UNIVERSIDAD INTERNACIONAL DE LA RIOJA

Universidad Internacional de La Rioja
Máster universitario en Seguridad Informática

Detección antiforense Open Source

Trabajo Fin de Máster

presentado por: Torres Álvarez, José Belarmino

Director/a: Botella Juan, Guillermo

Ciudad: Santa Cruz de Tenerife

Fecha: 21 de septiembre de 2016

Resumen

Las herramientas de investigación digital aportan resultados en forma de artefactos e históricos de actividad, que permiten la reconstrucción del incidente y la aportación de evidencias. La validez de estos resultados se apoya entre otros en el establecimiento de correlaciones analíticas entre los artefactos y el incidente, en el conocimiento de los sesgos introducidos por la herramienta, y en la investigación de posibles de prácticas antiforenses.

Este proyecto aporta un método de contraste sobre la credibilidad de líneas de tiempo (timelines) en el marco de investigación digital forense**,** en relación con el falseamiento de sellos de tiempo en archivos de dispositivos de almacenamiento con volúmenes NTFS.

Enfocado en el análisis post-mortem, se diseña partiendo del estudio de resultados de diferentes herramientas forenses Open Source, pretendiendo ofrecer un método alternativo de detección.

Palabras clave: Líneas de tiempo, NTFS, Detección, Timestomping, Postmortem

Abstract

Digital investigation tools produce results such as artifacts and timelines, allowing the incident reconstruction and evidence collection. The validity of these results is based on the establishment of analytic correlations between the incident and the artifacts, and the knowledge about the bias introduced by tools during the investigation of possible antiforensic practices.

This work offers a contrast method about the credibility of timelines in the forensic digital investigation field, in relation to manipulation of timestamps on files belonging to NTFS volumes.

Focused on the post mortem analysis, it is designed from the study of different Open Source forensic tools, pretending to offer an alternative detection method.

Keywords: Timelines, NTFS, detection, Timestomping, Postmortem

CONTENIDO

Índice de ilustraciones

Índice de tablas

1. Introducción

La investigación digital es un proceso donde se desarrollan y prueban hipótesis que dan respuesta a eventos digitales utilizando el método científico (Carrier, 2005) [1]. Sobre un incidente digital, se trataría de obtener evidencias y reconstruir los eventos que den soporte o refuten la hipótesis, ofreciendo garantías de integridad, trazabilidad y verificabilidad.

En el marco del análisis forense digital, *las referencias temporales a ocurrencias de eventos* (en adelante TMS), *se enmarcan en tres áreas principales* (Raghavan, 2014) [2]:

- *El estudio de diferentes fuentes*: Es el proceso de búsqueda de artefactos para establecer correlaciones y determinar sesgos.

- *El ordenamiento causal de eventos*: En la literatura se aprecian dos aproximaciones, el estudio de correlaciones en logs y el de *metadatos de artefactos*.

- El estudio de la semántica del TMS: Los TMS se interpretan en su contexto, por ejemplo, existen diferencias de interpretación entre los TMS de los cambios de zona y los del tiempo local (Schatz et al., 2006) [3].

Las metodologías para el análisis forense digital evolucionan, tal y como lo hacen otras ramas forenses científicas, hacia la comprensión total del incidente, considerando las aportaciones de las áreas anteriormente expuestas. Partiendo del planteamiento de metodologías en torno a secuencias cronológicas de eventos (Chen et al., 2003) [4], a luego apoyarse en estudios sobre artefactos y eventos relacionados (Gudjonsson, 2010) [5], y finalmente obtener inferencias sobre niveles superiores de abstracción (Hargreaves & Patterson, 2012) [6], con el empleo de modelos semánticos (Chabot et al., 2015) [7].

Una de las recomendaciones para la recopilación de evidencias digitales forenses, la RFC 3227 [8], considera que toda evidencia recopilada para que sea admisible desde el punto de vista judicial, debe ser auténtica, completa, fiable y creíble. De ahí *la importancia del empleo de metodologías en el campo del análisis digital forense,* y la *detección del uso de técnicas antiforenses* que intenten quebrar alguno de los citados principios.

2. Objetivos y contexto

Uno de los objetivos de las prácticas antiforenses es sembrar dudas sobre la credibilidad de la investigación forense, lo que en último término podría conllevar la invalidez de las pruebas en la defensa judicial del caso. **El objetivo general de este proyecto** consiste en mitigar este efecto, **mejorando la credibilidad atribuible a las secuencias temporales generadas** por diferentes herramientas forenses.

La secuencia temporal de eventos se puede representar a través de timelines utilizando *técnicas de reconstrucción* de eventos que aplican diferentes metodologías:

- Por correlación de eventos (Case et al., 2008) [9]
- Análisis sintáctico o semántico de eventos (Chabot et al., 2015) [7]
- Reconstrucción de eventos desde artefactos de bajo nivel (Turnbull & Randhawa, 2015) [10], o infierencia de eventos de alto nivel en base a los de bajo nivel (Hargreaves & Patterson, 2012) [6]
- *Reconstrucción de eventos por a extracción de TMS, pudiendo utilizar parsers* (Gudjonsson, 2010) [5] o aplicar heurísticas (Kälber, Dewald & Idler, 2014) [11]

A través del tiempo se ha ido perfilando un conjunto de características deseables para el diseño de metodologías sobre la reconstrucción de eventos:

- Empleo de *una ontología* (sistema de representación del conocimiento) adaptada a la metodología, que posibilite una reducción de tipologías de eventos que quedan fuera del dominio.
- *Contar con un proceso reproducible***,** lo suficiente, como para dar validez judicial a la investigación (algo que se ve favorecido si se emplea una ontología).
- Empleo de un *sistema de inferencia* que de soporte a escenarios significativos, evitando estudios excesivamente academicistas.
- *Uso de procesos eficientes*, que den respuesta a los problemas derivados de la existencia de grandes volumenes de datos. Utilizando técnicas asumibles bajo restricciones de tiempo, en cuanto a que el tiempo de proceso se traduce en un costo económico, que en muchos casos no puede asumirse, con lo que se pondría en entredicho la aplicación de la metodología.
- *Automatización del proceso*, facilitando su aplicabilidad a través de máquinas, liberando así al investigador de tareas tediosas o de larga duración.

En cuanto al empleo de *técnicas antiforenses para el falseamiento de TMS,* a través de la literatura podemos llegar a las siguientes conclusiones:

- *La posibilidad de realizarse utilizando técnicas simples*, como la alteración del tiempo del reloj del sistema o la *modificación de atributos MACE de ficheros* (Willassen, 2008) [12].
- *La posibilidad de detección a través del estudio de correlaciones:* El principio de intercambio de Lorcad [13], establece que “siempre que dos objetos entran en contacto transfieren parte del material que incorporan al otro objeto”, lo que da pie a utilizar como línea de investigación el estudio de correlaciones entre objetos forenses. En el caso particular de la manipulación de datos, puso de manifiesto que "las propiedades estadísticas del sistema varían tras una sobreescritura u ocultamiento".
- *La dificultad para su detección automática*: Zdzichowski [14] manifiesta sobre la automatización de la detección del falseamiento de TMS, que no existe una forma automatizada que revele facilmente este tipo de técnica antiforense.

En la definición de un *marco generalista* sobre el análisis post-mortem de dispositivos de almacenamiento, se deberá considerar como fuentes de evidencia, aquellos *artefactos siempre presentes en su sistema de archivos.*

La metodología propuesta **pretende mejorar la credibilidad de los timelines sobre el sistema de archivos NTFS**, partiendo del **estudio de correlaciones entre artefactos en imágenes post-mortem de dispositivos de almacenamiento.** Se implementa en una **herramienta Open Source** que de utilidad para **analistas forenses** de cara a contrastar el nivel de credibilidad de sus hallazgos.

Una vez vista la introducción, que definía el problema, el contexto, y los objetivos pretendidos, en el capítulo 3 se explican las características del sistema de archivos de referencia, y en el capítulo 4 se relacionan los artefactos que son de interés para el estudio. En el capítulo 5 se explican las posibles vías de manipulación, que serán empleadas en los casos de estudio definidos en el capítulo 6. El capítulo 7 define los componentes de la metodología, que serán empleados por la herramienta mostrada en el capítulo 8. El capítulo 9 analiza los resultados de la metodología en aplicación de la herramienta, enmarcada en los casos de uso definidos anteriormente. En los capítulos 10, 11 y 12, se resume la experiencia en base a sus garantías de aplicabilidad, conclusiones del trabajo y posibles mejoras.

3. NTFS

NTFS (New Technology File System) es el sistema de archivos transaccional que incluye Windows desde Windows NT, existiendo implementaciones en otros sistemas, como NTFS-3G en los kernels de MAC OSX y Linux. En este proyecto se estudia su versión 3.1 en el marco del sistema operativo Windows.

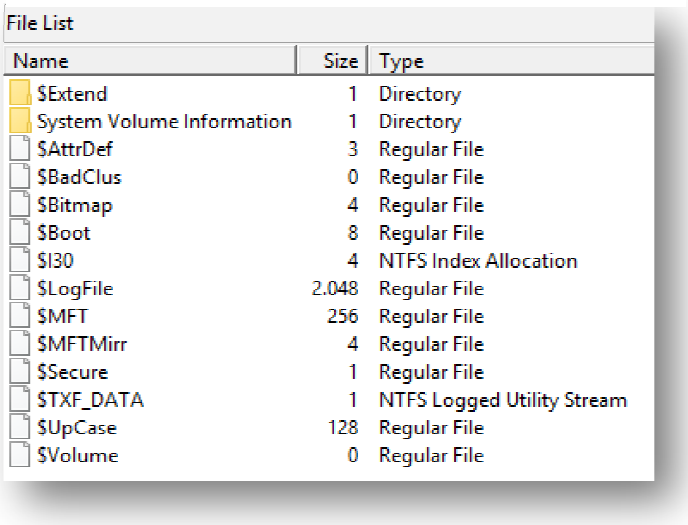
3.1. Sistema de archivos

Todos los componentes (o artefactos) de NTFS se implementan como archivos, siguiendo una estructura que admite retrocompatibilidad, ampliable a través de extensiones.

Tras el formateo NTFS de un volumen, se crea una estructura donde aparecen los archivos identificados como de metadatos del sistema (cuyos nombres empiezan por $), y que almacenan metadatos (datos de soporte) involucrados en la gestión del sistema de archivos.

Algunas herramientas forenses, como FTK Imager Lite, presentan también como artefactos a algunos extractos de determinados atributos (estructuras de metadatos) de especial interés forense, como $INDEX_ALLOCATION, conteniendo información sobre índices de archivos, denominado $I30, o el del atributo $LOGGED_UTILITY_STREAM, representado como $TXF_DATA, en referencia al stream de datos del log transaccional.

Ilustración 1. Artefactos NTFS creados en directorio raíz, via FTK imager



Los artefactos que dan soporte al sistema no fueron diseñados para que un usuario interactúe con ellos, y las estructuras de datos que no son siempre conocidas, siendo necesario contar con analizadores (parsers) que permitan su interpretación.

Aunque Microsoft publica muchas de estas estructuras, no todas se detallan, por lo que algunos de los parsers se diseñan a partir de trabajos de ingeniería inversa, perfeccionados a través de prueba y error [15]. Cuando se trabaja con ingeniería inversa, no se puede garantizar ni que los parsers extraigan toda la información realmente contenida en los artefactos, ni que contemplen toda la casuística, ya que se trata de un conocimiento inferido.

3.2. Timestamps (TMS)

En NTFS los sellos de tiempo (TMS) de los archivos se almacenan en formato UTC, y su resolución temporal es de 100 nanosegundos a contar desde el 1 de enero de 1601. Existen cuatro TMS asociados a operaciones sobre los archivos, que pueden expresarse formatos equivalentes: MACB, MACE o CAMR. Las referencias a estos formatos se emplean indistintamente en la literatura, por lo que conviene entender sus equivalencias.

Tabla 1. Equivalencias de formatos y condiciones de actualización de TMS [16]

MACE	MACB	CAMR	Actualización del TMS ($SI)
LastWriteTime (Modificación)			
M	**M**	**A**	Se actualiza al tiempo actual **cuando se crea el archivo** (IRP_MJ_Create), en una sobreescritura o en un renombrado. Cuando se **modifica el contenido del archivo**, se actualiza con el tiempo actual en el momento en que se cierra el handle del archivo. Se actualiza ante los requerimientos de IRP_MJ_FLUSH_BUFFERS y IRP_MJ_SET_INFORMATION (con determinadas clases)
LastAccessTime (Acceso)			
A	**A**	**R**	Se actualiza al tiempo actual **cuando se crea el archivo** (IRP_MJ_Create). Se actualiza al TMS actual **cuando un archivo se abre o es accedido de alguna manera**, como cuando se copia o mueve, cuando su handle está cerrado. La actualización de este dato no es inmediata, sino que se utiliza el dato almacenado en memoria, que más tarde se consolidará en la tabla contenedora de TMS de los archivos (MFT), **cuando la diferencia entre valores sea superior a 1 hora**. Se desactiva parcialmente su actualización cuando se activa el hive NtfsDisableLastAccessUpdate, en este caso sólo actualizaría cuando se crea un archivo, se mueve o se copia.
CreationTime (Creación)			
C	**B**	**C**	Se actualiza al tiempo actual **cuando se crea el archivo** (IRP_MJ_Create). Por defecto este TMS se tuneliza (transfiere) si un fichero se borra y se crea otro archivo con el mismo nombre en un intervalo inferior a 15 segundos. Se actualiza ante el requerimiento de IRP_MJ_SET_INFORMATION (con determinadas clases)
ChangeTime (Entrada modif. MFT)			
E	**C**	**M**	Se actualiza al tiempo actual **cuando se crea el archivo** (IRP_MJ_Create), y cuando se copia, pero no cuando se mueve de una ubicación a otra. Cuando se producen **cambios en la entrada MFT** del archivo, se utiliza el tiempo actual cuando se cierra su handle. Se actualiza ante los requerimientos de IRP_MJ_FLUSH_BUFFERS, IRP_MJ_SET_EA,IRP_MJ_SET_SECURITY, IRP_MJ_SET_INFORMATION (con determinadas clases)

A partir del trabajo de Carrier (Carrier, 2005) [1], se puede conocer el significado de cada una de las referencias temporales TMS, en relación con las operaciones sobre archivos. En el mundo del forense digital, existen investigaciones sobre la manera en que las operaciones más comunes realizadas archivos, actualizan el conjunto de TMS, en el marco de diferentes versiones del sistema operativo Windows [17].

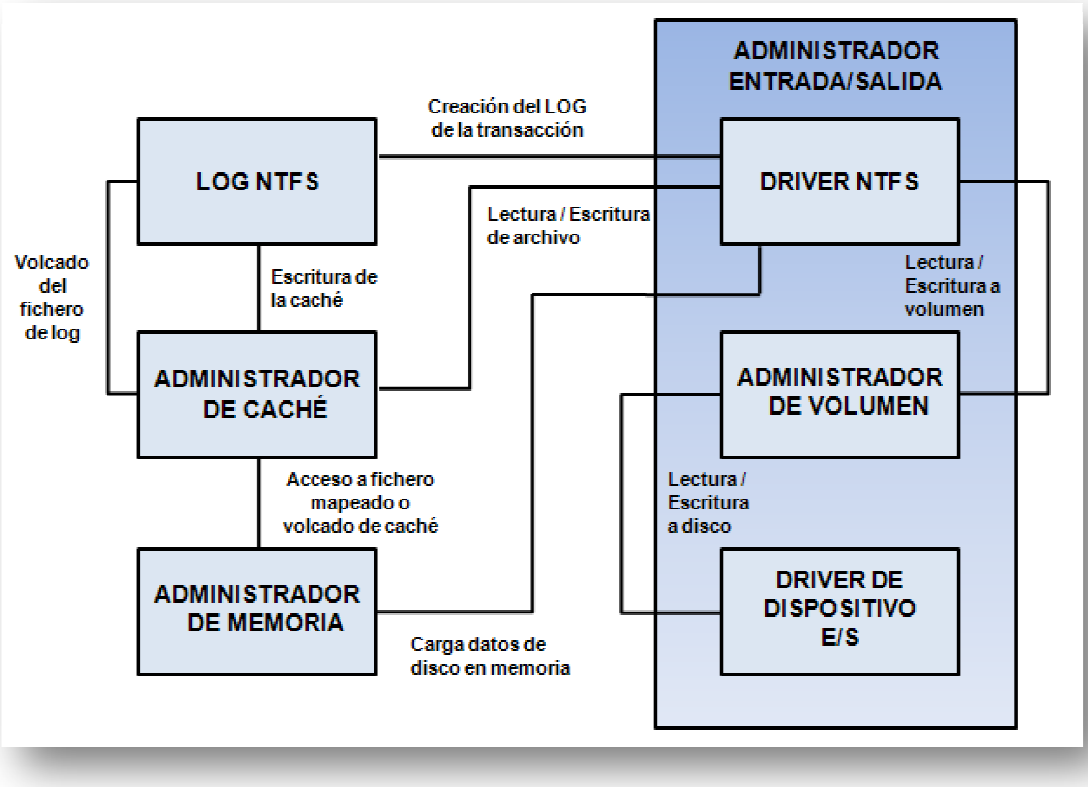
3.3. Administrador de Entrada/Salida

Para entender el proceso de cómo se produce la actualización de los TMS en los archivos, se debe entender cómo funciona la entrada/salida de datos en Windows.

El sistema de entrada/salida proporciona una capa de abstracción lógica y física sobre los dispositivos. Entre sus componentes ejecutivos nos encontramos con el administrador de entrada/salida, que incorpora entre otros mecanismos, la carga y descarga de drivers, la protección de recursos compartidos, y el acceso a dispositivos de forma transparente a través de funciones software.

En las actualizaciones de datos sobre el sistema de archivos participan: el administrador de entrada/salida, el driver NTFS, el administrador de caché y el de memoria. El driver NTFS no actualiza directamente al dispositivo, sino que se comunica con el servicio de log de ficheros cuando inicia una transacción, y con el administrador de caché cuando realiza lecturas o escritura a un archivo.

Ilustración 2. Arquitectura entrada/salida Windows en relación con NFTS [19]



El administrador de entrada/salida cuando recibe un requerimiento de entrada/salida, crea un paquete de requerimiento de entrada/salida (en adelante IRP) en memoria representando la operación. La comunicación de operaciones entre los componentes del sistema de entrada/salida se realiza con IRPs.

El administrador de memoria se encarga de gestionar tanto del mapeado entre la memoria física y virtual, como del paginado del contenido de memoria a disco, realizando escrituras al dispositivo a través de su escritor de páginas modificadas y mapeadas.

El administrador de caché se ejecuta en Windows de forma centralizada, dando soporte a actualizaciones de archivos y de streams de datos. La actualización no es inmediata, sino que implementa mecanismos como el *write-back cache* con *lazy write*, por la que la escritura efectiva de datos a disco desde la caché (flush), *se produce tiempo después de que el administrador de memoria realizase la modificación sobre los datos.*

Cuando lazy write *realiza el volcado a disco, el administrador de caché notifica al driver del sistema de archivos para que actualice su vista* [20] sobre la longitud válida del archivo (de la que se lleva cuenta en memoria).

El proceso de escritura de metadatos de archivos a disco es el siguiente:

- El driver escribe un registro de log el tipo de operación que va a realizar, y luego llama al administrador de caché para que vuelque a disco el registro de log, para asegurarse de que existe registro previo de la operación en disco
- El driver escribe los datos que desea actualizar, y el indicador del registro del log (LSN) en el administrador de caché, y la actualización a disco la realiza dicho administrador, en operaciones de volcado (flush).

En la escritura de streams de datos, éstos se protegen a través de transacciones para asegurar que se guarda el dato del registro de log antes de que se complete la escritura del stream.

Con la utilización de un driver de sistema de archivos (modo Kernel) o con el sistema de archivos desmontado y utilizando el handle de un dispositivo (accesible a través de la estructura de datos *file object* del Kernel), sería posible realizar escrituras en discos y volúmenes utilizando el driver del dispositivo.

De cara al resto del proyecto, denominaremos *escritura lógica* a la realizada a través del driver del sistema de archivos, y *escritura directa* a la que puentea a dicho driver, saltándose sus restricciones.

4. Fuentes de evidencias

Se considera un subconjunto de artefactos NTFS como fuentes de evidencia de las acciones realizadas sobre los archivos.

4.1 Master File Table

La master file table (en adelante *MFT*) es un fichero que se implementa como un conjunto de registros de metadatos que posibilita el acceso a los archivos y carpetas de un volumen. NTFS reserva el 12.5 porciento del tamaño del volumen para la MFT, en lo que se conoce como Zona MFT.

Los registros MFT representan a los archivos y directorios del volumen. Cada registro es típicamente de 1KB, según se especique en el sector de arranque, y apunta al cluster de comienzo del contenido del archivo o directorio que referencia.

Un fichero se identifica en la MFT a través de su número del número de la entrada de su registro en la tabla. Los números de entrada del 0 a la 23, son fijos, referenciando a determinados ficheros de metadatos del sistema, de tal manera que la posición 0 por ejemplo, siempre se asigna al registro del archivo $MFT, que apuntaría a su vez a la propia tabla MFT.

Cuando se crea un archivo, se genera o asigna una entrada en la MFT, aunque es posible que se asigne más de una, en cuyo caso la primera entrada se denomina *base file record,* y se utiliza como entrada de referencia al archivo en la MFT.

Cuando se elimina, se conservan los datos, sobreescribiendo la entrada marcándola como inactiva o libre, aunque con el tiempo la entrada se reutiliza, momento en el que se incrementa su número de secuencia.

Las entradas MFT utilizan valores *fixup* que se emplean en la versión en disco de la estructura de datos, reemplazando los dos últimos bytes de cada sector con este valor. Cada registro MFT comienza con una cabecera (Header) de 42 bytes:

Tabla 2. Cabecera de un registro MFT [1]

Offset Entrada (bytes)	Descripción
0x00	Firma "FILE"
0x04	Offset al array fixup
0x06	Tamaño del array fixup
0x08	Número de secuencia del $LogFile (LSN)
0x10	Número de secuencia
0x12	Cuenta de links
0x14	Offset al primer atributo
0x16	Flags (en uso y directorio)
0x18	Tamaño usado de la entrada MFT
0x1C	Tamaño asignado a la entrada MFT
0x20	File Reference (incluye el offset al registro base)
0x28	Id del siguiente atributo

El número de entrada para el archivo en la MFT, se calcula a partir del denominado *file record number* de 64 bits, que se almacena en el campo *File Reference* del registro base, siguiendo la estructura _MFT_SEGMENT_REFERENCE.

Ilustración 3. Estructura MFT_SEGMENT_REFERENCE [21]

```
typedef struct _MFT_SEGMENT_REFERENCE {
  ULONG  SegmentNumberLowPart;
  USHORT SegmentNumberHighPart;
  USHORT SequenceNumber;
} MFT_SEGMENT_REFERENCE, *PMFT_SEGMENT_REFERENCE;
```

La dirección se obtiene combinando HighPart con LowPart (*File number* de 48 bits), y con el número de secuencia (*Sequence number* de 16 bits) [20]. A través del número de secuencia se posibilita el chequeo de la integridad NTFS. El número de entrada se obtiene dividiendo la dirección de referencia entre el tamaño de la entrada (típicamente 1024 bytes).

Tras la cabecera, y hasta completar el tamaño asignado a las entrada (1024 bytes por defecto), se completa con registros denominados *atributos*, que son estructuras de tamaño variable que contienen metadatos de los archivos del volumen. Todos los atributos utilizan la mismo tipo de estructura en su encabezado, en la que se distingue si se trata de un atributo *residente*, que guarda su contenido en la propia MFT, o *no residente*, que guarda su contenido en disco, y que en la MFT incluye las referencias para su localización de su contenido.

En el encabezado del atributo se encuentra información que revela la naturaleza del atributo.

El campo *FormCode.* indica si el atributo es residente o no, debiendo decodificarse a través de la estructura *Resident* o la *NonResident*.

Ilustración 4. Encabezado de un atributo

```
typedef struct _ATTRIBUTE_RECORD_HEADER {
  ATTRIBUTE_TYPE_CODE TypeCode;
  ULONG               RecordLength;
  UCHAR               FormCode;
  UCHAR               NameLength;
  USHORT              NameOffset;
  USHORT              Flags;
  USHORT              Instance;
  union {
    struct {
      ULONG  ValueLength;
      USHORT ValueOffset;
      UCHAR  Reserved[2];
    } Resident;
    struct {
      VCN      LowestVcn;
      VCN      HighestVcn;
      USHORT   MappingPairsOffset;
      UCHAR    Reserved[6];
      LONGLONG AllocatedLength;
      LONGLONG FileSize;
      LONGLONG ValidDataLength;
      LONGLONG TotalAllocated;
    } Nonresident;
  } Form;
} ATTRIBUTE_RECORD_HEADER, *PATTRIBUTE_RECORD_HEADER;
```

El cluster físico del archivo donde encontrar el contenido del atributo No residente, se puede obtener a través del mapeado del número de cluster virtual dentro del archivo (VCN) a número lógico de clúster del volumen (LCN) indicado en el registro, que se gestiona de forma comprimida, y el número de clusters. Si la información no cabe en el segmento base, puede almacenarse en otro fichero externo, a cuyo cluster podrá accederse a través del atributo $ATTRIBUTE_LIST. Los atributos No Residentes, guardan su contenido en disco en lo que se conoce como *runs*.

Tabla 3. Atributos MFT y condiciones de residencia [20]

Atributos	Residente
$VOLUME_INFORMATION	Siempre
$VOLUME_NAME	Siempre
$STANDARD_INFORMATION	Siempre
$INDEX_ROOT	Siempre
$INDEX_ALLOCATION	Nunca
$EA_INFORMATION	Siempre
Resto de atributos	En ocasiones

Los *atributos más importantes desde el punto de vista forense en relación con los TMS de archivos son*:

- $STANDARD_INFORMATION (tipo 0x10, en adelante $SI): Sostiene los TMS de los archivos, y son los que muestra el explorador de archivos de Windows. Tal y como se indica en [A3], los TMS de $SI se actualizan siguiendo patrones fijos en respuesta a operaciones sobre archivos, lo que permite inferir a través del estudio de cambios en los TMS de un archivo, la operación realizada sobre el mismo.

Ilustración 5. Estructura del atributo $SI [1]

Offset Entrada (bytes)	Descripción
0x00	C - TMS de creación de archivo
0x08	M - TMS de modificación del archivo
0x10	E - TMS Alteración MFT
0x18	A – TMS de acceso al archivo
0x20	Flags
0x24	Máximo número de versiones
0x28	Número de versión
0x2C	Class ID
0x30	Owner ID
0x34	Security ID
0x38	Quota Charged
0x40	USN

- $FILE_NAME (tipo 0x30, en adelante $FN): Es un atributo residente de longitud dinámica, que contiene el nombre del archivo, referencia a su directorio padre y un conjunto de TMS que no se actualizan con tanta frecuencia como los de $SI [1].

  Al igual que ocurre con los TMS de $SI, los TMS de $FN se actualizan siguiendo patrones fijos en respuesta a operaciones sobre archivos, relacionadas con la creación, movimiento o renombrado del archivo, que en algunos casos es coincidente con el contenido de TMSs del $SI [A4].

  A través del contenido del atributo $FN, es posible reconstruir la jerarquía directorios del sistema de archivos utilizando las referencias a los directorios padre. *Esta característica puede ser empleada para la construcción de timelines.*

Ilustración 6. Estructura del atributo $FN [1]

Offset Entrada (bytes)	Descripción
-	Encabezado estándar de atributo
0x00	Referencia de archivo al directorio padre
0x08	C - TMS de creación de archivo
0x10	M - TMS de modificación del archivo
0x18	E - TMS Alteración MFT
0x20	A – TMS de acceso al archivo
0x28	Tamaño asignado del archivo
0x30	Tamaño real del archivo
0x38	Flags (directorio, comprimido, etc.)
0x3c	Valor de reparse
0x40	Longitud del nombre de archivo
0x41	Espacio de nombres del archivo (Posix, Win32, DOS, Win32 & DOS)
0x42	Nombre del archivo

- $INDEX_ROOT (tipo 0x90) y $INDEX_ALLOCATION (tipo 0xA0): Ambos se identifican internamente bajo el nombre ($I30), y referencian a los índices en las entradas de los directorios. Sus TMS se actualizan por copia desde $SI.

La MFT se emplea típicamente para la detección simple de falseamientos de TMS a través de la comparación de los TMS de sus $SI y $FN [23], pudiéndose aplicar algunas de estas heurísticas:

- No debe ocurrir que todos los TMS de $SI sean anteriores a los TMS de $FN [A4], puesto que en todas las operaciones con archivos, cuando se modifica algún TMS de $FN siempre existe algún TMS de $SI que se ve afectado.
- La entrada de la MFT y su número de secuencia indicarán si el archivo se encuentra fuera del rango que debería en relación con el resto de entradas.
- El TMS de cambio de un archivo en condiciones normales debería ser más reciente que el de creación de su directorio padre.

4.2 Indexado NTFS

NTFS mantiene bufferes de índices sobre sus archivos y directorios conformando un árbol B+ con índices sobre los nombres de los archivos y directorios. Su primer nivel se implementa en el atributo $INDEX_ROOT, y sus hijos en el atributo $INDEX_ALLOCATION.

El atributo $INDEX_ROOT es de tamaño dinámico, pudiendo contener un pequeño número de entradas de índice.

El atributo $INDEX_ALLOCATION es de tamaño fijo múltiplo de 4096 bytes, y sus registros se almacenan de forma contigua (comenzando por el registro cabecera), de tal manera que si no se cubre el tamaño completo del nodo, existirá un área vacía al final de los registros "slack space", que se extenderá hasta completar su tamaño.

Una de las utilidades forenses sobre los bufferes de índices es la identificación de nombres de ficheros borrados y sus TMS. Cuando se elimina un archivo, el driver no sobreescribe estos datos, sino que aumenta el "slack space" del nodo, modificando las direcciones lógicas se sus registros a través de los nombres de archivos.

$INDEX_ALLOCATION está compuesto por un conjuto de registros índice. Cada uno de ellos conteniendo una cabecera, un encabezado de nodo y un conjunto de entradas de índice común para archivos y directorios, en la que se almacena a través de una estructura de atributo del tipo $FN los TMS del fichero referenciado .

Tabla 4. Entrada de índice en $INDEX_ALLOCATION [1]

Offset Entrada (bytes)	Descripción
0x00	MFT Reference para el nombre de fichero (en caso de entrada de directorio)
0x08	Tamaño de esta entrada
0x0A	Tamaño del atributo $FN
0x0C	Flags (0x01 Existe nodo hijo, 0x02 Última entrada en la lista)
0x10	**Atributo $FN (si la longitud es >0)**
Últimos 8 bytes	VCN del nodo hijo en $INDEX_ALLOCATION (si se indica en el flag)

La estructura del atributo $FN está ampliada [24]:

- Nombre del archivo (es clave de ordenación de nodos hijo)
- Tamaños lógico y físico del archivo

- TMS de modificación, acceso, cambio y creación
- Se incluye al final una estructura de registro del atributo $FN

Desde el punto de vista forense se pueden utilizar las siguientes heurísticas:

- Los TMS registrados *los copia de los TMS incluidos en el $SI* de la MFT, y son actualizados con mayor frecuencia y bajo diversas circunstancias [45].
- A través del análisis de los TMS de cambio de MFT es posible entender como fueron movidos los archivos.

Gracias a esta última funcionalidad *es posible obtener evidencias de la práctica antiforense para conseguir una actualización automática del $FN a partir del movimiento de un archivo en local,* aprovechándose del efecto tunneling que implementa el sistema de archivos NTFS.

4.3 Journal de cambios de archivos

NTFS ubica bajo la carpeta $Extend de su sistema de archivos, un conjunto de artefactos que dan soporte a funcionalidades del sistema incorporadas en forma de extensiones.

El artefacto que implementa el Journal de cambios producidos en los archivos de NTFS es \$Extend*$UsnJrnl*. Se introduce a partir de Windows Vista con el objetivo de agilizar los procesos de búsqueda sobre la información de archivos, y es utilizado tanto por el Sistema de Replicación de Archivos de Windows, como por el servicio Windows Search.

Se trata de un *sparse file* (fichero disperso que no almacena explícitamente los valores no inicializados) que referencia a través de su número de secuencia de actualización (USN), un log persistente con todos los cambios realizados sobre los ficheros dentro del volumen.

Se trata de un log circular que crece en disco con el tiempo hasta un tamaño máximo predefinido, y que cuenta con mecanismos FIFO para la desasignación de bloques (Russon & Fledel, 2004) [25]. Con el paso del tiempo se pierde información al reciclarse las entradas.

Windows no siempre lo crea inicialmente en todos los dispositivos, pero puede crearse y configurarse con la utilidad fsutil, especificando tanto su tamaño total (parámetro m) y su tamaño de asignación de bloque (parámetro a).

Ilustración 7. Creación manual del un Journal de cambios

```
C:\Windows\system32>fsutil usn createjournal m=1000 a=100 f:

C:\Windows\system32>fsutil usn queryjournal f:
Id. de diario de USN    : 0x01d2006ffa1a4f5c
Primer USN              : 0x0000000000000000
Siguiente USN           : 0x0000000000000000
USN mínimo válido       : 0x0000000000000000
USN máximo              : 0x7ffffffffff0000
Tamaño máximo           : 0x0000000000100000
Diferencia de asignación: 0x0000000000040000
Versión del registro mínima admitida: 2
Versión del registro máxima admitida: 4
Seguimiento de intervalos de escritura: deshabilitado
```

Los cambios realizados sobre los archivos se almacenan en su stream de datos alterno **$J**, implementado como fichero disperso.

Cuando el fichero $UsnJrnl sobrepasa el tamaño total especificado (almacenado en el stream de datos alterno $Max), se comienzan a rellenar con ceros los datos que preceden a la ventana de cambio hasta alcanzar un tamaño igual al máximo del journal. El tamaño del journal se reduce cuando alcanza el doble del máximo configurado.

Desde el punto de vista forense sirve para confirmar los eventos NTFS de (creación, borrado, modificación, renombrado y movimiento de un fichero y directorio) en un período específico, siendo posible encontrar trazas de un fichero que haya sido eliminado.

En las últimas versiones, se puede habilitar el “seguimiento de intervalos de escritura” (range tracking), por el que también puede realizarse un seguimiento de las partes modificadas de los archivos.

El funcionamiento del Journal de cambios es dependiente de la versión del registro USN con el que fue creado, estableciendo una compatibilidad de versiones mínima y máxima [A2]. En la estructura de este registro se almacenan los cambios producidos en los archivos.

Ilustración 8. Estructura de registro de USNJrnl (V4) [26]

```
typedef struct {
  USN_RECORD_COMMON_HEADER Header;
  FILE_ID_128              FileReferenceNumber;
  FILE_ID_128              ParentFileReferenceNumber;
  USN                      Usn;
  DWORD                    Reason;
  DWORD                    SourceInfo;
  DWORD                    RemainingExtents;
  WORD                     NumberOfExtents;
  WORD                     ExtentSize;
  USN_RECORD_EXTENT        Extents[1];
} USN_RECORD_V4, *PUSN_RECORD_V4;
```

- *Header:* refencia una estructura que describe los tamaños y versiones admitidas por el registro (record_size, major_version y minor_version). Se incluyen referencias al archivo y al padre del archivo, así como el *USN* (Update Sequence Number), que identifica unívocamente a las instancias del registro.

- Reason: referencia a la razón del cambio en el archivo desde que el fichero o directorio fue abierto. Un extracto de las razones que se entienden más interesantes desde el punto de vista forense, se exponen a continuación.

Tabla 5. Reason de cambio en atributos TMS de archivos [18]

REASON	Descripción
USN_REASON_BASIC_INFO_CHANGE 0x00008000	Un usuario ha cambiado uno o más atributos de un archivo o directorio (por ejemplo, atributo de sólo lectura, oculto, sistema, archivo, o atributo de disperión), o uno o más de sus TMS

- *SourceInfo*: referencia a información sobre la fuente del cambio, utilizando banderas.

- *RemainingExtents:* referencia al número de registros del tipo USN_RECORD que suceden al actual.

- *NumberOfExtents:* referencia al número de extensiones del tipo USN_RECORD_EXTENT se encuentran incluidas en el registro actual, de un tamaño *ExtentSize* en bytes, y almacenadas en el array *Extents* en el registro actual.

4.4 Journal NTFS

Se trata de un Log circular que registra acciones y cambios en los metadatos de un volumen de almacenamiento NTFS, manteniendo una estructura transaccional a partir de la que poder restaurar la consistencia del sistema en caso de fallo. Se implementa a través del artefacto *$LogFile*.

Según el investigador forense David Cowen [27], el *$Logfile "es un gran artefacto para obtener un alto nivel de granularidad sobre exactamente qué cambios han ocurrido a un sistema de ficheros*".

Guarda pista de todos los cambios en el sistema de archivos, registrando en referencia a los archivos la siguiente información:

- Archivos creados y de sus registros MFT completos
- Archivos renombrados, incluyendo el nombre de archivo antiguo y nuevo
- El TMS de algunos registros

Las entradas de este artefacto son "registros *LSN*", que conforman listas enlazadas representando transacciones sobre el volumen, y referenciando operaciones de rehacer (redo) o deshacer (undo).

El tamaño del Log se puede configurar a través de chkdsk, con un tamaño mínimo para dar soporte transaccional de 2KB:

chkdsk {unidad} /L:{tamaño en bytes}

```
C:\Windows\system32>chkdsk f: /L
El tipo del sistema de archivos es NTFS.
El tamaño actual del archivo de registro es de 2048 KB.
El tamaño del archivo de registro predeterminado para este volumen es de 2048 KB.
```

Ha de tenerse en cuenta que al ser un log circular, el conocimiento histórico se verá limitado debido a la sobreescritura de evidencias menos recientes.

Desde el punto de vista forense, los atributos interesantes son:

- Redo_Operation: Operación de rehacer
- Undo_Operation: Operación de deshacer
- Filename: Para correlaciones por nombre de archivo
- CAMR: Sellos de tiempo (TMS)

4.5 NTFS transaccional

El NTFS transaccional (*TxF*), es una funcionalidad extendida de NTFS soportada por un administrador de recursos. Se trata de un conjunto de APIs con las que Windows implementa un sistema de transacciones en volumenes NTFS.

Con estas transacciones se consigue aislamiento de los archivos frente a cambios no deseados. El nivel de aislamiento puede definirse para un archivo, para un conjunto de archivos o entre diferentes fuentes.

Las referencia a TxF se realiza a través del atributo $LOGGING_UTILITY_STREAM de la MFT, y su contenido se relaciona con $LogFile y $UsnJrnl:$J., lo que sugiere *la posibilidad de utilizar este artefacto con fines forenses.*

4.6 System Volume Information

Es un directorio cuya función principal es albergar los puntos de restauración del sistema, la base de datos de indexado de archivos que utiliza Index Server para realizar búsquedas rápidas, o la información requerida por las Shadow Copy.

Las Shadow Copy son artefactos de alto valor forense, ya que se pueden utilizar posible contrastar el sistema de archivos frente a una línea base anterior, a partir de la información de archivos y carpetas que existían en el momento de su actualización.

5 Modos de escritura

Los escenarios del proyecto plantean en base al tipo de acceso utilizado para realizar acciones de manipulación de TMS sobre los archivos: escrituras lógicas y escrituras directas.

Estos tipos de accesos sugieren diferentes métodos de análisis para la interpretación del contenido de los artefactos forenses:

- En los casos de *escritura lógica* se modifican los TMS del $SI o a lo sumo los de $FN del MFT, dejando rastros evidentes en otros artefactos, por lo que *es adecuado emplear una metodología basada en contraste de contenidos de los artefactos*.

- En los casos de *escritura directa* se pueden realizar cualquier tipo de cambio, debiendo realizar operaciones de maquillaje para guardar la integridad del sistema. En este caso se considera más adecuada *una orientación hacia la detección de incoherencias, empleando artefactos* relacionados.

5.1 Escritura lógica

Se trata de escrituras sobre un sistema de archivos montado, sujetas a la aplicación de los mecanismos de gestión que implementa dicho sistema, utilizando las librerías dispuestas por el sistema operativo. Para realizar manipulaciones sobre metadatos deberá contarse con los privilegios suficientes.

Para la ejecución de este tipo de herramientas antiforenses, se deben poseer permisos tanto para manipular los arefactos NTFS, como una posterior *eliminación de indicios* que pudieran haber sido registrados en artefactos del sistema.

A la hora de estudiar las manipulaciones sobre los TMS a través de escrituras lógicas han de tenerse en cuenta las siguientes aseveraciones:

- Es posible realizar modificaciones sobre los TMS del $SI de la MFT utilizando API (librerías) del sistema
- La escritura del TMS de último acceso de MFT sólo se actualiza automáticamente cuando está habilitada la clave de registro Windows *NtfsDisableLastAccessUpdate*, que se encuentra deshabilitada por defecto en las últimas versiones de Windows para mejorar la respuesta del sistema.
- Existen herramientas antiforenses que con escrituras lógicas permiten modificar de los 4 TMS del $SI (clasicamente Timestomp).
- Un cambio en el reloj del sistema puede inducir referencias erróneas en el histórico de archivos
- Diferentes artefactos que recopilan información sobre las modificaciones realizadas en el sistema de archivos
- Los propios valores de los TMS son indicios de las operaciones que se pudieran haber realizado sobre los archivos que los poseen [41]

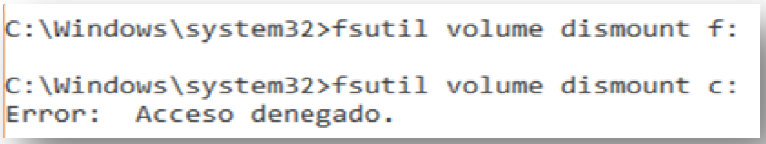
Las escrituras lógicas utilizarían la API proporcionada por el sistema Windows NtSetInformationFile, que *proporciona funciones para la modificación de los $SI de la MFT, pero no sobre los $FN,* por lo que si se pretende realizar una manipulación de TMS, ha de recurrirse a alguna alternativa.

5.2 Escritura directa

Microsoft ha introducido cambios en la pila de los drivers de almacenamiento, a partir de Windows Vista en adelante, para impedir muchas de las escrituras directas a discos y volumenes [29], conscientes de su empleo en técnicas maliciosas. En cualquier caso, estas restricciones no se aplican cuando existe un desmontaje previo del volumen.

El desmontaje del volumen donde se instala el sistema operativo, tampoco se permite cuando se invoca desde el espacio de usuario (desde Windows Vista en adelante), por lo que para este caso, si no se cuenta con un driver certificado que realice dicha ejecución en modo Kernel, la opción más factible es accediendo desde un sistema operativo portable.

Ilustración 9. Desmontaje de volumen de sistema en Windows



Las acciones antiforenses a través de escrituras directas, se ejemplifican con la herramienta antiforense *SetMace*, que realiza escrituras directas utilizando un acceso en modo Kernel a través de un driver, o en su defecto, a través del desmontaje del volumen.

Estas actuaciones son difíciles de detectar a través de procedimientos simples de verificación, puesto que vulneran los sistemas de control que establece el sistema. A la vez que son complejas de implementar, ya que es necesario emplear *técnicas para conservar la integridad y dar coherencia a los cambios* entre distintos artefactos, respetando el comportamiento nativo del sistema.

Según Joakim Schicht [30], autor de SetMace: “This new method of timestamp manipulation is a whole lot harder to detect. In oder to do so I can think of analyzing other artifacts like a shadow copy, $UsnJrnl and maybe $LogFile on not so heavily used volumes”.

6 Caso de estudio

En la construcción de timelines de archivos, se consideran las lecturas de los *TMS incluidos en los registros de la MFT,* y para detectar su falseamiento, el procedimiento más habitual consiste en contrastar de los TMS de $SI con los de $FN [23].

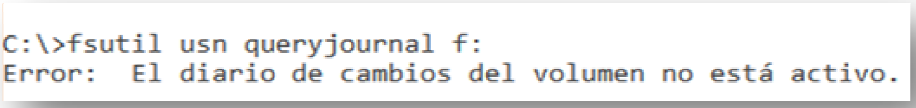
6.1 Alcance

Windows no ofrece una función en sus APIs para establecer direcamente los TMS de $FN, pero existen prácticas antiforenses, como la del doble movimiento del archivo en local o las escrituras directas en disco, con las que es posible realizar esta manipulación. Por lo que para su detección, será necesario contar con otros tipos de artefactos con los que establecer correlaciones.

El artefacto *$UsnJrnl*, implementa un journaling de cambios sobre archivos, aportando información valiosa desde el punto de vista forense, en relación con las operaciones realizadas sobre los archivos desde el momento en que se abren hasta que se cierran, pero la activación del journaling de cambios USN, como la de otros sistemas NTFS es opcional.

A partir de Windows Vista USN se activa por defecto sobre la partición del sistema, pudiendo gestionarse a través de políticas, pero es posible que no se cree automáticamente tras formatear un volumen [31], sino ante la demanda de determinados servicios.

Ilustración 10. Consulta al Journal de cambios tras formateo de dispositivo externo



Se comprueba tras el formateo ntfs de varios dispositivos a través de Windows, que el USN Journal no se activa automáticamente.

La desactivación del USN journal una vez activado, puede afectar a servicios del sistema que necesitan de esta característica, como el File Replication Service o el Indexing Service, lo que impacta negativamente en los tiempos de respuesta del equipo. Pero también puede emplearse como práctica antiforense, para perder evidencia de operaciones realizadas sobre archivos.

Al no ser infrecuente encontrarnos con dispositivos que no tengan activado el Journal de cambios, el proyecto considera utilizar también otros artefactos que puedan suplir su ausencia en el estudio de los casos de falseamiento TMS.

6.2 Evidencias en parsers Open Source

El estudio de evidencias a través de parsers Open Source presenta ventajas frente a los privativos al disponer de acceso no restrictivo a su código fuente, permitiendo realizar un análisis de funcionalidades en mayor profundidad, pudiendo justificar en caso de necesitarlo, el ámbito y alcance de sus resultados.

El empleo de diferentes parsers sobre un mismo artefacto se justifica por un lado en el conocimiento inferido que emplean, en la ampliación de información que pueden incluir procedente de otros artefactos, y en la incorporación de diferentes funcionalidades en cuanto a precisión de la información extraída, formatos de salida, capacidad de procesamiento, etc.

En cuanto al empleo de diferentes versiones de los parsers, las nuevas versiones siempre aportan correcciones, mejoras en funcionalidades y mayores compatibilidades con cambios en los sistemas, por lo que resulta conveniente disponer de las últimas para el estudio comparativo entre diferentes parsers.

A continuación se analiza las evidencias de la manipulación de TMS de la MFT, considerando algunos parsers Open Source consolidados en la comunidad forense.

6.2.1 Fls/mactime

La herramienta *fls* 4.2.0 se produce un listado de archivos y carpetas del directorio especificado en base a la información de la MFT de la imagen de disco proporcionada, que se puede utilizar como generador de timelines (con la opción –m). Presenta en un formato nativo no legible (body file), que debe formatearse con la herramienta *mactime* para obtener una versión entendible por humanos (ambas utilidades forman parte del toolkit SleuthKit.4.2.0 para Windows 32bits).

```
# fls –r –m /  imagen.dd > imagen.body

# mactime –b imagen.body –d > timeline.csv
```

El timeline producido en principio es simple. Muestra los TMS en formato MACB, el tamaño del registro, los permisos aplicados que indicaría si se trata de un archivo o directorio, el UID (id de usuario), GID (id de grupo), un grupo de tres datos separados por guiones que hacen

referencia a la ubicación del dato en la MFT, la ruta completa del fichero, y si el TMS obtenido se extrae de $SI o de $FN.

Ilustración 11. Timeline producido por fls-mactime

Date	Size	Type	Mode	UID	GID	Meta	File Name
Wed Sep 07 2016 19:39:32	9	macb	r/rrwxrwxrwx	0	0	39-128-1	/logico.txt
Wed Sep 07 2016 19:39:32	86	macb	r/rrwxrwxrwx	0	0	39-48-2	/logico.txt ($FILE_NAME)

Con la información aportada se puede iniciar una investigación forense con garantías. En la ilustración, el metadato 39-48-2, hace referencia a la entrada número 39 de la MFT, y al su atributo 48 ($FILE_NAME), identificado como el número 2 de los presentes en ese registro MFT, tal y como se puede comprobar con la herramienta *istat.*

Ilustración 12. Contenido de la entrada 39 MFT

```
C:\Users\yose\Documents\ntfs-TMS-eval\tsk>istat -f ntfs ..\imglogicobf.dd 39
MFT Entry Header Values:
Entry: 39        Sequence: 1
$LogFile Sequence Number: 1059679
Allocated File
Links: 1

$STANDARD_INFORMATION Attribute Values:
Flags: Archive
Owner ID: 0
Security ID: 264  ()
Last User Journal Update Sequence Number: 160
Created:        2016-09-07 19:39:32.393516700 (Hora de verano romance)
File Modified:  2016-09-07 19:39:32.393516700 (Hora de verano romance)
MFT Modified:   2016-09-07 19:39:32.393516700 (Hora de verano romance)
Accessed:       2016-09-07 19:39:32.393516700 (Hora de verano romance)

$FILE_NAME Attribute Values:
Flags: Archive
Name: logico.txt
Parent MFT Entry: 5     Sequence: 5
Allocated Size: 0       Actual Size: 0
Created:        2016-09-07 19:39:32.393516700 (Hora de verano romance)
File Modified:  2016-09-07 19:39:32.393516700 (Hora de verano romance)
MFT Modified:   2016-09-07 19:39:32.393516700 (Hora de verano romance)
Accessed:       2016-09-07 19:39:32.393516700 (Hora de verano romance)

Attributes:
Type: $STANDARD_INFORMATION (16-0)   Name: N/A   Resident   size: 72
Type: $FILE_NAME (48-2)   Name: N/A   Resident   size: 86
Type: $DATA (128-1)   Name: N/A   Resident   size: 9
```

La información Meta 39-128-1, se comprueba que se obtiene se captura de $SI (atributo 16), y a través de la referencia al atributo 128 ($Data), podríamos acceder al contenido residente del archivo.

Se analiza el comportamiento de este parser en la detección antiforense en escritura directa a disco. Para ello se emplea la herramienta Setmace, para manipular los 4 TMS del archivo

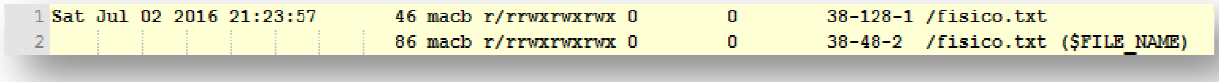
fisico.txt, de $SI y $FN. Posteriormente se extrae la imagen de disco para su procesamiento por FLS.

```
# fls –r –m /  imagen-af.dd > imagen-af.body

# mactime.pl –b imagen-af.body > timeline-af.csv
```

El resultado es un timeline normal en el que los TMS de $SI y $FN son iguales

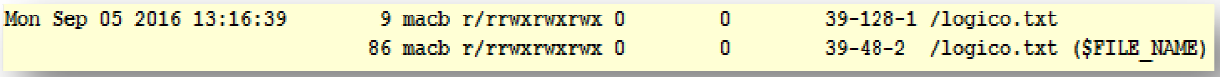
Ilustración 13. Timeline fls/mactime inicial antes de escritura directa



Tras ejecutar la herramienta sobre los 4 TMS de $SI y $FN

```
# Setmace64.exe f:\fisico.txt -z “2016:06:10:13:15:16:789:1234” -x
```

Como resultado, el fichero fisico.txt, aparece con sus TMS modificados, *no apareciendo otros archivos con TMS posteriores a la fecha de creación original del archivo*.



Se estudia de forma similar el comportamiento del parser ante una manipulación de TMS a través de escritura lógica utilizando Timestomp.

```
# echo “hola” > f:\logico.txt

# timestomp f:\logico.txt -z “Monday 7/25/2016 01:23:45 AM”
```

Comprobamos los TMS del fichero creado con fls antes y después de la ejecución de ejecuta la herramienta antiforense *Timestomp*, y luego se crea el timeline

```
# fls –r –m /  imagen-af.dd > imagen-af.body

# mactime.pl –b imagen-af.body > timeline-imagen-af.csv
```

Analizando el timeline resultante, se comprueba como la herramienta antiforense modifica los TMS de $SI pero no los de $FN, con lo que la detección antiforense a través de la ejecución de timestomp es detectable a través del contraste $SI con $FN.

```
Mon Jul 25 2016 01:23:45        9 macb r/rrwxrwxrwx 0        0        39-128-1 /logico.txt
Mon Sep 05 2016 13:16:39       86 macb r/rrwxrwxrwx 0        0        39-48-2  /logico.txt ($FILE NAME)
```

Puesto que los TMS de $SI se actualizan con mayor frecuencia, a través de este parser *no se podría concluir una manipulación por escritura lógica de TMS cuando los TMS de $SI manipulados fueran posteriores a los de $FN.*

Si se fuerza la actualización de los TMS de $FN con los de $SI, también sería posible evitar la aparición de TMS de $FN anteriores a los $SI. En principio se desarrolló la técnica de realizar un movimiento en local del archivo dentro del volumen, pero presupone la existencia de subdirectorios para los que se cuenta con permisos de escritura, pero William Ballenthin, descubrió que también es posible forzar la copia de TMS a $FN simplemente con renombrar el archivo, con lo que se evita la dificultad anterior.

Llevando a cabo esta última idea:

```
# format f: /FS:NTFS /V:TunnelTimestomp
# echo "hola" > f:\logico.txt
# ren f:\logico.txt logico.txt2
# timestomp f:\logico.txt2 -z "Monday 7/25/2016 01:23:45 AM"
# ren f:\logico.txt2 logico.txt
# timestomp f:\logico.txt -z "Monday 7/25/2016 01:23:45 AM"
```

Se comprueba que se actualizan también los TMS de $FN, con lo que ya no se es capaz de detectar este tipo de manipulación a partir del timeline generado por fls/mactime, apareciendo con los TMS propios de un fichero recién creado.

```
Mon Jul 25 2016 01:23:45        9 macb r/rrwxrwxrwx 0        0        38-128-1 /logico.txt
                               86 macb r/rrwxrwxrwx 0        0        38-48-4  /logico.txt ($FILE_NAME)
```

Sería posible realizar manipulaciones de TMS utilizando el efecto tunneling de NTFS [32], por el que se suplantan los TMS de un archivo por los de otro, siempre que el sistema no se haya configurado para evitarlo.

Este parser puede emplearse para la detección antiforense de falseamiento TMS en el caso en que la herramienta antiforense no llegue a modificar el $FN, y se comprobase que los TMS de $FN fueran más recientes que sus respectivos $SI.

En escrituras directas este parser está expuesto a la habilidad de la herramienta antiforense para enmascarar los TMS de $SI y $FN de la MFT. En las escrituras lógicas, puede ser eficaz en algunos casos, pero debido a que en su timeline muestra hasta una precisión de segundos, resulta insuficiente por si misma para detectar el caso de escritura lógica con copia automática $FN apoyado en Timestomp, ya que no puede aprovechar el defecto de Timestomp de no permitir establecer el contenido de los TMS a partir de los milisegundos en adelante ("DayofWeek Month\Day\Year HH:MM:SS [AM|PM]").

6.2.3 Supertimeline

La segunda herramienta es Plaso para Windows versión 1.4.0, que genera un supertimeline a partir de correlaciones entre parsers que implementa internamente. Para el estudio, se ejecuta sobre la imagen de la partición antes de realizar la manipulación.

```
# log2timeline buffer.plaso imagen-org.dd
# psort –z GMT –w supertimeline-org.csv buffer.plaso
```

Al no incluirse en la imagen artefactos del sistema distintos a los de NTFS que puedan emplearse como fuente de información, se comprueba como Plaso hace uso sólo de sus parsers *filestat.py* "File system stat object parser", y de "UsnJrnl" para recrear el timeline del sistema de archivos de la imagen.

Filestat.py es un script python que accede a los sistemas de archivos de forma virtualizada (vfs). En este caso se instancia al tipo NFS, y luego se recopila el estado de los objetos del sistema de archivos [A1]. La captura de las entradas del directorio NTFS se realiza con ntfs_file_entry.py, dando preferencia a la información procedente de la MFT [A2]. UsnJrnl parsea directamente del contenido del artefacto $UsnJrnl.

Si ejecutamos una escritura lógica con Timestomp:

```
# timestomp f:\logico.txt -z "Sunday 7/24/2016 23:23:45 PM"
```

Se comprueba que el parser usnjrnl recoge un rastro de actualización sobre el archivo manipulado bajo la razón *USN_REASON_BASIC_INFO_CHANGE*.

Ilustración 14. Supertimeline sobre manipulación lógica

	parser	TMS	CAMR	fuente	message
1	filestat	2016-07-24T23:23:45+00:00	atime;crtime;ctime;mtime	TSK:/logico.txt	TSK:/logico.txt
2	usnjrnl	2016-09-05T11:18:27.076377+00:00	Metadata Modification Time	TSK:/$Extend/$UsnJrnl:$J	logico.txt File reference: 39-1 Parent file reference: 5-5 Update reason: USN_REASON_BASIC_INFO_CHANGE USN_REASON_CLOSE
3	usnjrnl	2016-09-05T11:18:27.076377+00:00	Metadata Modification Time	TSK:/$Extend/$UsnJrnl:$J	logico.txt File reference: 39-1 Parent file reference: 5-5 Update reason: USN_REASON_BASIC_INFO_CHANGE

Se procede a comprobar al parser a través de escritura directa:

```
# Setmace64.exe f:\fisico.txt -z “2016:07:02:12:34:56:789:1234” -x
```

Se obtiene como resultado:

Ilustración 15. Supertimline sobre manipulación directa

	parser	TMS	CAMR	fuente	message
1	filestat	2016-07-02T12:36:14.912340+00:00	atime;crtime;ctime;mtime	TSK:/fisico.txt	TSK:/fisico.txt

Comparando ambos resultados se puede comprobar que no existen cambios apreciables, salvo la lógica modificación del TMS del fichero manipulado, que presenta pequeñas diferencias en el valor mostrado, pero la herramienta no deja rastro de su actividad en UsnJrnl.

En el caso de escrituras lógicas, a través de su parser *UsnJrnl se puede detectar la existencia de cambios en el fichero manipulado ante el indicio de la aparición de la razón USN_REASON_BASIC_INFO_CHANGE*, pero son difíciles de relacionar con una práctica antiforense. A través de escrituras directas, como la realizada por Setmace, no se detectan cambios en UsnJrnl, por lo que *este tipo de manipulaciones directas pasarían en principio desapercibidas.*

6.2.4 Mft2Csv

A diferencia de FLS o el Supertimeline de Plaso, el parser Mft2Csv para Windows v.2.0.0.36, realiza un volcado detallado de los registros MFT involucrados, incluyendo los TMS, lo que da pie a establecer nuevas vías de investigación a través de correlaciones con otros artefactos. Este parser extrae muchos de los datos incluidos en la MFT [A5], dando la posibilidad de establecer correlaciones con otros artefactos a bajo nivel.

Se realiza un análisis con el parser sobre una escritura lógica del archivo logico.txt.

Ilustración 16. Extracto del parser Mft2Csv

```
RecordOffset|RecordNumber|NumSequence|FN_FileName|SI_USN|HEADER_LSN|SI_CTime
0x00009C00  |39          |1          |logico.txt |160   |1060036   |05/09/2016 11:16:39.2173282
                                                                    |SI_ATime
                                                                    |05/09/2016 11:16:39.2318949
                                                                    |SI_MTime
                                                                    |05/09/2016 11:16:39.2318949
                                                                    |SI_RTime
                                                                    |05/09/2016 11:16:39.2173282
```

Se ejecuta Timestomp sin truco de movimiento:

```
# timestomp f:\logico.txt -z "Sunday 7/24/2016 23:23:45 PM"
```

Y se obtiene que aparte de cambiar su $SI por efecto de la manipulación, también cambia la referencia del registro a $USNJrnl y que su referencia al $LogFile.

Ilustración 17. Parseado con Mft2CSV tras manipulación lógica

```
RecordOffset|RecordNumber|NumSequence|FN_FileName|SI_USN|HEADER_LSN|SI_CTime
0x00009C00  |39          |1          |logico.txt |320   |1061424   |24/07/2016 23:23:45.0000000
                                                                    |SI_ATime
                                                                    |24/07/2016 23:23:45.0000000
                                                                    |SI_MTime
                                                                    |24/07/2016 23:23:45.0000000
                                                                    |SI_RTime
                                                                    |24/07/2016 23:23:45.0000000
```

Como se vio anteriormente con Supertimeline, la actuación de Timestomp genera trazas en $USNJrnl, pero ahora se comprueba también en $LogFile.

En resumen:

Este parser nos da la posibilidad de acceder de forma directa a las tablas de Journal a través de sus campos *SI_USN y HEADER_LSN*.

6.2.5 IndxParse

El parser IndxParse es un script en python que realiza un análisis de los atributos índice de la MFT, extrayendo los nombres de archivos, los TMS y los tamaños de archivo, permitiendo extraer información de los "slack space" de dichos atributos.

La herramienta IndxParse admite como entrada atributos índice del tipo $I30 [33], que se pueden extraer al igual que el resto de atributos utilizando la herramienta *icat* de Sleuthkit.

En el estudio, se emplea la entrada de la MFT para el directorio raíz (5), sobre su atributo $INDEX_ALLOCATION (160).

```
# icat ..\imglogicozbf.dd 5-160 > Index_Allocation.bin
```

Ejecutando IndxParse sobre el Index_Allocation antes de manipular el fichero se obtiene:

	FILENAME	PHYSICAL SIZE	LOGICAL SIZE	MODIFIED TIME	ACCESSED TIME	CHANGED TIME	CREATED TIME
1	logico.txt	16	9	2016-09-07 17:39:32.393517	2016-09-07 17:39:32.393517	2016-09-07 17:39:32.393517	2016-09-07 17:39:32.393517

Realizando una escritura lógica con Timestomp,

```
# timestomp f:\logico.txt -z "Sunday 7/24/2016 23:23:45 PM"
```

Se obtiene un nuevo resultado del parser tras extraer previamente el $Index_Allocation.

	FILENAME	PHYSICAL SIZE	LOGICAL SIZE	MODIFIED TIME	ACCESSED TIME	CHANGED TIME	CREATED TIME
1	logico.txt	16	9	2016-07-24 23:23:45	2016-07-24 23:23:45	2016-07-24 23:23:45	2016-07-24 23:23:45

El efecto de Timestomp, aparte de modificar el $SI de la MFT, también se traslada de forma transparente al atributo $INDEX_ALLOCATION.

Para el caso de escrituras directas, según Joakim Schicht, esta actualización de atributos índice se puede producir como efecto colateral de una llamada a la función NtQueryInformationFile [30]. Esto es algo que implementa en su herramienta como una mejora a partir de la versión v1.0.0.13.

Al existir un automatismo por parte del driver para preservar la integridad de TMS con los atributos índice y considerando que cuando se elimina o recicla un registro de la MFT, no se eliminan sus información de índice, pudiéndose acceder a los TMS anteriores a través de su "slack space", se *puede considerar esta información como fuente alternativa de validación para los TMS de $SI*. Esta idea ha sido sugerida por el autor de este parser cuando trata con archivos eliminados.

6.2.6 Usn2CSV

Este parser implementa el análisis del artefacto $UsnJrnl.

Tras crear un volumen NTFS, se ha de asegurar la activación del Journal de cambios.

Ilustración 18. Activación del Journal de Cambios

```
C:\Windows\system32>fsutil usn createjournal m=1000 a=100 f:

C:\Windows\system32>fsutil usn queryjournal f:
Id. de diario de USN    : 0x01d20170aedda9cd
Primer USN              : 0x0000000000000000
Siguiente USN           : 0x0000000000000000
USN mínimo válido       : 0x0000000000000000
USN máximo              : 0x7ffffffffffff0000
Tamaño máximo           : 0x0000000000100000
Diferencia de asignación: 0x0000000000040000
Versión del registro mínima admitida: 2
Versión del registro máxima admitida: 4
Seguimiento de intervalos de escritura: deshabilitado
```

El contenido inicial de $UsnJrnl, revela las operaciones involucradas en el proceso de creación del archivo logico.txt (echo "hola" > logico.txt):

	Offset	FileName	USN	Timestamp	Reason	MFTReference	MFTReferenceSeqNo
1	0x00000000	logico.txt	0	05/09/2016 11:16:39.2173282	FILE_CREATE	39	1
2	0x00000050	logico.txt	80	05/09/2016 11:16:39.2318949	DATA_EXTEND+FILE_CREATE	39	1
3	0x000000A0	logico.txt	160	05/09/2016 11:16:39.2318949	CLOSE+DATA_EXTEND+FILE_CREATE	39	1

Se procede a realizar al análisis del caso de escritura lógica con Timestomp:

```
# timestomp f:\logico.txt -z "Sunday 7/24/2016 23:23:45 PM"
```

Dando como resultado:

	Offset	FileName	USN	Timestamp	Reason	MFTReference	MFTReferenceSeqNo
1	0x000000F0	logico.txt	240	05/09/2016 11:18:27.0763779	BASIC_INFO_CHANGE	39	1
2	0x00000140	logico.txt	320	05/09/2016 11:18:27.0763779	BASIC_INFO_CHANGE+CLOSE	39	1

Se confirma que *la reason BASIC_INFO_CHANGE* que también encontró el parser Logtimeline, *está involucrada en el cambio*, pero en este caso el TMS que se nos ofrece aparece es dato que se ha registrado en $UsnJrnl, y no el que se ha actualizado.

6.2.7 LogFileParser

El parser LogFileParser para Windows 2.0.0.35, estudia las referencias a la MFT contenidas en el artefacto *$LogFile*, y las amplía con información procedente de la MFT cuando se le información adicional sobre dicho artefacto.

Se recopila gran cantidad de información [A6], lo que permite inferir contenido que se puede encontrar en otros artefactos. Desde el punto de vista forense, esta característica es de alto valor de cara a detectar inconsistencias.

Para analizar el comportamiento del parser se procede a experimentar con el caso de escritura lógica. De la imagen previa a la manipulación, observamos como el parser nos presenta un conjunto de LSN relacionados con el Filename “logico.txt”, que en principio se se presentan de forma cronológica, al contar con un LSN ascendente.

Ilustración 19. Extracto del parser LogFileParse

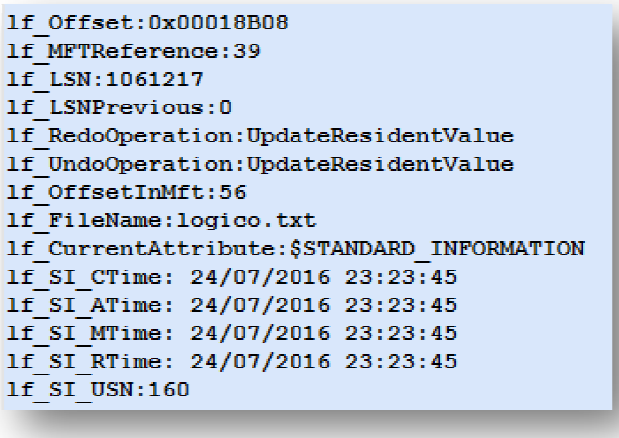
```
lf_MFTReference|lf_LSN |lf_RedoOperation           |lf_FileName|lf_CurrentAttribute
5              |1059622|AddIndexEntryAllocation    |logico.txt |$INDEX_ALLOCATION:$I30
39             |1059646|InitializeFileRecordSegment|logico.txt |$STANDARD_INFORMATION(1)+$FILE_NAME(1)+$DATA(1)
38             |1059774|UpdateNonResidentValue     |logico.txt |$DATA:$J
38             |1059852|UpdateNonResidentValue     |logico.txt |$DATA:$J
39             |1059903|UpdateResidentValue        |logico.txt |$DATA
39             |1059925|UpdateResidentValue        |logico.txt |$STANDARD_INFORMATION
38             |1059996|UpdateNonResidentValue     |logico.txt |$DATA:$J
39             |1060036|UpdateResidentValue        |logico.txt |$STANDARD_INFORMATION
```

Se ejecuta Timestomp sobre el fichero logico.txt:

```
# timestomp f:\logico.txt -z “Sunday 7/24/2016 23:23:45 PM”
```

Sobre el resultado obtenido, tras eliminar los LSN visualizados anteriormente, se encuentra que la operación de manipulación se relaciona con la entrada LSN = 1061217, ya que muestra la actualización del TMS manipulado.

Ilustración 20. Extracto de LogFileParse tras manipulación lógica



A través de la MFTReference suministrado, se puede acceder al atributo $FN de la entrada 39 de la MFT y aplicar las reglas de detección estándar que se aplican sobre la MFT.

Este parser permite el acceso a los datos del Journal clasificados por artefacto, en estructuras como AllTransactionHeaders, INDX_I30, USNJrnl, TxfData, etc., *que facilitan el cruce de datos con los parsers orientados a artefacto.*

En resumen:

Este artefacto recopila información histórica a través de la cual puede realizarse contraste TMS contra las entradas activas en la MFT.

6.3 Conclusiones

Cada artefacto poseedor de TMS, aporta suficiente información como para poder representar un timeline en un momento dado, que puede ser contrastable con el de otros.

En la tabla siguiente se resumen las capacidades de los diferentes parsers analizados para su uso en la detección de actividades antiforenses TMS y en la generación de timelines.

Tabla 6. Uso de parsers Open source en la detección antiforense TMS

Parser	Indicios de manipulación	Aportación
Fls/mactime	Incoherencias en el timeline entre TMS de $SI y $FN, cuando el TMS de $FN son posteriores a los de $SI	Timeline que extrae TMS de la MFT ($SI y $FN)y permite el contraste entre ellos. Se puede correlacionar indirectamente con otras herramientas a través del uso de atributos MFT
Supertimeline	Incoherencias en el timeline entre TMS de $SI y $UsnJrnl	Timeline que combina TMS de MFT con USNJrnl, con un análisis hecho, por lo que está poco orientado a combinarlo con otros parsers externos
Mft2Csv	Incoherencias de TMS entre $UsnJrnl y $LogFile	Timeline detallado de todos los TMS de la MFT. Da la posibilidad de acceder de forma directa a las tablas de Journal a través de sus campos SI_USN y HEADER_LSN
IndxParse	Incoherencia entre TMS de atributos índice y los de la MFT, teniendo en cuenta los diferentes momentos de actualización	Timeline con la información de TMS de los atributos índice de la MFT, con la posibilidad de considerar TMS de archivos borrados a través de su slack space
Usn2CSV	Incoherencia de TMS en secuencias $USNJrnl frente a MFT	Timeline que muestra todos los datos USNJrnl. Extrae los USN del artefacto $USNJrnl, lo que resulta interesante para poder relacionarlo con otros artefactos. Como herramienta generadora de timelines sobre $USNJrnl, es superado por el Supertimeline, ya que relaciona en un mismo reporte tanto los TMS cambiados como las reason del cambio
LogFileParser	Incoherencia TMS entre la MFT referenciada y las operaciones y transacciones contenidas en $LogFile	Timeline con demasiada información, por lo que requiere de un post-procesamiento. Ofrece referencias a nivel de campo a los artefactos importantes del sistema de archivos, con lo que se puede integrar perfectamente con ellos para realizar contraste de la información

La utilización conjunta de las características de algunos de estos parsers *(Mft2CSV, Usn2CSV y LogFileParser)*, se muestra como una buena alternativa para el desarrollo de un sistema de detección del falseamiento de TMS de archivos.

7. Metodología

Se propone una metodología para el uso de artefactos NTFS en la detección de falseamiento de TMS a través del uso de parsers Open Source sobre los mismos, aplicable a los escenarios antiforenses propuestos: las escrituras lógicas, y las directas al dispositivo.

7.1 Modelo de correlación

Como se ha visto anteriormente, existen varios los artefactos NTFS donde se pueden encontrar referencias directas o indirectas a los TMS de los archivos, muchos de ellos siempre presentes, como la MFT y sus atributos (que incluyen a los atributos índice), y el Journal NTFS. Otros artefactos como el Journal USN, son importantes desde el punto de vista forense, pero no es infrecuente que estén inactivos.

La no activación de artefactos o el simple hecho de que implementen su propio sistema de "reciclaje" de datos cuando alcanzan el máximo de su tamaño permitido, origina dado el caso, la pérdida de datos históricos, que puede ser salvada cuando existen registros de los mismos en otros artefactos.

La metodología plantea un modelo de correlación para acceder a estos registros duplicados, planteando diferentes opciones de detección, que sean válidas ante la ausencia de fuentes de registro, lo que permite cubrir mayores casuísticas de imágenes de dispositivos.

En base a esto, se han desarrollado diversas metodologías que los utilizan para la detección de manipulaciones en sus TMS, como las realizadas sobre los números de secuencia de los registros de la *MFT* [12] (Willassen, 2008), sobre los patrones de detección en el journaling *LogFile* [34] (Cho, 2013), o sobre atributos *Indice* de directorios [35] (Minnaard, 2014).

Las relaciones existentes entre la MFT y los otros artefactos NFTS de interés en relación con los TMS, se pueden resumir como:

- El atributo $SI de la MFT, contiene el *USN* que referencia al registro del artefacto *$USNJrnl* más reciente relacionado con dicha entrada.
- El atributo *$INDEX_ALLOCATION*, contiene referencias TMS de su *directorio padre y de sus nodos hijos.*
- El atributo $SI de la MFT, contiene el **LSN** que apunta al valor más reciente de la entrada *$LogFile* que contiene datos históricos relacionados con la entrada MFT.
- El atributo $FN de la MFT, contiene el *nombre del archivo* y referencias a la entrada de su padre en la MFT.

- La número de la entrada en la MFT del archivo/directorio, y su número de secuencia, son utilizados en el contenido de muchos de los artefactos NTFS para referenciarse con la MFT.

También podemos encontrar metodologías que utilizan correlaciones entre varios artefactos, como la Triforce [27] o la propuesta por Uijtewaal & Prooijen (2016) [36].

El modelo Open Source propuesto por Uijtewaal & Prooijen [36], toma como referencia los artefactos $LogFile, $MFT y $UsnJrnl, estableciendo relaciones entre ellos, y proponiendo ecuaciones para operar con las mismas:

```
MFT entry = [VCN x Tamaño Cluster] + [[MFT Cluster Index x Tamaño Bloque] / Tamaño entrada MFT]

$MFT.$Standard_Information.USN = $UsnJrnl.USN

$LogFile.FileReference.[MFT entry] = $MFT.[MFT.entry]
```

Teniendo en cuenta los resultados del análisis de los parsers Open Source, se constata que es posible realizar una automatización en torno a *LogFileParser* teniendo en cuenta las relaciones de Uijtewaal & Prooijen [36].

Como regla metodológica *para la detección de casos basados en Log, se considera que un TMS de Creación histórico no puede ser posterior al actual*, ya que en los cambios de TMS realizados por el sistema (salvo los conocidos casos de copia de TMS), se actualiza al TMS del sistema de ese momento, resultando siempre posterior (salvo manipulación de reloj del sistema) al de cualquier otro dato histórico anterior.

7.2 Detectores

En torno a los parsers propuestos y a los artefactos que se relacionan con ellos, se define el conjunto de parsers que implementan la metodología.

7.2.1 Detector MFT

Define una heurística que sólo emplea el artefacto MFT, y se basa el contraste entre los TMS de creación de los atributos $SI y $FN de la MFT.

Tal y como se muestra en [A3], el TMS de creación de $SI se actualiza con el TMS del sistema cuando se crea o copia un archivo, y el de $FN [A4] además cuando se mueve en local. Como se vio en la sección anterior del análisis del parser FLS, el truco del movimiento

local en las escrituras lógicas, copia los TMS de $SI a $FN, por lo que la información de los TMS de creación de $SI y $FN es en muchos casos coincidente.

Muchas de las herramientas para la modificación de TMS por escritura lógica, sólo modifican los TMS de $SI (como Timestomp), ya que no existe una función API en Windows que permita cambiar directamente los de $FN, lo que da pie a considerar como mecanismo de detección la diferencia entre los TMS de creación de $SI y $FN.

Detectable cuando: $SI.Creation TMS <> $FN.Creation TMS

Puesto que $FN puede contener TMS obsoletos, este detector es propenso a falsos positivos. Esta situación se compensa en la metodología con el ajuste del coeficiente de confianza y de sensibilidad asignados al detector, y su eficacia se verá apoyada por las capacidades de detección de los otros detectores empleados en la metodología.

7.2.2 Detector MFT/USN

El Journal de cambios USN registra operaciones sobre archivos en escrituras lógicas, y a través de sus “reasons” (identificadores del tipo de operación), se tiene constancia de la realización de cambios en los TMS. Las escrituras directas tienen como objetivo no dejar indicios de la manipulación, por lo que no resulta habitual encontrar manipulaciones sobre este registro, lo que a priori concede cierto grado de veracidad a la información contenida.

La correlación utilizada entre MFT y USN, es la utilizada por Uijtewaal [36] para los dos artefactos, $SI.SI_USN=$USNJrnl.USN, que referencia al último registro USN relacionado con la entrada MFT. Según se observó en el análisis de los parsers, el USN del momento final en que se dan por sentado los cambios en los TMS, está referenciado por la *reason* BASIC_INFO_CHANGE+CLOSE, siendo además coincidente con el contenido $SI.SI_USN tras la manipulación.

En momentos posteriores, tras la realización de otras operaciones con el archivo, el valor de $SI.SI_USN cambiaría para referenciarlas, por lo que la detección ya no sería efectiva.

Detectable cuando: $USNJrnl.Reason = “BASIC_INFO_CHANGE+CLOSE” (si fue la última operación realizada sobre el archivo)

Se considera a priori como de *media sensibilidad y confianza media*, ya que debe apoyarse en otros detectores que pongan de manifiesto cambios en el TMS de Creación.

7.2.3 Detector MFT/USN/LOG

Se trata de una versión del detector MFT/USN, que considera la información contenida en el Log de NTFS, lo que permite una proyección histórica del detector MFT/USN.

La correlación se implementa como $USNJrml.MFTReference=$MFT.MFTRecordNumber, apoyada en $USNJrml.FileName=$MFT.FN_FileName, lo que permite considerar el histórico del log USN asociado con la entrada MFT, asegurando a través del nombre del archivo que no se hacen referencias indebidas ante el caso de reutilización de la entrada por otro archivo.

> Detectable cuando: $USNJrnl.Reason = "BASIC_INFO_CHANGE+CLOSE"
> (ante cualquier operación de cambio TMS registrada en el Log NTFS sobre el archivo)

Mejora la sensibilidad de MFT/USN, apoyándose en un tercer artefacto para su detección, lo que permite obtener indicios frente a la desaparición de datos del USN ante el reciclaje del espacio del objeto.

Se considera a priori como de *sensibilidad alta y de confianza baja*, y al igual que MFT/USN, debe apoyarse en otros detectores que pongan de manifiesto cambios en el TMS de Creación.

7.2.4 Detector MFT/LOG/IDX

Esta detección toma referencias sobre los TMS de creación en base a información contenida en el Log NTFS sobre los atributos índice de los directorios de MFT.

Los atributos índices se actualizan en diferentes momentos con la información contenida en $SI [30], en respuesta a requerimientos de información como tras la ejecución de NtQueryInformationFile [30] o ZwSetinformationFile [37].

El detector utiliza los TMS ($SI) del archivo contenido en el Log, para contrastar con los análogos del archivo referenciado, utilizando la correlación del número de entrada y de

secuencia de la MFT. Se considera que el TMS de Creación $SI contenido en el log no puede ser posterior al que posee el registro en la MFT, ya que evidenciaría un atraso de la fecha de creación, y por tanto una manipulación.

La correlación utiliza sólo el número de la entrada en la MFT y su número de secuencia, no utilizando como FileName como en el detector anterior. Se diseña para hacerlo persistente ante cambios de nombre del archivo, y actúe de forma complementaria a MFT/USN/LOG.

> Detectable cuando: $SI.Creation TMS < $INDEX_ALLOCATION.SI_Creation TMS (con los datos de índices contenidos en Log NTFS, relacionados por el número de entrada y secuencia MFT)

Se considera a priori como de *sensibilidad media y confianza alta*, no requiere apoyo de otros detectores que pongan de manifiesto cambios en el TMS de Creación.

7.3 Indicador

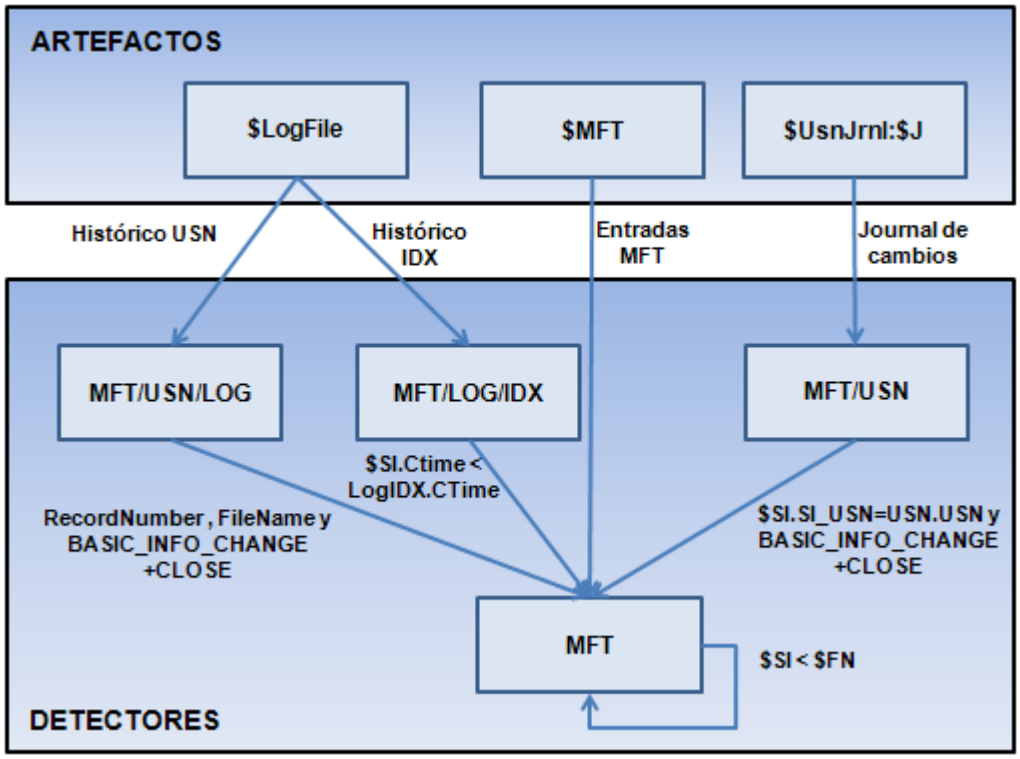
La metodología define un indicador de detección basado en la capacidad de inferencia de la herramienta, y es aplicable a cada uno de los archivos del sistema, referenciados a través de entradas de la MFT.

> **Nivel de Credibilidad de Archivo:** nivel ALTO, MEDIO o BAJO, según los detectores involucrados en detecciones positivas al procesar un archivo

A través de los casos de prueba se va realizando el ajuste de indicadores hasta encontrar un nivel adecuado de positivos, consistente en:

- Estudio de la tabla de inferencia, realizando ajustes propios a la detección antiforense en procesos automáticos (Zdzichowski et al., 2014) [14].
- Traslado de ajustes a los *niveles de sensibilidad y confianza* de los detectores.

Ilustración 21. Diagrama de la metodología (artefactos y detectores)



8. La herramienta metodológica

La herramienta metodológica realiza la implementación de la metodología, con el objetivo de reportar los resultados de su aplicación sobre una imagen de dispositivo suministrada.

8.1 Procedimiento

Partiendo de una imagen post-mortem perteneciente a un dispositivo con formato NTFS, y contando con un sistema Windows donde se ha instalado la herramienta metodológica junto con los parsers Open Source, y previo ajuste inicial de los coeficientes de confianza de los detectores, se aplica el siguiente procedimiento guiado por la herramienta:

1. Extracción de artefactos de la imagen.

2. Ejecución de los parsers Open Source sobre los artefactos extraídos: *Mft2Csv* sobre $MFT para obtener los datos de referencia sobre la MFT, y de los parsers *IndxParse*, *LogFileParser y Usn2Csv,* sobre el resto de artefactos extraidos, con un volcado de resultados a base de datos (SQLite).

3. Aplicación automática de los patrones del modelo de correlación, confeccionando una lista de “detecciones” que hacen referencia a los patrones de detección positivos, al TMS supuestamente comprometido, y al TMS inferido.

4. Procesamiento de los indicadores, utilizando los coeficientes de confianza aplicables según los patrones empleados en cada caso.

5. Creación de un timeline inferido a partir de entradas de la MFT, decorado con los datos de detección procedentes de los artefactos NTFS.

8.2 Carga de datos

La herramienta requiere contar con los datos parseados de los artefactos para implementar la metodología. Para ello se realiza una procesamiento y carga de datos hacia una base de datos SQLite. A través de línea de comando se suministran los parámetros de entrada, debiendo especificarse entre otros la imagen raw del dispositivos NTFS y el directorio de instalación de los parsers.

```
C:\Users\yose\Documents\ntfs-TMS-eval>tms-evaluador.py

La sintaxis de llamada es tms-evaluador.py -i <inputfile> -o <outputfile> -p <ruta-parsers> -t <tag>

       -i: fichero imagen raw ntfs
       -o: fichero de resultados
       -p: directorio padre donde se encuentran instalados los parsers:
                SleuthKit, Log2timeline, Mft2Csv, LogFileParser, USN2Csv e IndxParse

Ejemplo: tms-evaluador.py -i c:\tms\img.dd -o c:\tms\resultado.csv -p c:\tms\parsers -t test
```

8.3 Análisis de credibilidad

Uno de los objetivos de la metodología es que el análisis sea compatible con la mayor parte de imágenes NTFS de distintos dispositivos, ya puedan contar con hives del registro del Sistema Operativo que aporten información extra, como con volúmenes empleados como meros almacenes de datos, que tengan desactivadas alguna de sus características del sistema de archivos.

El análisis de credibilidad automático que realiza la herramienta, parte de la aplicación de las diferentes heurísticas de los detectores sobre los artefactos, que al combinarse, aumentan la capacidad de detección.

Tabla 7: Niveles de sensibilidad y confianza de los detectores

Detector	Sensibilidad	Confianza	Análisis
MFT	media	alta	Contrasta si los TMS Creación son diferentes entre $SI y $FN Comprueba que el TMS Creación $SI<$FN, utilizando detección conservadora (podría emplearse $SI<>$FN, lo que aumentaría su sensibilidad, pero disminuiría su confianza)
MFT/USN	media	media	Información del artefacto $USNJrnl relacionada con MFT.$SI.SI_USN Cuando se apoya en el detector MFT la credibilidad es alta. Requiere apoyo en el detector MFT. Es de alta sensibilidad si la operación de manipulación es reciente.
MFT/USN /LOG	alta	baja	Información de log relacionada con NumEntrada MFT/FileName Propenso a rebajar la sensibilidad cuando existe renombrado del archivo, pero procesa más datos que MFT/USN Requiere apoyo en los detectores MFT o MFT_LOG_IDX
MFT_LOG_IDX	media	alta	Información de log relacionada con NumEntrada MFT/NumSec MFT Equilibrado, en tanto lo que pierde en sensibilidad, lo gana en confiabilidad al procesar por NumSec. No necesita confirmación con detector MFT.

Los resultados se acumulan y ponderan, partiendo de los niveles de confianza y sensibilidad de los detectores positivos, y se traducen en el *Nivel de Confianza de Archivo*, a través de la aplicación de las fórmulas.

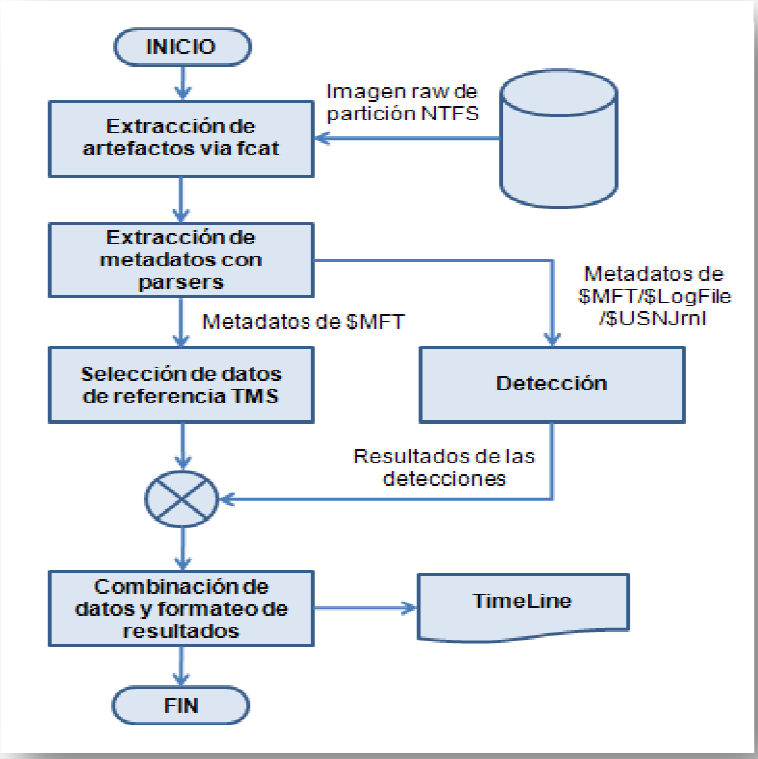
Tabla 8. Fórmulas de Nivel de Confianza de Archivo

ALTO = MFT y MFT/USN y MFT/USN/LOG y MFT/LOG/IDX
MEDIO = (MFT o MFT/USN o (MFT/USN/LOG y MFT/LOG/IDX))
BAJO = MFT/USN/LOG o MFT/LOG/IDX

8.4 Reporte de la herramienta

La herramienta reporta los resultados a través de pantalla, donde muestra los indicadores, y un resumen con las entradas detectadas, y el detalle lo incluye en un fichero en formato CSV, que permite realizar investigaciones más a fondo, ya que incluye referencias a las entradas de la MFT, acompañado con el detalle de los TMS inferidos por cada uno de los detectores.

Ilustración 22. Diagrama de la herramienta (obtención del timeline)



El reporte obtenidos sigue el formato de mactime, separando las entradas $SI de las $FN para que puedan realizarse contrastes visuales.

Ilustración 23. Fichero de resultados CSV

resultados.csv

```
RecordOffset,TMS                    ,Meta,Fichero                          ,status,Tipo      ,MFT_RecNo,MFT_SeqNo
0x00000000  ,17/09/2016 17:59:31.1578715,MACB,.\$MFT                       ,      ,archivo   ,0        ,1
0x00000000  ,17/09/2016 17:59:31.1578715,MACB,.\$MFT ($FILE_NAME)          ,      ,archivo   ,0        ,1
0x00000400  ,17/09/2016 17:59:31.1578715,MACB,.\$MFTMirr                   ,      ,archivo   ,1        ,1
0x00000400  ,17/09/2016 17:59:31.1578715,MACB,.\$MFTMirr ($FILE_NAME)      ,      ,archivo   ,1        ,1
0x00000800  ,17/09/2016 17:59:31.1578715,MACB,.\$LogFile                   ,      ,archivo   ,2        ,2
0x00000800  ,17/09/2016 17:59:31.1578715,MACB,.\$LogFile ($FILE_NAME)      ,      ,archivo   ,2        ,2
0x00000C00  ,17/09/2016 17:59:31.1578715,MACB,.\$Volume                    ,      ,archivo   ,3        ,3
0x00000C00  ,17/09/2016 17:59:31.1578715,MACB,.\$Volume ($FILE_NAME)       ,      ,archivo   ,3        ,3
0x00001000  ,17/09/2016 17:59:31.1578715,MACB,.\$AttrDef                   ,      ,archivo   ,4        ,4
0x00001000  ,17/09/2016 17:59:31.1578715,MACB,.\$AttrDef ($FILE_NAME)      ,      ,archivo   ,4        ,4
0x00001400  ,17/09/2016 17:59:31.1578715,...B,.                            ,      ,directorio,5        ,5
0x00001400  ,17/09/2016 17:59:31.1578715,MACB,. ($FILE_NAME)               ,      ,directorio,5        ,5
0x00001800  ,17/09/2016 17:59:31.1578715,MACB,.\$Bitmap                    ,      ,archivo   ,6        ,6
0x00001800  ,17/09/2016 17:59:31.1578715,MACB,.\$Bitmap ($FILE_NAME)       ,      ,archivo   ,6        ,6
0x00001C00  ,17/09/2016 17:59:31.1578715,MACB,.\$Boot                      ,      ,archivo   ,7        ,7
0x00001C00  ,17/09/2016 17:59:31.1578715,MACB,.\$Boot ($FILE_NAME)         ,      ,archivo   ,7        ,7
0x00002000  ,17/09/2016 17:59:31.1578715,MACB,.\$BadClus                   ,      ,archivo   ,8        ,8
0x00002000  ,17/09/2016 17:59:31.1578715,MACB,.\$BadClus ($FILE_NAME)      ,      ,archivo   ,8        ,8
0x00002400  ,17/09/2016 17:59:31.1578715,MACB,.\$Secure                    ,      ,archivo   ,9        ,9
0x00002400  ,17/09/2016 17:59:31.1578715,MACB,.\$Secure ($FILE_NAME)       ,      ,archivo   ,9        ,9
0x00002800  ,17/09/2016 17:59:31.1578715,MACB,.\$UpCase                    ,      ,archivo   ,10       ,10
0x00002800  ,17/09/2016 17:59:31.1578715,MACB,.\$UpCase ($FILE_NAME)       ,      ,archivo   ,10       ,10
0x00002C00  ,17/09/2016 17:59:31.1578715,MACB,.\$Extend                    ,      ,directorio,11       ,11
0x00002C00  ,17/09/2016 17:59:31.1578715,MACB,.\$Extend ($FILE_NAME)       ,      ,directorio,11       ,11
0x00003000  ,17/09/2016 17:59:31.1578715,MACB,                             ,      ,archivo   ,12       ,12
0x00003400  ,17/09/2016 17:59:31.1578715,MACB,                             ,      ,archivo   ,13       ,13
0x00003800  ,17/09/2016 17:59:31.1578715,MACB,                             ,      ,archivo   ,14       ,14
```

9. Resultados

Los casos de prueba para la validación de la metodología se conformarán tanto en base a manipulaciones del TMS de Creación con escrituras directas (con manipulación de $SI y $FN) y lógicas (con manipulación del $SI), como a comportamientos cotidianos del sistema exentos de manipulaciones, sobre los que deberían limitarse los falsos positivos.

9.1 Análisis sobre las operaciones típicas de uso

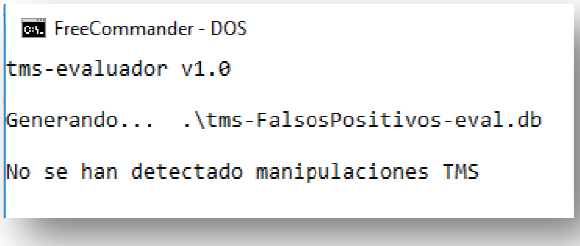
Para solventar la deficiencia de casos de prueba y favorecer el análisis de falsos positivos, se crea una imagen sobre la que se ha ejecutado un guión de operaciones típicas del uso de un dispositivo, y posteriormente se la somete al análisis de la herramienta.

Ilustración 24. Guión de casos de prueba (falsos positivos)

1. Crear el archivo guion.txt
2. Editar el archivo y guardar
3. Copiar guion.txt a guion-copia.txt
4. Crear la carpeta carpeta 1
5. Mover el archivo guion-copia a la carpeta 1
6. Crear carpeta 2
7. Mover carpeta 1 con todo su contenido a carpeta 2
8. Renombrar carpeta 2 a carpeta 3
9. Eliminar la carpeta 3
10. Modificar el atributo de guion a solo lectura
11. Copiar guion.txt a guion-copia.txt
12. Copiar guion-copia.txt a guion-copia-copia.txt
13. Sobreescribir guion-copia-copia.txt al reemplazarlo por otro con el mismo nombre

El resultado de la herramienta es que no detecta casos de manipulación.

Ilustración 25. Resultados sobre falsos positivos



9.2 Análisis de manipulaciones

Se prepara un conjunto de casos con operaciones atómicas realizables sobre archivos [17] y manipulaciones lógicas. Nos centramos en las relacionadas con el TMS de Creación, que es una de las limitaciones que tomado la metodología en su sistema de detección.

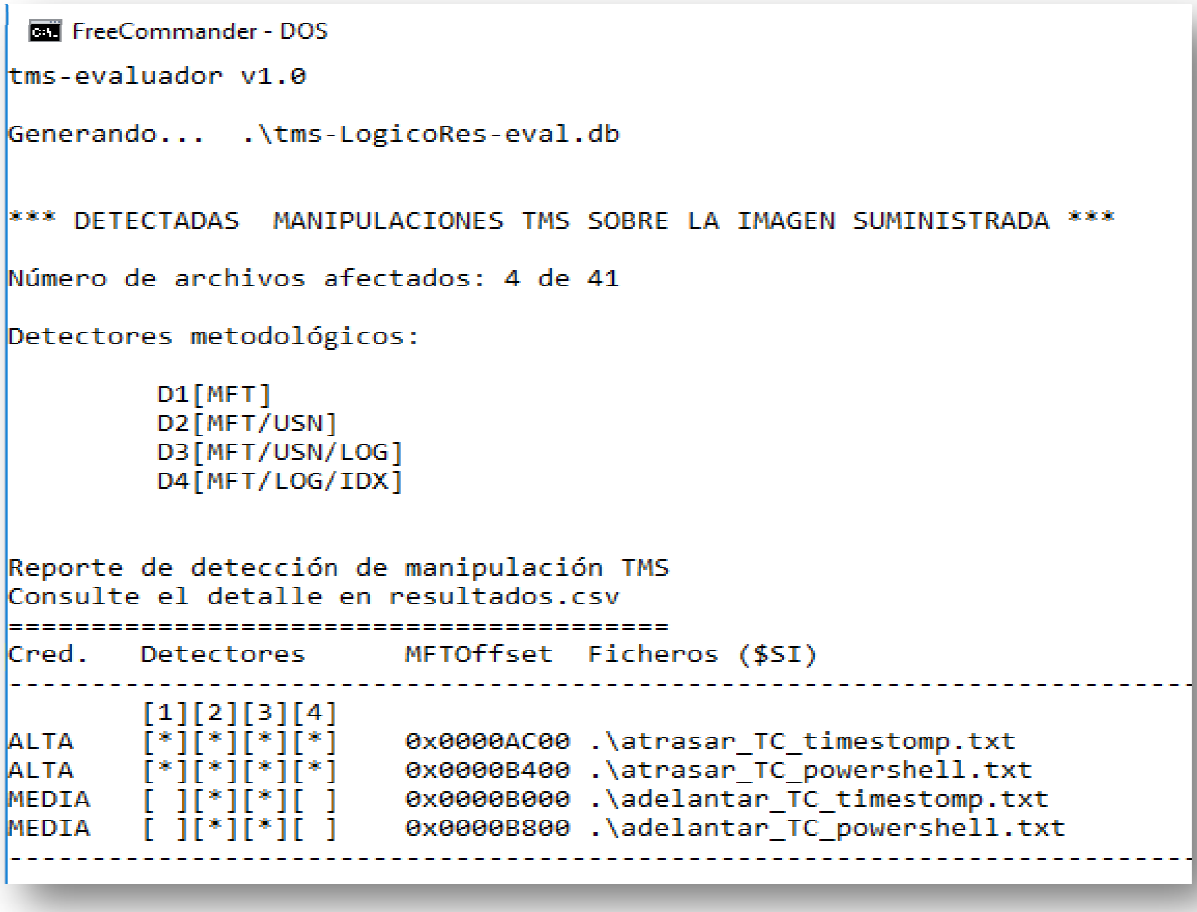
Ilustración 26. Casos de prueba con operaciones y manipulaciones (lógicas y directas)

Archivo	Operación	TMS Creación Original. $SI	TMS Creación Original. $FN	TMS Creación Modific. $SI	TMS Creación Modificado $FN
.\mueve_volumen.txt	Mueve a .\mueve_volumen.txt	17/09/2016 19:59:37	17/09/2016 19:59:37	**17/09/2016 20:09:56**	**17/09/2016 20:09:56**
.\copiame.txt	Copia a .\copiame.txt	17/09/2016 19:59:37	17/09/2016 19:59:37	**17/09/2016 20:10:03**	**17/09/2016 20:10:03**
.\renombrame.txt	Renombra a .\just_renamed.txt	17/09/2016 19:59:37	17/09/2016 19:59:37	17/09/2016 19:59:37	17/09/2016 19:59:37
.\tmp\mueve_local.txt	Mueve a .\mueve_local.txt	17/09/2016 19:59:37	17/09/2016 19:59:37	17/09/2016 19:59:37	17/09/2016 19:59:37
.\borrame.txt	Borrado	17/09/2016 19:59:37	17/09/2016 19:59:37	17/09/2016 19:59:37	17/09/2016 19:59:37
.\atrasar_TC_timestomp.txt	Atraso	17/09/2016 19:59:37	17/09/2016 19:59:37	**08/10/2005 14:34:56**	17/09/2016 19:59:37
.\adelantar_TC_timestomp.txt	Adelanto	17/09/2016 19:59:37	17/09/2016 19:59:37	**17/09/2016 20:09:38**	17/09/2016 19:59:37
.\atrasar_TC_powershell.txt	Atraso	17/09/2016 19:59:38	17/09/2016 19:59:38	**19/07/2016 19:59:38**	17/09/2016 19:59:38
.\adelantar_TC_powershell.txt	Adelanto	17/09/2016 19:59:38	17/09/2016 19:59:38	**17/09/2016 20:09:38**	17/09/2016 19:59:38
.\atrasar_setmace.txt	Atraso	17/09/2016 19:27:49	17/09/2016 19:27:49	**02/07/2016 12:34:56**	**02/07/2016 12:34:56**
.\adelantar_setmace.txt	Adelanto	17/09/2016 19:27:49	17/09/2016 19:27:49	**17/09/2016 21:28:49**	**17/09/2016 21:28:49**

Se constata que las operaciones realizadas por el sistema sobre una misma entrada de un archivo en la MFT, no producen cambios en los TMS de Creación [17].

Se procede a ejecutar la herramienta metodológica sobre los casos de prueba lógicos incluidos en la relación anterior. Los resultados obtenidos se muestran en la ilustración siguiente.

Ilustración 27. Resultado de detección de escrituras lógicas



```
FreeCommander - DOS
tms-evaluador v1.0

Generando...  .\tms-LogicoRes-eval.db


*** DETECTADAS  MANIPULACIONES TMS SOBRE LA IMAGEN SUMINISTRADA ***

Número de archivos afectados: 4 de 41

Detectores metodológicos:

         D1[MFT]
         D2[MFT/USN]
         D3[MFT/USN/LOG]
         D4[MFT/LOG/IDX]


Reporte de detección de manipulación TMS
Consulte el detalle en resultados.csv
=======================================
Cred.   Detectores      MFTOffset  Ficheros ($SI)
-------------------------------------------------------------------------------
        [1][2][3][4]
ALTA    [*][*][*][*]    0x0000AC00 .\atrasar_TC_timestomp.txt
ALTA    [*][*][*][*]    0x0000B400 .\atrasar_TC_powershell.txt
MEDIA   [ ][*][*][ ]    0x0000B000 .\adelantar_TC_timestomp.txt
MEDIA   [ ][*][*][ ]    0x0000B800 .\adelantar_TC_powershell.txt
-------------------------------------------------------------------------------
```

Se detectan los cuatro casos de manipulación lógicos realizados, clasificando los atrasos con un mayor nivel que los adelantos, ya que los adelantos se detectan por el hecho de la existencia de la reason=”BASIC_INFO_CHANGE+CLOSE”, que el detector D2 detecta porque se trata de la última operación realizada sobre el archivo. Si no fuera así, la hubiera detectado D3, pero en este caso estaría clasificada con nivel de credibilidad BAJA.

De igual manera, se ejecutan los casos de escritura directa, sobre los archivos *atrasar_setmace.txt* y *adelantar_setmace.txt*. El resultado es el que se muestra en la ilustración siguiente, revelando que la detección de manipulaciones por escrituras directas (en el caso de Setmace), no es tan eficaz como en la de escrituras lógicas.

Ilustración 28. Resultado de detección de escritura directa

```
FreeCommander - DOS
tms-evaluador v1.0

Generando...  .\tms-FisicoRes-eval.db


*** DETECTADAS  MANIPULACIONES TMS SOBRE LA IMAGEN SUMINISTRADA ***

Número de archivos afectados: 1 de 33

Detectores metodológicos:

         D1[MFT]
         D2[MFT/USN]
         D3[MFT/USN/LOG]
         D4[MFT/LOG/IDX]


Reporte de detección de manipulación TMS
Consulte el detalle en resultados.csv
=====================================
Cred.   Detectores      MFTOffset  Ficheros ($SI)
-------------------------------------------------------------------------------
        [1][2][3][4]
BAJA    [ ][ ][ ][*]    0x00009800 .\atrasar_setmace.txt
-------------------------------------------------------------------------------
```

En escrituras directas sólo es capaz de detectar uno de los casos de atraso a través del detector MFT/LOG/IDX, utilizando entradas de índices no eliminadas del log por la acción de SetMace.

9.3 Timeline generado

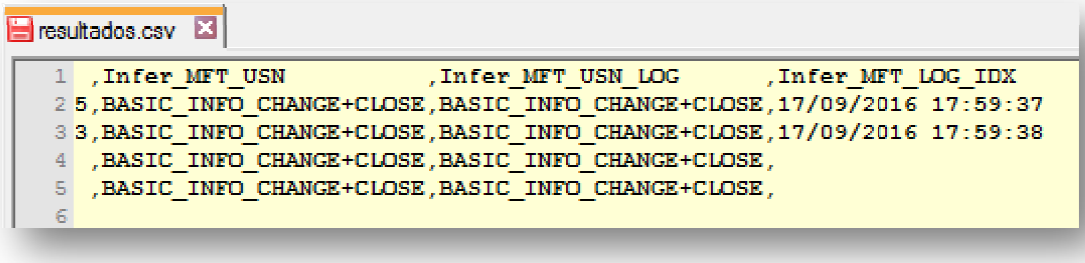
En el timeline generado se detallan las detecciones, estableciendo un TMS inferido, que en el caso del detector MFT representa al TMS de Creación de $FN, en el caso de MFT/LOG/IDX, representa al TMS de Creación que se almacena en el índice (copia antigua de un $SI). Para los casos de MFT/USN y MFT/USN/LOG, no se pronuncia, estableciendo

resultados.csv

	TMS	Fichero	Infer_MFT	
2	08/10/2005 12:34:56.0000000	.\atrasar_TC_timestomp.txt	17/09/2016 17:59:37.9083825	
3	19/07/2016 17:59:38.0023803	.\atrasar_TC_powershell.txt	17/09/2016 17:59:38.0023803	
4	17/09/2016 18:00:37.0000000	.\adelantar_TC_timestomp.txt		
5	17/09/2016 18:09:38.0799383	.\adelantar_TC_powershell.txt		
6				

una referencia a la reason de detección “BASIC_INFO_CHANGE+CLOSE”. En este caso,

hubiera sido posible establecer el del registro de la entrada en el USNJrnl, pero no sería suficientemente preciso para fines forenses.



10. Garantías de aplicabilidad

La metodología se considera aplicable, en cuanto se dispone de una herramienta que la implementa, y que ha sido creada en un lenguaje portable como Python, evitando el uso de software privativo en su desarrollo. Al sustentarse en parsers Open Source, se entiende que existe cierto grado de portabilidad a través del uso de compliaciones de los parsers en diferentes entornos, aunque el desarrollo se haya realizado en un entorno Windows con compilaciones para dicho sistema.

Los resultados obtenidos son referidos al driver de Windows para NTFS 3.1, por lo que es posible que las implementaciones de drivers para otro sistemas operativos el comportamiento pueda variar, atendiendo a la forma en que se actualicen los TMS de los archivos en respuesta a las operaciones sobre ellos.

11. Conclusiones

Este proyecto afronta una problemática de suma importancia en la investigación forense digital que es clave en la reconstrucción de hechos, que pueden resultar decisivos a la hora de concluir si ha existido o no una conducta malintencionada por parte del usuario.

En el presente Trabajo Fin de Máster se ha desarrollado una metodología que permite recopilar indicios de manipulación presentes en diferentes artefactos y utilizarlos de forma conjunta para conformar un sistema de detección.

Los indicadores metodológicos han sido definidos atendiendo a la complementariedad en las detecciones, y se puede considerar que el nivel alto se puede equiparar a una detección

real. Para obtener mayor potencia de detección, se han combinado indicios de fuentes diversas, de tal manera que si se pierde información de un artefacto, o el indicio no es lo suficientemente significativo, se puede complementrar con el de otro.

La metodología ha sido probada y testeada a través de la herramienta y en función de los niveles de sensibilidad y credibilidad asignados de cada uno de los detectores.

Los resultados obtenidos a través de la herramienta *se consideran positivos en la detección de manipulaciones realizadas a través de escrituras lógicas, pero por otro lado, las detecciones sobre manipulaciones por escritura directa han sido parciales y clasificadas con nivel bajo de confiabilidad,* por lo que se considera que aún se requieren mejoras en las heurísticas.

El proyecto ha cumplido su objetivo en tanto que se dispone de una metodología y una herramienta que permiten la detección de manipulaciones, limitado a la detección de manipulaciones sólo sobre TMS de creación, obviando la de otros TMS del archivo.

La aportación realizada por este Trabajo Fin de Máster es una herramienta que emplea la capacidad de detección de diferentes artefactos NTFS y que traduce en “indicadores de compromiso” la manipulación de TMS de archivos, postulándose como una alternativa Open Source de apoyo a los investigadores forenses. Teniendo además en cuenta que los métodos de detección estándar no van más allá del análisis de la MFT y del contraste de los TMS de $SI contra los de $FN.

Finalmente, los resultados de la herramienta no deben tomarse como concluyentes, sino como un elemento de contraste en la tarea del investigador forense sobre la detección y seguimiento del falseamiento de archivos.

12. Mejoras

En principio la metodología ha probado con imágenes NTFS 3.1 formateadas bajo Windows, pero podría mejorarse si se amplia el estudio a imágenes gestionadas por drivers NTFS de otras plataformas (Mac OSX o Linux), con el objetivo de poder afrontar un análisis forense a cualquier dispositivo físico que posea un volumen NTFS, independientemente de su origen.

Las heurísticas empleadas son simples, por lo que una mejora evidente es la aplicación de heurísticas más potentes, que tuvieran por ejemplo presente la afectación producida sobre los TMS de sus directorios padre.

Otra de las posibles mejoras consiste en crear un parser propio sobre los artefactos NTFS e integrarlo con la herramienta, de tal manera que se pueda prescindir del uso de software de terceros.

Enlaces

[1] Carrier, B. (2005). File System Forensic Analysis.

[2] Raghavan, S. (2014). A Framework for Identifying Associations in Digital Evidence Using Metadata

[3] Schatz, B., Mohay, G., Clark, A. (2006). A correlation method for establishing provenance of timestamps in digital evidence, Digital Investigation, Volume 3, Supplement, September 2006, Pages 98-107.

[4] Chen, K., Clark, A., De Vel, O., & Mohay, G. (2003). ECF - Event correlation for forensics. In First Australian Computer Network and Information Forensics Conference.

[5] Gudjonsson, K. (2010). Mastering the super timeline with log2timeline. Sans Institute. Infosec forensic reading room.

[6] Hargreaves, C., Patterson, J. (2012). An automated timeline reconstruction approach for digital forensic investigations. Digital investigation, 9,69-79.

[7] Chabot, Y., Bertaux, A., Nicolle, C., & Kechadi, T. (2015). An Ontology-Based Approach for the Reconstruction and Analysis of Digital Incidents Timelines.

[8] The Internet Society. (2002). RFC 3227. Guidelines for Evidence Collection and Archiving. Recuperado el 2 de julio de 2016 de https://tools.ietf.org/pdf/rfc3227

[9] Case, A., Cristina, A., Marziale, L., Richard, G. & Roussev, V. (2008). FACE: Automated digital evidence discovery and correlation, Digital Investigation 5, 65-75.

[10] Turnbull, B., Randhawa, S. (2015). Automated event and social network extraction from digital evidence sources with ontological mapping, Digital Investigation, Volume 13, June 2015, Pages 94-106

[11] Kälber, S., Dewald, A., & Idler, S. (2014). Forensic Zero-Knowledge Event Reconstruction on Filesystem Metadata.

[12] Willassen, S. (2008). Hypothesis based investigation of digital timestamps

[13] Wikipedia. Edmond Lorcad. Recuperado el 29 de junio de 2016 de http://en.wikipedia.org/wiki/Edmond_Locard

[14] Zdzichowski, P., Sadlon, M., Véiséinen, T., Munoz, A., Filipczak, K. (2014). Anti-Forensic Study

[15] Ballenthin, W. NTFS INDX Parsing. Recuperado el 5 de junio de 2016 de http://www.williballenthin.com/forensics/indx/

[16] Microsoft. File System Behavior in the Microsoft Windows Environment. Microsoft.com. Recuperado el 6 de septiembre de 2016 de http://download.microsoft.com/download/4/3/8/43889780-8d45-4b2e-9d3a-c696a890309f/File%20System%20Behavior%20Overview.pdf

[17] Sans DFIR. Windows forensic analysis poster. Recuperado el 10 de septiembre de 2016 de https://www.sans.org/security-resources/posters/windows-forensics-evidence-of/75/download

[18] Microsoft Development Network. USN_RECORD_V4 structure. Recuperado el 8 de agosto de 2016 de https://msdn.microsoft.com/en-us/library/aa365722.aspx

[19] Gendowne, D., Ivancic, C., Dampier, D. (2012). Who Dun it: A Study of MAC Time Update on NTFS. ASEE Southeast Section Conference.

[20] Solomon A., Russinovich M., Ionescu A. (2012). Windows Internals, Sixth Edition, Part 2. Microsoft Press.

[21] MFT_Segment_Reference. Microsoft Development Network. Recuperado el 1 de septiembre de 2016 de https://msdn.microsoft.com/en-us/library/bb470211%28v=vs.85%29.aspx

[22] Microsoft. FILE_RECORD_SEGMENT_HEADER. Microsoft Development Network. Recuperado el 2 de septiembre de 2016 de https://msdn.microsoft.com/en-us/library/bb470124%28v=vs.85%29.aspx

[23] Windows Incident Response Blog. How do you “do” analysis. Harlan Carvey. 2015. Recuperado el 8 de julio de 2016 de http://windowsir.blogspot.com.es/2015/03/how-do-you-do-analysis.html.

[24] Ballenthin, W., Hamm, J. (2012). Incident Response With NTFS INDX Buffers – Part 4: The internal structures of an Indx attribute. FireEye. Recuperado el 27 de agosto de 2016 de https://www.fireeye.com/blog/threat-research/2012/10/incident-response-ntfs-indx-buffers-part-4-br-internal.html

[25] R. Russon, Y. Fledel (2004). Ntfs documentation. Recuperado el 1 de agosto de 2016 de http://dubeyko.com/development/FileSystems/NTFS/ntfsdoc.pdf

[26] USN_RECORD_V4 structure. Microsoft Development Network. Recuperado el 27 de agosto de 2016 de https://msdn.microsoft.com/en-us/library/windows/desktop/mt684964%28v=vs.85%29.aspx

[27] Hacking exposed computer forensic blog. David Cowen. Happy new year, new post The NTFS Forensic Triforce. Recuperado el 8 de julio de 2016 de http://www.hecfblog.com/2013/01/happy-new-year-new-post-ntfs-forensic.html

[28] NTFS reparse point. Wikipedia. Recuperado el 14 de agosto de 2016. https://en.wikipedia.org/wiki/NTFS_reparse_point

[29] Blocking Direct Wirte Operations to Volumes and Disks. Microsoft Devlopment Network. Recuperado el 2 de septiembre de 2016 de https://msdn.microsoft.com/en-us/library/windows/hardware/ff551353%28v=vs.85%29.aspx

[30] Repositorio de Setmace en Github. Joakim Schicht. Recuperado el 8 de julio de 2016 de https://github.com/jschicht/SetMace/blob/master/readme.txt

[31] Creating, Modifying, and Deleting a Change Journal. Microsoft Development Network. Recuperado el 11 de agosto de 2016 de https://msdn.microsoft.com/en-us/library/aa363877%28v=vs.85%29.aspx

[32] Microsoft. Windows NT Contains File System Tunneling Capabilites. Recuperado el 2 de septiembre de 2016 de https://support.microsoft.com/en-us/kb/172190

[33] W. Ballenthin, J. Hamm. (2012). Incident Response With NTFS INDX Buffers – Part 1: Extracting and Indx attribute. FireEye. Recuperado el 27 de agosto de 2016 de https://www.fireeye.com/blog/threat-research/2012/09/striking-gold-incident-response-ntfs-indx-buffers-part-1.html

[34] Cho, G. (2013). A computer forensic method for detecting timestamp forgery in NTFS

[35] Minnaard, W. (2014). Timestomping NTFS

[36] Uitjewaal, F., Prooijen, J. (2016). UsnJrnl Parsing for File System History

[37] Microsoft. FILE_BASIC_INFORMATION structure. Recuperado el 10 de septiembre de 2016 de https://msdn.microsoft.com/en-us/library/windows/hardware/ff545762(v=vs.85).aspx

Anexos

[A1] Módulo de detección de la herramienta

tms-evaluador.py

```
31
32  class Detector():
33
34      def __init__(self,cnn):
35          self.cnn = cnn
36
37          self.TimeLineInferido = ""
38          self.curTL = self.__CreaTablaTimeLine()
39          self.SQL = ""
40          self.SQL_List = []
41          self.ltag = ""
42          self.params = ""
43
44      def __CreaTablaTimeLine(self):
45
46          try:
47              self.cnn.Execute("Drop table if exists timeline")
48          except:
49              debug(DEBUG_STATUS)
50
51          try:
52              self.cnn.Execute("Drop table if exists resultados")
53          except:
54              debug(DEBUG_STATUS)
55
56          try:
57              self.cnn.Execute(u"CREATE TABLE timeline AS SELECT * FROM " + TAG + "MFT2CSV")
58
59              self.cnn.Execute(u"update timeline set SI_CTime=substr(SI_CTime,7,4) || '-' ||
                substr(SI_CTime,4,2)  || '-' || " +
60              "substr(SI_CTime,1,2) || ' ' || substr(SI_CTime,12, length(SI_CTime)-11)") "
61
62              self.cnn.Execute(u"update timeline set SI_ATime=substr(SI_ATime,7,4) || '-' ||
                substr(SI_ATime,4,2)  || '-' || " +
63              "substr(SI_ATime,1,2) || ' ' || substr(SI_ATime,12, length(SI_ATime)-11)") "
64
65              self.cnn.Execute(u"update timeline set SI_MTime=substr(SI_MTime,7,4) || '-' ||
                substr(SI_MTime,4,2)  || '-' || " +
66              "substr(SI_MTime,1,2) || ' ' || substr(SI_MTime,12, length(SI_MTime)-11)") "
67
68              self.cnn.Execute(u"update timeline set SI_RTime=substr(SI_RTime,7,4) || '-' ||
                substr(SI_RTime,4,2)  || '-' || " +
69              "substr(SI_RTime,1,2) || ' ' || substr(SI_RTime,12, length(SI_RTime)-11)")
70
71              self.cnn.Execute(u"update timeline set FN_CTime=substr(FN_CTime,7,4) || '-' ||
                substr(FN_CTime,4,2)  || '-' || " +
72              "substr(FN_CTime,1,2) || ' ' || substr(FN_CTime,12, length(FN_CTime)-11)")
73
74              self.cnn.Execute(u"update timeline set FN_ATime=substr(FN_ATime,7,4) || '-' ||
                substr(FN_ATime,4,2)  || '-' || " +
75              "substr(FN_ATime,1,2) || ' ' || substr(FN_ATime,12, length(FN_ATime)-11)")
76
77              self.cnn.Execute(u"update timeline set FN_MTime=substr(FN_MTime,7,4) || '-' ||
                substr(FN_MTime,4,2)  || '-' || " +
78              "substr(FN_MTime,1,2) || ' ' || substr(FN_MTime,12, length(FN_MTime)-11)")
79
80              self.cnn.Execute(u"update timeline set FN_RTime=substr(FN_RTime,7,4) || '-' ||
                substr(FN_RTime,4,2)  || '-' || " +
81              "substr(FN_RTime,1,2) || ' ' || substr(FN_RTime,12, length(FN_RTime)-11)")
```

tms-evaluador.py

```
83
84          self.SQL =  u"WITH RECURSIVE mft AS (select timeline.Header_MFTRecordNumber
            RecordNo, timeline.Header_SequenceNo SequenceNo, " +
85                      "timeline.FN_ParentReferenceNo, timeline.FN_ParentSequenceNo, " +
86                      "timeline.FN_FileName from timeline where Header_MFTRecordNumber='5' "
                        +
87                      "union select mft2.Header_MFTRecordNumber, mft2.Header_SequenceNo,
                        mft2.FN_ParentReferenceNo, mft2.FN_ParentSequenceNo,  " +
88                      "mft.FN_Filename || '\\' || mft2.FN_FileName FN_FileName  " +
89                      "from timeline mft2 inner join mft on
                        mft2.FN_ParentReferenceNo=mft.RecordNo and
                        mft2.FN_ParentSequenceNo=mft.SequenceNo where
                        mft2.Header_MFTRecordNumber<>'5') " +
90                      "select * from mft "
91
92          try:
93
94              cur = self.cnn.GetCursor(self.SQL)
95              cur2 = self.cnn.Cursor()
96
97              if not (cur is None):
98                  cols = cur.fetchone()
99
100                 while not (cols is None):
101                     self.SQL = u"update timeline set filepath='" + cols[4] + "' where
                        HEADER_MFTREcordNumber='" +
102                     cols[0] + "' and HEADER_SequenceNo='" + cols[1] + "' "
103                     try:
104                         cur2.execute(u"" + self.SQL)
105                         cols = cur.fetchone()
106                     except:
107                         return None
108
109                 cur2.close()
110                 cur.close()
111             else:
112                 print(u"No funciona el cursor recursivo")
113                 return None
114         except:
115             debug(DEBUG_STATUS)
116             return None
117     except:
118         debug(DEBUG_STATUS)
119         return None
120
121     try:
122         self.SQL = "Select * from timeline "
123         return  self.cnn.GetCursor(self.SQL)
124     except:
125         return None
126
```

tms-evaluador.py

```
126
127
128 -    def __SqlUpd_Validacion_MFT(self):
129          #Sensibilidad: MEDIA
130          #Confiabilidad: ALTA
131          #POSITIVO SI: El Creation time de $SI es anterior al de $FN
132          SQL = "(select CASE WHEN resultados.jtime=julianday(FN_CTime) and resultados.FN='1'
             THEN '' ELSE " +
133          "substr(FN_CTime,9,2) || '/' || substr(FN_CTime,6,2)  || '/' || substr(FN_CTime,1,4)
             || ' ' || substr(FN_CTime,12, length(FN_CTime)-11) END SuspectedTMS " +
134          "from timeline where (julianday(SI_CTime)<>julianday(FN_CTime)) and
             timeline.RecordOffset=resultados.RecordOffset " +
135          "and (instr(resultados.TMS,'B')>0) )"
136
137          return "update resultados set TMSChecked='1', mft_suspectedct=" + SQL + "where exists "
              + SQL
138
139 -    def __SqlUpd_Validacion_MFT_USN(self):
140          #Sensibilidad: MEDIA
141          #Confiabilidad: MEDIA ... Se basa en al último USN registrado por la entrada y existe
             una operación sospechosa
142          #POSITIVO SI: El usn de la entrada MFT apunta a una reason BASIC_INFO_CHANGE+CLOSE
143          SQL = u"update " + TAG + "usnjrnl set [Timestamp]=substr([Timestamp],7,4) || '-' ||
             substr([Timestamp],4,2)  || '-' || substr([Timestamp],1,2) || ' ' ||
             substr([Timestamp],12, length([Timestamp])-11) where instr([Timestamp],'/')>0 "
144
145          self.cnn.Execute(SQL)
146
147          SQL = u"(select CASE WHEN resultados.jtime=julianday(usn.[Timestamp]) and
             resultados.FN='1' THEN '' ELSE " +
148          " 'BASIC_INFO_CHANGE+CLOSE' END SuspectedTMS " +
149          "from timeline timeline " +
150          "inner join " + TAG + "usnjrnl usn  on timeline.SI_USN=usn.usn " +
151          "where usn.reason='BASIC_INFO_CHANGE+CLOSE' " +
152          "and timeline.RecordOffset=resultados.RecordOffset and resultados.FN='0' and
             (instr(resultados.TMS,'B')>0)) "
153
154          return "update resultados set TMSChecked='1', MFT_USN_SuspectedCT=" + SQL + "where
             exists " + SQL
155
156 -    def __SqlUpd_Validacion_MFT_USN_LOG(self):
157          #Sensibilidad: ALTA
158          #Confiabilidad: BAJA
159          #POSITIVO SI: Cuando usando log de  usn,  la entrada MFT apunta a una reason
             BASIC_INFO_CHANGE+CLOSE cruzando con
160          #              MFT.MFTReference y Filename
161          SQL = u"(select CASE WHEN resultados.FN='1' THEN '' ELSE 'BASIC_INFO_CHANGE+CLOSE'
             END SuspectedTMS " +
162          "from timeline timeline " +
163          "inner join " + TAG + "logfile_lfusnjrnl usn on
             usn.MFTReference=timeline.HEADER_MFTRecordNumber and
             usn.FileName=timeline.FN_FileName " +
164          "where usn.reason='BASIC_INFO_CHANGE+CLOSE' " +
165          "and timeline.RecordOffset=resultados.RecordOffset and resultados.FN='0' and
             (instr(resultados.TMS,'B')>0)) "
166
167          return "update resultados set TMSChecked='1', MFT_USN_LOG_SuspectedCT=" + SQL + "where
             exists " + SQL
168
```

tms-evaluador.py

```
168
169 -    def __SqlUpd_Validacion_MFT_LOG_IDX(self):
170          #Sensibilidad: MEDIA... Comprueba tanto $SI como $FN en relación con que el TMS de
             Creación de $SI
171          #              en el log, haya sido en algún momento posterior al de creación que
             figura en MFT.SI_CTime
172          #Confiabilidad: ALTA
173          #POSITIVO SI: El $SI_CTime es anterior al CTime en el log de atributos índice,
             cruzando por entrada MFT
174          #             y Número de secuencia
175 -        SQLInterior = "select timeline.RecordOffset,logs.TMS_Ref  SuspectedCT " +
176                        "from timeline  " +
177 -                      "inner join " +
178 -                          "(select lfI30.lf_MFTReference MFTRecordNo ,
                               lfI30.lf_MFTReferenceSeqNo MFTRecordSeqNo, " +
179                                "substr(lf_CTime,7,4) || '-' || substr(lf_CTime,4,2)  || '-'
                                   || substr(lf_CTime,1,2) " +
180                                "|| ' ' || substr(lf_CTime,12, length(lf_CTime)-11) TMS_Ref "
                                   +
181                                "from " + TAG + "LogFile_INDX_I30 lfI30 " +
182                        ") logs on  logs.MFTRecordNo=timeline.HEADER_MFTRecordNumber and
                           logs.MFTRecordSeqNo=timeline.Header_SequenceNo " +
183                        "Where ((timeline.RecordOffset=resultados.RecordOffset) and
                           (instr(resultados.TMS,'B')>0)) " +
184                        " and (((resultados.FN='0') and
                           (julianday(timeline.SI_CTime)<julianday(logs.TMS_Ref))) )"
185
186          SQL = "(select " +
187          "substr(SuspectedCT,9,2) || '/' || substr(SuspectedCT,6,2)  || '/' ||
             substr(SuspectedCT,1,4) || ' ' || " +
188          "substr(SuspectedCT,12, length(SuspectedCT)-11) SuspectedTMS " +
189          "from ( " + SQLInterior + " ) K where K.RecordOffset=resultados.RecordOffset and
             (instr(resultados.TMS,'B')>0)) "
190
191          return "update resultados set TMSChecked='1', MFT_LOG_IDX_SuspectedCT=" + SQL + "where
             exists " + SQL
192
193
194 -    def __GetFullPathMftEntry(self):
195
196          cur = self.cnn.GetCursor(u"select Header_MFTRecordNumber RecordNo, Header_SequenceNo
             SequenceNo, " +
197          "FN_ParentReferenceNo, FN_ParentSequenceNo, " +
198          "FN_FileName from " + TAG + "mft2csv mft where Header_MFTRecordNumber='5' ")
199
200          cols = cur.fetchone()
201          secuencia=[]
202
203 -        while not (cols is None):
204              RecordNo = cols['RecordNo']
205              SequenceNo = cols['SequenceNo']
206              Filename = cols['FN_FileName']
207
208              secuencia.append(cols)
209
210              SQL = u"select mft2.Header_MFTRecordNumber, mft2.Header_SequenceNo,
                 mft2.FN_ParentReferenceNo, " +
211              "mft2.FN_ParentSequenceNo, '" + Filename + "' || '\\' || mft2.FN_FileName
                 FN_FileName " +
```

tms-evaluador.py

```
212             "from " + TAG + "mft2csv mft2 where mft2.FN_ParentReferenceNo='" + RecordNo + "' " +
213             "and mft2.FN_ParentSequenceNo='" + SequenceNo + "' where
                mft2.Header_MFTRecordNumber<>'5'"
214
215             cur.execute(SQL)
216             cols = cur.fetchone()
217
218         cur.close()
219         return secuencia
220
221     def __GetSecuencia(self, lsn_inicio):
222
223         cur = self.cnn.GetCursor(u"select lf_lsn from " + TAG + "LogFile where lf_lsn='" +
            lsn_inicio +
224         "' and lf_lsnprevious='0' ")
225
226         cols = cur.fetchone()
227         secuencia=[]
228
229         while not (cols is None):
230             lsn = cols[0]
231             secuencia.append(lsn)
232             cur.execute(u"select lf_lsn from " + TAG + "LogFile where lf_lsnprevious='"  + lsn
                + "'")
233             cols = cur.fetchone()
234
235         cur.close()
236         return secuencia
237
238     def __CreaResultados(self,params):
239         """ Exporta en CSV """
240
241         SQL = u"create table resultados as " +
242                 "select pivote.jTime, tl.RecordOffset, " +
243                 "   substr(pivote.dTime,9,2) || '/' || substr(pivote.dTime,6,2)  || '/' ||
                    substr(pivote.dTime,1,4) || ' ' || " +
244                 "   substr(pivote.dTime,12, length(pivote.dTime)-11) TMS_Ref, " +
245                 "   CASE WHEN tl.mSI_MTime=pivote.dTime THEN 'M' ELSE '.' END || " +
246                 "   CASE WHEN tl.mSI_ATime=pivote.dTime THEN 'A' ELSE '.' END || " +
247                 "   CASE WHEN tl.mSI_CTime=pivote.dTime THEN 'C' ELSE '.' END || " +
248                 "   CASE WHEN tl.mSI_BTime=pivote.dTime THEN 'B' ELSE '.' END TMS, " +
249                 "   '0' FN, tl.FilePath Filename, " +
250                 "   CASE WHEN tl.RecordActive='ALLOCATED' THEN '' ELSE '(Eliminado)' END
                    status,  " +
251                 "   CASE WHEN tl.Header_Flags='FOLDER' THEN 'directorio' ELSE 'archivo' END
                    tipoarch, " +
252                 "   tl.Header_MFTRecordNumber RecordNo, tl.Header_SequenceNo SequenceNo, " +
253                 "   '' TMSChecked, '' TMS_Msg, '' MFT_SuspectedCT, '' MFT_USN_SuspectedCT, ''
                    MFT_USN_LOG_SuspectedCT, " +
254                 "   '' MFT_LOG_IDX_SuspectedCT " +
255                 "from  " +
256                 "(  " +
257                 "   select distinct jTime, dTime " +
258                 "   from " +
259                 "   (select julianday(SI_CTime) jTime, SI_CTime dTime from timeline " +
260                 "    union select julianday(SI_MTime), SI_MTime from timeline " +
261                 "    union select julianday(SI_ATime), SI_ATime from timeline " +
262                 "    union select julianday(SI_RTime), SI_RTime from timeline) jLista " +
```

tms-evaluador.py

```
263                 "    order by 1 asc" +
264                 ") pivote " +
265                 "inner join " +
266                 "( " +
267                 "   select  RecordOffset, SI_CTime mSI_BTime,SI_MTime mSI_MTime,  SI_ATime
                    mSI_ATime, SI_RTime mSI_CTime, " +
268                 "          FN_Filename, Header_MFTRecordNumber, Header_SequenceNo,
                    RecordActive,Header_Flags, FilePath " +
269                 "   from timeline  " +
270                 "   order by 1, 2, 3, 4
271                 ") tl  on (pivote.dTime=mSI_BTime or pivote.dTime=mSI_MTime or
                    pivote.dTime=mSI_ATime or pivote.dTime=mSI_CTime) " +
272                 "where pivote.jtime is not null "
273
274 -       SQL = SQL + "UNION " +
275                 "select  " +
276                 "   pivote.jTime ,tl.RecordOffset, " +
277                 "   substr(pivote.dTime,9,2) || '/' || substr(pivote.dTime,6,2)  || '/' ||
                    substr(pivote.dTime,1,4) || ' ' || " +
278                 "   substr(pivote.dTime,12, length(pivote.dTime)-11) TMS_Ref, " +
279                 "   CASE WHEN tl.mFN_MTime=pivote.dTime THEN 'M' ELSE '.' END || " +
280                 "   CASE WHEN tl.mFN_ATime=pivote.dTime THEN 'A' ELSE '.' END || " +
281                 "   CASE WHEN tl.mFN_CTime=pivote.dTime THEN 'C' ELSE '.' END || " +
282                 "   CASE WHEN tl.mFN_BTime=pivote.dTime THEN 'B' ELSE '.' END TMS, " +
283                 "   '1' FN, tl.FilePath || ' ($FILE_NAME)' Filename, " +
284                 "   CASE WHEN tl.RecordActive='ALLOCATED' THEN '' ELSE '(Eliminado)' END
                    status,  " +
285                 "   CASE WHEN tl.Header_Flags='FOLDER' THEN 'directorio' ELSE 'archivo' END
                    tipoarch, " +
286                 "   tl.Header_MFTRecordNumber RecordNo, tl.Header_SequenceNo SequenceNo,  " +
287                 "   '' TMSChecked, '' TMS_Msg, '' MFT_SuspectedCT, '' MFT_USN_SuspectedCT, ''
                    " +
288                 "   MFT_USN_LOG_SuspectedCT, '' MFT_LOG_IDX_SuspectedCT " +
289                 "from  " +
290                 "( " +
291                 "   select distinct jTime, dTime " +
292                 "   from " +
293                 "   (
294                 "       select julianday(FN_CTime) jTime, FN_CTime dTime from timeline " +
295                 "       union select julianday(FN_MTime), FN_MTime  from timeline " +
296                 "       union select julianday(FN_ATime), FN_ATime from timeline " +
297                 "       union select julianday(FN_RTime) , FN_RTime from timeline " +
298                 "    ) jLista " +
299                 "    order by 1 asc " +
300                 ") pivote " +
301                 "inner join  " +
302                 "( " +
303                 "   select  RecordOffset, FN_CTime mFN_BTime,FN_MTime mFN_MTime,  FN_ATime
                    mFN_ATime, " +
304                 "          FN_RTime mFN_CTime, FN_Filename,  Header_MFTRecordNumber,
                    Header_SequenceNo, " +
305                 "          RecordActive, Header_Flags, FilePath " +
306                 "   from timeline  " +
307                 "   order by 1, 2, 3, 4 " +
308                 ") tl  on (pivote.dTime=mFN_BTime or pivote.dTime=mFN_MTime " +
309                 "                or pivote.dTime=mFN_ATime or pivote.dTime=mFN_CTime) " +
310                 "where pivote.jtime is not null " +
311                 "order by 1"
312
```

tms-evaluador.py

```
313             self.cnn.Execute(SQL)
314
315 -           if not self.cnn.Execute(self.__SqlUpd_Validacion_MFT()):
316                 print("No se ejecutó el detector MFT")
317 -           if not self.cnn.Execute(self.__SqlUpd_Validacion_MFT_USN()):
318                 print("No se ejecutó el detector MFT_USN")
319 -           if not self.cnn.Execute(self.__SqlUpd_Validacion_MFT_USN_LOG()):
320                 print("No se ejecutó el detector MFT_USN_LOG")
321 -           if not self.cnn.Execute(self.__SqlUpd_Validacion_MFT_LOG_IDX()):
322                 print("No se ejecutó el detector MFT_LOG_IDX")
323
324
325             SQL = u"select " +
326             "               RecordOffset,TMS_Ref TMS,TMS Meta,Filename Fichero,status,tipoarch
            Tipo,RecordNo " +
327             "               MFT_RecNo,SequenceNo MFT_SeqNo,TMS_Msg Evaluacion,MFT_SuspectedCT
            Infer_MFT," +
328             "               MFT_USN_SuspectedCT Infer_MFT_USN, MFT_USN_LOG_SuspectedCT
            Infer_MFT_USN_LOG, " +
329             "               MFT_LOG_IDX_SuspectedCT Infer_MFT_LOG_IDX " +
330             "       from resultados "
331
332             return  Extractor().ExportaSQL2CSV(self.cnn, SQL, PROJ_PATH + "\\" + params.outputfile.
                replace(".csv","") + ".csv")
333
334 -       def Procesa(self,params):
335
336             return self.__CreaResultados(params)
337
```

[A2] Parámetros del Journal de Cambios obtenidos a través de fsutil

```
C:\Windows\system32>fsutil usn queryjournal f:
Id. de diario de USN    : 0x01d2006ffa1a4f5c
Primer USN              : 0x0000000000000000
Siguiente USN           : 0x0000000000000000
USN mínimo válido       : 0x0000000000000000
USN máximo              : 0x7fffffffffff0000
Tamaño máximo           : 0x0000000000100000
Diferencia de asignación: 0x0000000000040000
Versión del registro mínima admitida: 2
Versión del registro máxima admitida: 4
Seguimiento de intervalos de escritura: habilitado
Tamaño de fragmento de seguimiento de intervalos de escritura: 16384
Umbral de tamaño de archivo de seguimiento de intervalos de escritura: 1048576
```

[A3] Relación entre operaciones sobre archivos y TMS del atributo $STANDARD_INFORMATION [41].

TMS $SI	Si se Renombra	Si se mueve en local	Si se mueve en volumen	Si se copia	Si se accede	Si se modifica	Si se crea	Si se borra
M						*Cambia*	*Cambia*	
A			*Cambia*	*Cambia*	*Cambia (No en Win 7/8)*		*Cambia*	
C				*Cambia*			*Cambia*	
E	*Cambia*	*Cambia*	*Cambia*	*Cambia*		*Cambia*	*Cambia*	

[A4] Relación entre operaciones sobre archivos y TMS del atributo $FILE_NAME [41].

TMS $FN	Si se Renombra	Si se mueve en local	Si se mueve en volumen	Si se copia	Si se accede	Si se modifica	Si se crea	Si se borra
M		*Cambia*	*Cambia*	*Cambia*			*Cambia*	
A			*Cambia*	*Cambia*			*Cambia*	
C			*Cambia*	*Cambia*			*Cambia*	
E		*Cambia*	*Cambia*	*Cambia*			*Cambia*	

[A5] Datos de MFT2CSV extraíbles de la MFT

Atributo	Elementos
Localización del registro	*RecordOffset*
Integridad	*Signature/IntegrityCheck/Style*
VOLUME	*VOLUME_NAME_NAME/VOL_INFO_NTFS_VERSION/VOL_INFO_FLAGS*
HEADER	*HEADER_MFTREcordNumber/HEADER_SequenceNo/Header_HardLinkCount/HEADER_LSN/HEADER_RecordRealSize/HEADER_RecordAllocSize/HEADER_BaseRecord/HEADER_BaseRecSeqNo/HEADER_NextAttribID*
SI	*SI_USN/SI_HEADER_Flags/SI_MaxVersions/SI_VersionNumber/SI_ClassID/SI_OwnerID/SI_SecurityID/SI_Quota*
FN	*FN_ParentReferenceNo/FN_ParentSequenceNo/FN_FileName/FilePath/HEADER_Flags/RecordActive/FileSizeBytes/SI_FilePermission/FN_Flags/FN_NameType/ADS/SI_CTime/SI_ATime/SI_MTime/SI_RTime/MSecTest/FN_CTime/FN_ATime/FN_MTime/FN_RTime/CTimeTest/FN_AllocSize/FN_RealSize/FN_EaSize*
	FN_CTime_2/FN_ATime_2/FN_MTime_2/FN_RTime_2/FN_AllocSize_2/FN_RealSize_2/FN_EaSize_2/FN_Flags_2/FN_NameLength_2/FN_NameType_2/FN_FileName_2
	FN_CTime_3/FN_ATime_3/FN_MTime_3/FN_RTime_3/FN_AllocSize_3/FN_RealSize_3/FN_EaSize_3/FN_Flags_3/FN_NameLength_3/FN_NameType_3/FN_FileName_3
DATA	*DATA_Name/DATA_Flags/DATA_LengthOfAttribute/DATA_IndexedFlag/DATA_VCNs/DATA_NonResidentFlag/DATA_CompressionUnitSize/ DATA_AllocatedSize/DATA_RealSize/DATA_InitializedStreamSize*
	DATA_Name_2/DATA_NonResidentFlag_2/DATA_Flags_2/DATA_LengthOfAttribute_2/DATA_IndexedFlag_2/DATA_StartVCN_2/DATA_LastVCN_2/DATA_VCNs_2/DATA_CompressionUnitSize_2/DATA_AllocatedSize_2/DATA_RealSize_2/DATA_InitializedStreamSize_2
	DATA_Name_3/DATA_NonResidentFlag_3/DATA_Flags_3/DATA_LengthOfAttribute_3/DATA_IndexedFlag_3/DATA_StartVCN_3/DATA_LastVCN_3/DATA_VCNs_3/DATA_CompressionUnitSize_3/DATA_AllocatedSize_3/DATA_RealSize_3/DATA_InitializedStreamSize_3

GUID	*GUID_ObjectID/GUID_BirthVolumeID/GUID_BirthObjectID/GUID_BirthDomainID*
FLAGS	*STANDARD_INFORMATION_ON/ATTRIBUTE_LIST_ON/FILE_NAME_ON/OBJECT_ID_ON/SECURITY_DESCRIPTOR_ON/VOLUME_NAME_ON/VOLUME_INFORMATION_ON/DATA_ON/INDEX_ROOT_ON/INDEX_ALLOCATION_ON/BITMAP_ON/REPARSE_POINT_ON/EA_INFORMATION_ON/EA_ON/PROPERTY_SET_ON/LOGGED_UTILITY_STREAM_ON*

[A6] Atributos de $LogFile extraíbles por LogFileParser

Función	Nombre del atributo
Desplazamiento en la tabla	*lf_Offset*
Referencias a la MFT	*lf_MFTReference,lf_RealMFTReference, lf_MFTBaseRecRef, lf_OffsetInMft, lf_MftClusterIndex*
Lista de registros	*lf_LSN, lf_LSNPrevious*
Nombre del archivo	*lf_FileName*
Operaciones REDO/UNDO	*lf_RedoOperation, lf_UndoOperation*
TMS ($FN+$SI)	*lf_FN_Ctime, lf_FN_Atime, lf_FN_Mtime, lf_FN_RTime*
	lf_SI_Ctime, lf_SI_Atime, lf_SI_Mtime, lf_SI_RTime
Virtual Clusters	*lf_DT_StartVCN, lf_DT_LastVCN*
Cambio de nombre	*lf_FileNameModified*
Transacción	*lf_transaction_id*
Números de cluster virtual y logico del target	*lf_target_vcn, lf_target_lcn*