OLE se encuentran:



Propiedades del fichero

ADSs con estructuras OLE:

- **Document SymmaryInformation**
- **SebiesnrMkudrfcoIaamtykdDa**
- **SummaryInformation**

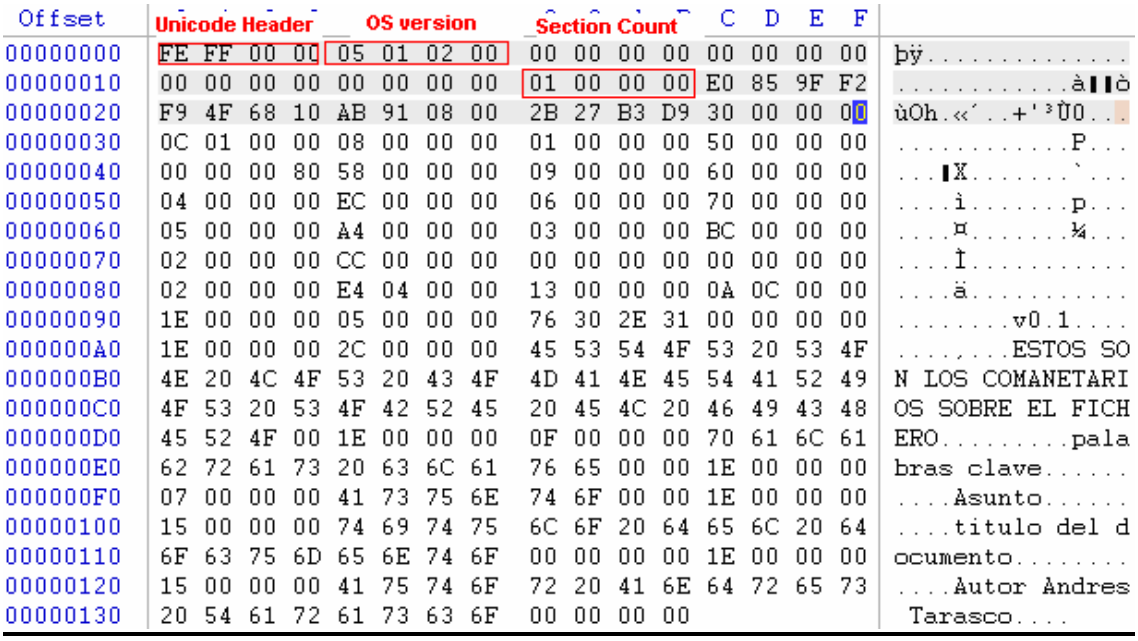


Identificación de ADS usando Rkdetector

Como ejemplo para desarrollar este análisis nos basaremos en las propiedades establecidas sobre nuestro fichero test.txt y que se encuentran almacenadas en varios ADS.

Análisis de la información:

A continuación se muestra el contenido del stream de datos almacenados en el fichero *\test.txt:♣SummaryInformation*



Vista del stream de datos ♣SummaryInformation

Los primeros 48 bytes del fichero conforman la cabecera del documento. Esta cabecera es imprescindible para identificar una estructura OLE.

Análisis de cabeceras:

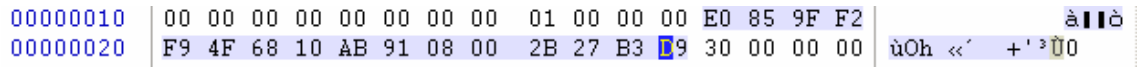
WORD Unicode: FEFF (siempre)
WORD Zero: (siempre 0)
8 Bytes que identifican la versión del sistema operativo y la plataforma en la que se generó el documento. (v5.1,0,0)
ClassID es un identificador de 16Bytes que suele valer 0 en esta cabecera.
WORD sectioncount**:** numero de secciones que se encontrarán a después de la cabecera.

```
typedef struct _header { //0x1C
    WORD Unicode;   //FEFF
    WORD Zero;    //0000
    char OSH;     /GetOs Version
    char OSL;    //GetOs Version
    char BuildVersion;
    char dwPlatformId;
    unsigned char ClassID[16];
    DWORD SectionCount;
} HEADER;
```

Análisis de lista de secciones:

La lista de secciones, que se muestra a continuación, se compone de un uid de 16bytes al que le sigue un segundo *DWORD* que especifica la posición en la que se encuentra la sección.

struct _uuid INTERFACENAME;
DWORD OffsetCount;



***uid de la sección* //{f29f85e0-4ff9-1068-ab91-802b27b3d9}**

Este uid es un identificador único para cada tipo de sección. El uid nos ayudará a saber como interpretar los datos que se encuentren en la sección.

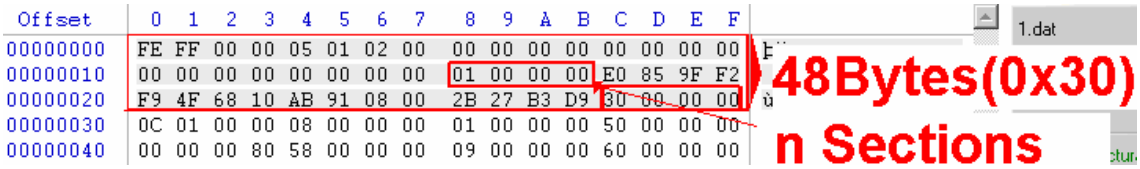
{0, "**SummaryInformation**", "\xE0\x85\x9F\xF2\xF9\x4F\x68\x10\xAB\x91\x08\x00\x2B\x27\xB3\xD9"},
{1,"**DocumentSummaryInformation**", "\x02\xd5\xcd\xd5\x9c\x2e\x1b\x10\x93\x97\x08\x00\x2b\x2c\xf9\xae"},
{2,"**DocumentSummaryInformationII**", "\x05\xD5\xCD\xD5\x9C\x2E\x1B\x10\x93\x97\x08\x00\x2B\x2C\xF9\xAE"},
{3,"**SebiesnrMkudrfcoIaamtykdDa**", "\x92\x04\x44\x64\x8b\x4c\xd1\x11\x8b\x70\x08\x00\x36\xb1\x1a\x03"}

Ejemplo de la estructura uid:

```
INTERFACENAME = { /*fc1a093a-a986-43d1-99f5-6b98f279bf61*/
    0xfc1a093a,     //DWORD
    0xa986,         //WORD
    0x43d1,         //WORD
    {0x99, 0xf5, 0x6b, 0x98, 0xf2, 0x79, 0xbf,0x61}//Bytes
  };
```

Serán precisamente estos uids los que utilizaremos para identificar cabeceras OLE validas cuando realicemos búsquedas en ficheros. Esta estructura (***InterfaceName/ OffsetCount***) se repetirá tantas veces como numero de secciones hayan sido definidas en la cabecera (en nuestro ejemplo el numero de secciones vale 0x00000001).

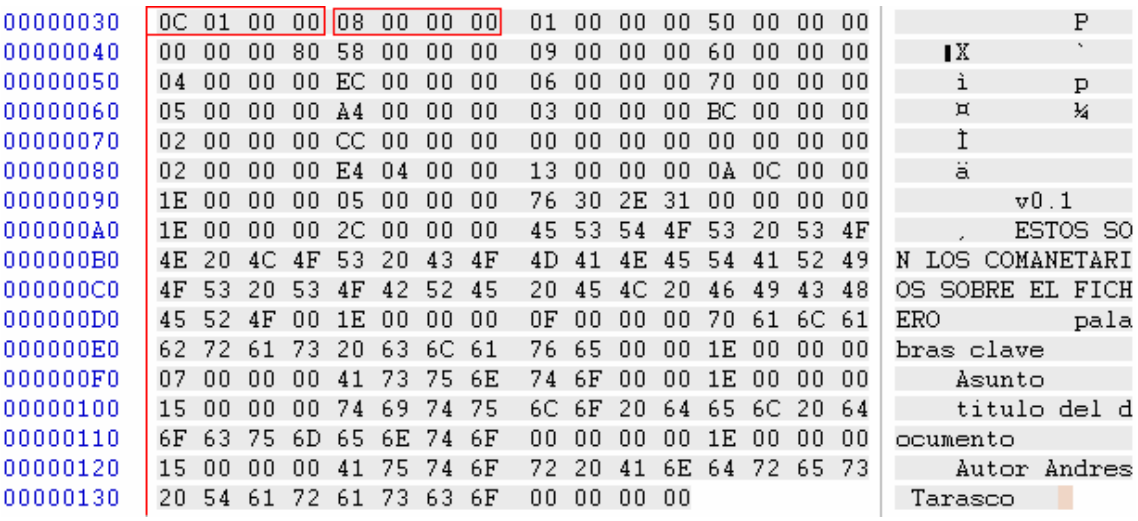
Como se puede ver en la siguiente imagen, el offset a la sección vale 0x00000030 y accederemos a esta dirección del fichero para examinar la primera sección.



Saltamos a la sección SummaryInformation en el offset 0x30

<u>Análisis de secciones:</u>

El primer *DWORD* de la nueva cabecera, 0x0000010c indica el tamaño total de la sección, y a continuación se identifica con otro *DWORD* el numero de propiedades que contiene la sección (0x08).

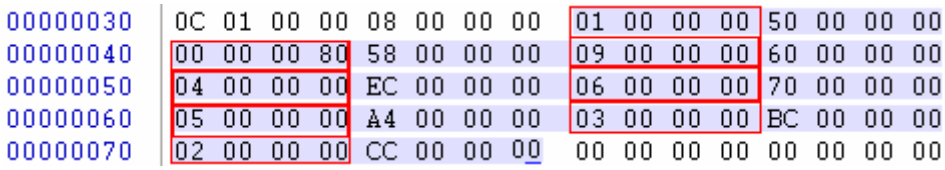


Datos de una sección de tamaño 0x00000x0c con 0x00000008 propiedades

Después de estos 8 bytes de la cabecera de la sección, encontramos la lista de propiedades. La lista de propiedades es de tamaño variable (el definido por el *DWORD npropiedades*) y su estructura, que consta de dos *DWORDS*, es descrita a continuación:

Struct _ListaPropiedades {
DWORD idPropiedad; **Tipo de propiedad.**
DWORD OffsetToENTRADA; **Posición relativa al inicio de la sección en la que se encuentran los datos.**
}

Dependiendo del UID asociado a la sección (que identifica si estamos analizando una sección del tipo *SummaryInformation*, *DocumentSummaryInformation*, ..) cada *id* significará algo diferente. Dado que cada desarrollador puede utilizar nomenclaturas diferentes, no existe una forma segura de identificar el significado de cada propiedad a priori.



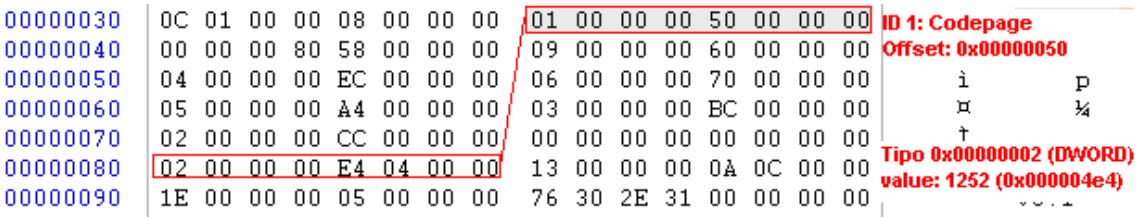
Identificadores de propiedad contenidos en esta sección

La siguiente tabla referencia algunas las propiedades (del 2 al 8) de dos secciones asignadas diferentes y su significado:

SUMMARYINFORMATION_TITLE 2	**DPCSUMMARYINFORMATION_CATEGORY 2**
SUMMARYINFORMATION_SUBJECT 3	**DPCSUMMARYINFORMATION_PTarget 3**
SUMMARYINFORMATION_AUTHOR 4	**DPCSUMMARYINFORMATION_Bytes 4**
SUMMARYINFORMATION_KEYWORDS 5	**DPCSUMMARYINFORMATION_Lines 5**
SUMMARYINFORMATION_COMMENTS 6	**DPCSUMMARYINFORMATION_Paragraphs 6**
SUMMARYINFORMATION_TEMPLATE 7	**DPCSUMMARYINFORMATION_Slides 7**
SUMMARYINFORMATION_LASTSAVEDBY 8	**DPCSUMMARYINFORMATION_Notes 8**

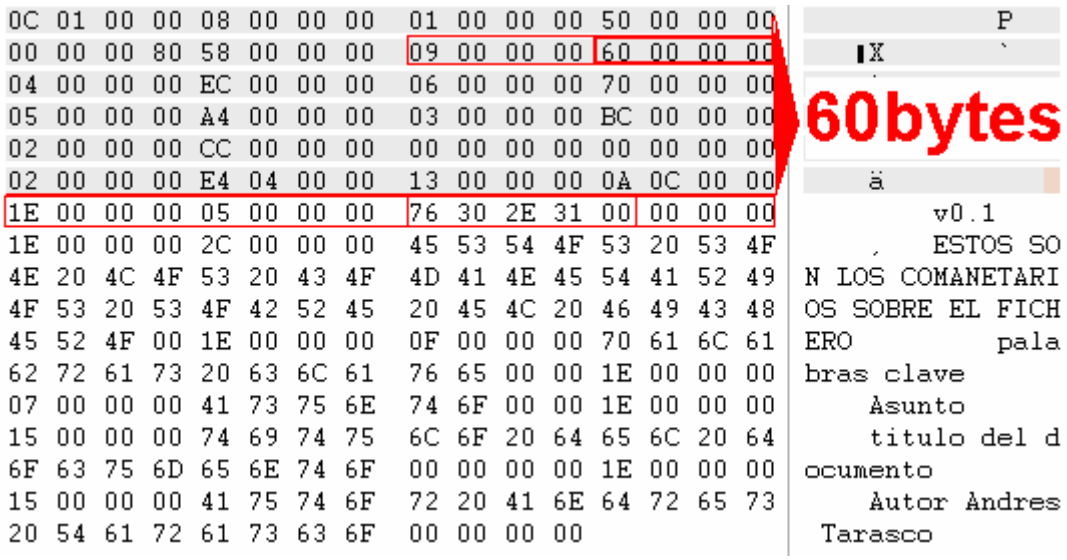
Existen además propiedades genéricas que tienen el mismo significado en todos los streams de datos. Estos son por ejemplo:

ID 0x00000001: CodePage (**DWORD**) ID 0x80000000: Locale (**DWORD**)



Análisis de un elemento de la lista de propiedades

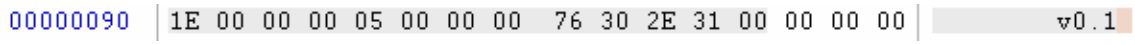
Hacemos uso del offset de una de las entradas en la tabla de propiedades (por ejemplo propiedad: 0x00000009 Offset 0x00000060) y accedemos al principio de la propiedad utilizando el valor del offset.



Contenido de la propiedad 0x09 situada en el offset 0x60

<u>Análisis de propiedades:</u>

Los datos que se encuentran en la posición 0x60 (relativa a la cabecera de la sección) son denominados por Microsoft tipos *"variants"*. Esto significa que la estructura y tamaño de cada propiedad variará dependiendo de este código. Cada tipo de datos "*variant*" consta de un *DWORD* que identifica como se interpretarán los datos asociados a él. En nuestro ejemplo:



Detalle de la propiedad 0x09

Tipo Propiedad: 0x0000001e (Cadena de texto ASCII)
DWORD size: 0x00000005 (longitud de la cadena 5 bytes)
Datos: 76 30 2e 31 00 (cadena: v0.1\0)

Existen multitud de tipos de propiedades diferentes. Entre los más utilizados, y la forma de interpretarlos, se encuentran:

TIPO: 0x03 (INT): Datos: Dword (valor)
TIPO: 0x05 (array de cadenas): Datos: DWORD (numero cadenas) DWORD (len) [..]..
TIPO: 0x1e (cadena): Datos: DWORD (len) […..](Cadena)
TIPO: 0x1f (Unic): Datos: DWORD (len ascii)[….] (cadena unicode de len*2)
TIPO:0x40 (fecha): Datos: FILETIME (fecha)
TIPO: 0x1c (array propiedades): Datos DWORD (N estructuras) DWORD (tipo de propiedad)[..] (datos)DWORD (tipo de propiedad)[…] (datos)….

El funcionamiento de estos tipos puede además verse afectada por la presencia de flags como por ejemplo 0x1000 que provoca que la propiedad sea interpretada como un array de las estructuras apuntadas por el identificador del tipo de propiedad.

Ejemplo del análisis del contenido del buffer mostrado anteriormente:

```
Operative System:       5.1 (Build: 2 Platform 0)
Section Numbers:        0x1
-------------------------------------------------
CurrentSection: 0x0
uuid:           {f29f85e0-4ff9-1068-ab91-802b27b3d9}
Type:           SummaryInformation
OffsetCount:    00000030
Size:           0x0000010c
blocks:         0x00000008

ID: 0x01 OFFSET: 0x0050 - Code Page          : 1252
ID: 0x80000000 OFFSET: 0x0058 - Locale       : 3082
ID: 0x09 OFFSET: 0x0060 - RevisionNumber     : v0.1  (5 Bytes)
ID: 0x04 OFFSET: 0x00ec - Author             : Autor Andres Tarasco  (21 Bytes)
ID: 0x06 OFFSET: 0x0070 - Comments           : ESTOS SON LOS COMANETARIOS SOBRE EL
FICHERO  (44 Bytes)
ID: 0x05 OFFSET: 0x00a4 - Keywords           : palabras clave  (15 Bytes)
ID: 0x03 OFFSET: 0x00bc - Subject            : Asunto  (7 Bytes)
ID: 0x02 OFFSET: 0x00cc - Title              : titulo del documento  (21 Bytes)
```

La prueba de concepto desarrollada junto con este *paper*, genera un resultado similar al mostrado y puede ser utilizado para la búsqueda de streams OLE en ficheros del disco y en documentos de office.

El paquete **reversing_ole.zip** esta disponible en http://www.514.es