NTFS Analysis - Digital Forensics : TryHackMe Walkthrough

22 min read · Mar 7, 2025

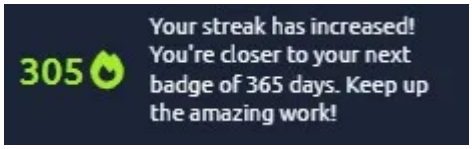
RosanaFSS Follow

Listen Share

Explore the NTFS file system, its layout, and important components.

March 7, 2025

Image built based on TryHackMe´s property.



NTFS Analysis is part of my 305th day on TryHackMe. It is classified as a medium-level walkthrough, and premium: for subscribers only. Can be accessed through the link below. Let's get started!

Task 1 . Introduction

NTFS (New Technology File System) is the default file system for Windows operating systems, developed by Microsoft. This file system is known for its robustness, scalability, advanced features like file permissions, encryption, journaling, and support for large files and partitions. NTFS organizes data in a way that makes it efficient for the operating system and significant for forensic investigations.

This image and all the theoretical content of the present article is TryHackMe´s property.

As we dive deep into NTFS, we will learn that it provides a wealth of information about the system we are investigating or about user activities that could come in handy during the investigation. In this room, we will dive into the details of what NTFS is, how it is structured, and how important it is to understand NTFS structure and capabilities from a forensics perspective.

Learning Objective

In this room, we will cover the following learning objectives:
- What is NTFS and its structure?
- What is the Master File Table (MFT) and what information does it contain?

- How to identify the deleted files.
- How to track activities using Journals.

Prerequisites

This room expects users to complete or go through the following rooms:
- FAT32 Analysis
- MBR and GPT Analysis

Let's dive in.

Answer the question below

1.1. Continue to the next task.

```
No answer needed
```

Task 2 . Lab Connection

Before moving forward, start the lab by clicking the Start Machine button. It will take 3–5 minutes to load properly. The VM will be accessible on the right side of the split screen. If the VM is not visible, use the blue Show Split View button at the top of the page.

This image and all the theoretical content of the present article is TryHackMe´s property.

We will examine the NTFS of a live machine attached to this task. The machine has all the required tools.

Answer the question below

2.1. I have successfully turned on the VM.

```
No answer needed
```

Task 3 . NTFS Overview

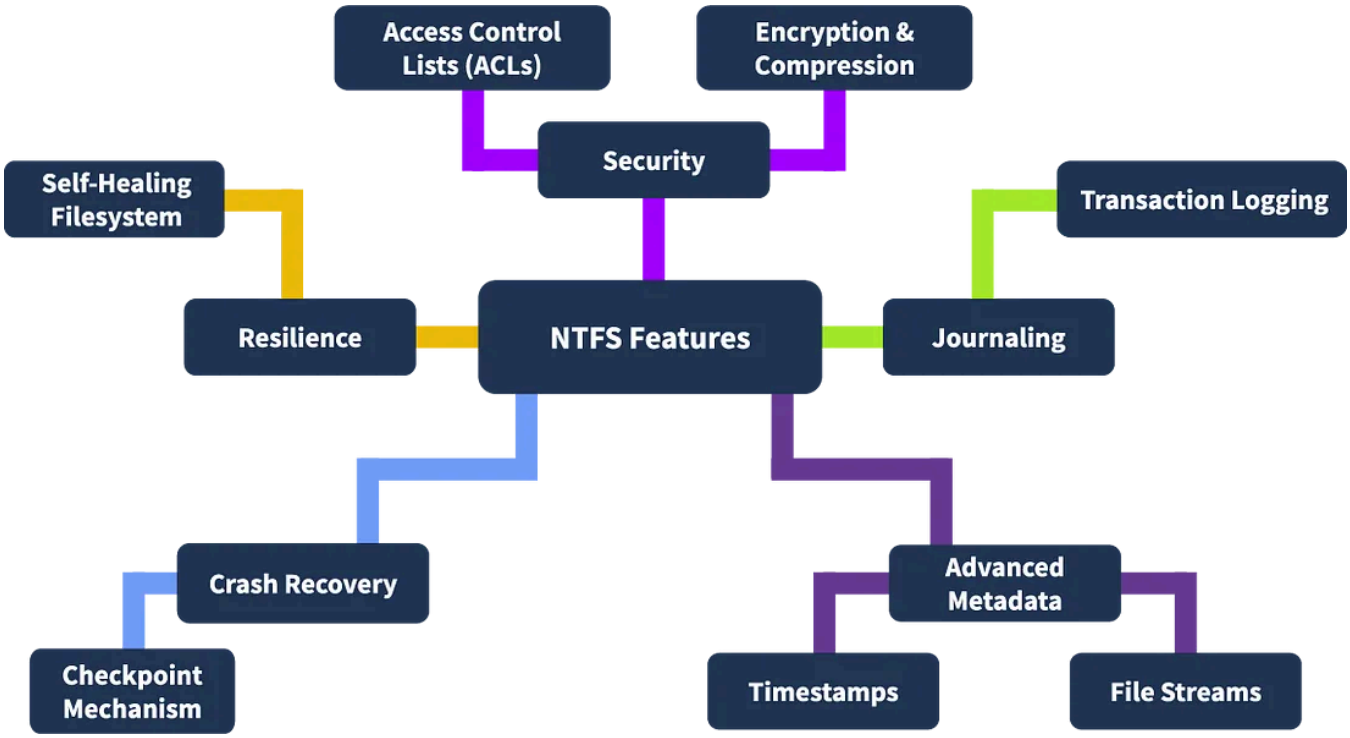
NTFS Overview

As already covered, **NTFS, or New Technology File System, is a file system developed by Microsoft.** It's been the default file system for Windows operating systems since Windows NT 3.1, way back in 1993. **A file system is the way our computer organizes and stores files on a drive.**

It is trivial to understand that NTFS can also be classified as a Journaling file system because it is designed to maintain the file system's consistency by recording or writing down the changes within the file system in a journal (log) before applying them to the main file system structures. From an OS point of view, this is useful when recovering the OS from unexpected system crashes, power failures, etc. NTFS has some features that make it stand out from other file systems.

Some of the key features of NTFS are:

- Advanced Metadata: NTFS maintains detailed metadata, including file creation, modification, and access times, which is invaluable for forensic investigations.
- Journaling: NTFS uses a journaling feature to keep track of changes. This feature helps recover data after crashes and provides a trail of file system activities. This can also be helpful for forensics investigations.
- **Security Features**: NTFS supports advanced security features like encryption (EFS) and permissions, which can be analyzed to understand access patterns and security breaches.
- **Large Volume Support**: NTFS can handle large volumes and files, making it suitable for modern storage needs.
- **Resilience**: NTFS is more resilient to corruption compared to older file systems like FAT32, thanks to its robust structure and recovery mechanisms.



This image and all the theoretical content of the present article is TryHackMe´s property.

NTFS Comparison with Other File Systems

Here's a simple comparison to help you understand how NTFS compares to other file systems:

Feature	NTFS	FAT32	exFAT
Max Drive Size	256 TB	2 TB	128 PB
Max File Size	256 TB	4 GB	16 EB
Crash Recovery	Yes (keeps track of all changes)	No	No
Encryption	Yes (built-in encryption)	No	No
Compression	Yes	No	No
File Permissions	Yes (you can control who accesses what)	No	No
Reliability	High (less likely to corrupt)	Low (older and more prone to issues)	Moderate (better than FAT32 but not as robust as NTFS)
Works Across Devices	Mostly Windows (limited compatibility with Mac/Linux)	Universal (works on almost anything)	Universal (great for flash drives and SD cards)
Best For	Modern Windows systems	Older systems or small USB drives	Flash drives, SD cards, and external drives

This image and all the theoretical content of the present article is TryHackMe´s property.

We have discussed a few points that make NTFS better than other file systems. Let's move on to the next task to explore key components found within NTFS.

Answer the question below

3.1. Which feature does NTFS use to keep track of the changes within the file system?

```
journaling
```

Task 4 . NTFS Components

Before diving into the details of NTFS, it is important to understand how the NTFS disk is actually structured and what are the main components of NTFS.

The NTFS disk is organized into several key components, each serving specific purposes to ensure efficient data storage, retrieval, and file system integrity. From a forensic perspective, many of these components hold critical information valuable in investigations, such as timestamps, deleted files, and traces of system activity.

This image and all the theoretical content of the present article is TryHackMe´s property.

Below is a detailed explanation of the NTFS disk structure, its components, and their forensic significance.

Partition Boot Sector (PBS)

The Partition Boot Sector is the first sector of an NTFS volume and plays a vital role in the booting process. It contains the necessary information to locate and load the operating system. It also includes the BIOS Parameter Block (BPB), which defines the disk layout. It maintains critical file system parameters such as sectors per cluster and the location of the Master File Table.

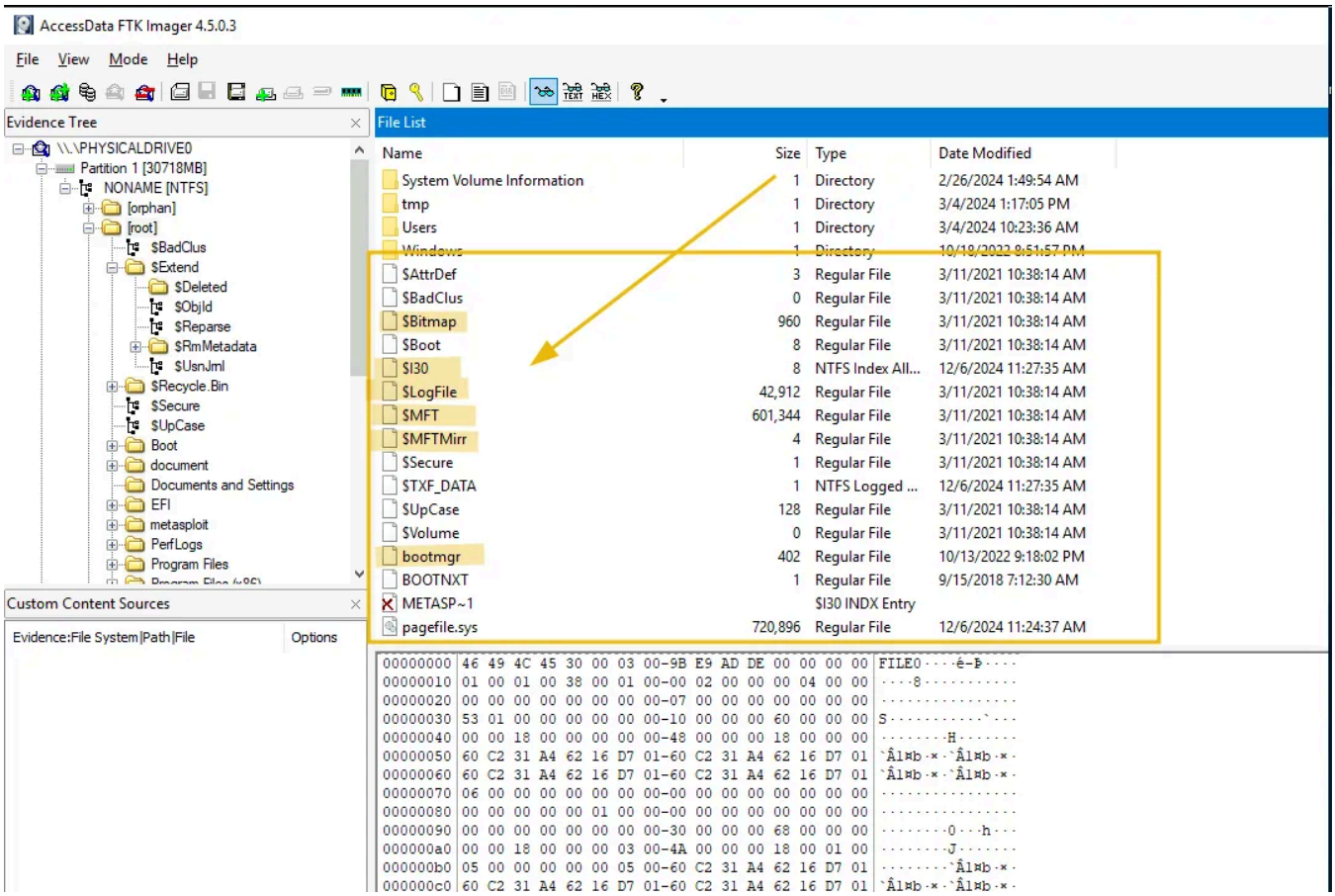
Key Contents

Some of the key components of PBS are:

- Jump instruction to bootstrap code.
- File system type indicator (e.g., NTFS).
- Location of the MFT and the MFT Mirror.
- End-of-sector marker (0x55AA).

Forensics Value

Looking at the PBS from the forensics perspective can reveal information about the boot process and disk configuration. This sector can be analyzed to determine whether it has been tampered with (e.g., by malware or rootkits) or to validate the integrity of the boot code and the BIOS Parameter Block (BPB).



This image and all the theoretical content of the present article is TryHackMe´s property.

Master File Table (MFT)

One of the key components/sectors of the NTFS disk is the Master File Table (MFT). This is considered the backbone of NTFS, acting as a database that contains metadata about every file and directory. It provides a comprehensive structure for the local file data and can be used to track all the files and directories on the disk, including the user data and the system files. We will explore this in the later task.

Characteristics

Some of the key characteristics of MFT are:

— Divided into fixed-size records (typically 1 KB).

— Each file and directory gets an MFT entry.

— File attributes such as timestamps, permissions, and data location are stored as metadata.

Forensics Value

Some of the key forensics values provided by MFT are:

— Deleted file records may still exist in the MFT, aiding data recovery.

— It is essential for timeline analysis, as it contains timestamps for creation,

modification, and last access.
— Useful in detecting unauthorized changes to File metadata.

$MFTMirr

As the name suggests, the MFT Mirror is a duplicate copy of the first few records of the Master File Table, designed to ensure redundancy. Its main purpose is to provide a backup copy to recover data in case the primary MFT becomes corrupted. It also ensures file system stability and reliability. It is typically located in a separate part of the disk.

Forensics Value

From a forensics point of view, it can be used to cross-verify the integrity of the MFT. If the primary MFT is damaged or tempered, it can be used to get useful metadata.

System Files

System files in NTFS are reserved files that manage the internal operations of the file system. Some of the key examples of system files are:
— **$MFT:** Master File Table.
— **$MFTMirr**: MFT Mirror.
— **$LogFile**: A transactional log for consistency.
— **$Bitmap**: Tracks allocated and free clusters.
— **$Boot**: Stores the boot sector information.
— **$BadClus**: Tracks bad sectors.
— **$UpCase**: Stores uppercase character mapping for case-insensitive comparisons.

Forensics Value

From the forensics perspective, some of the files mentioned below are important:
— System files **$LogFile** can reveal a history of file system operations.
— **$Bitmap** It can help identify recent cluster allocations and deletions.
— **$BadClus** It might expose sectors marked badly to hide data intentionally.

File Data Area

The area where user and system file content is stored, either within the MFT (resident) or in separate clusters (non-resident). It stores the actual content of files and directories. It provides flexibility by storing small files directly in the MFT.

Forensics Value

This is the primary source for recovering the user data. It can be analyzed for file

fragments and slack space, which may contain the residual data.

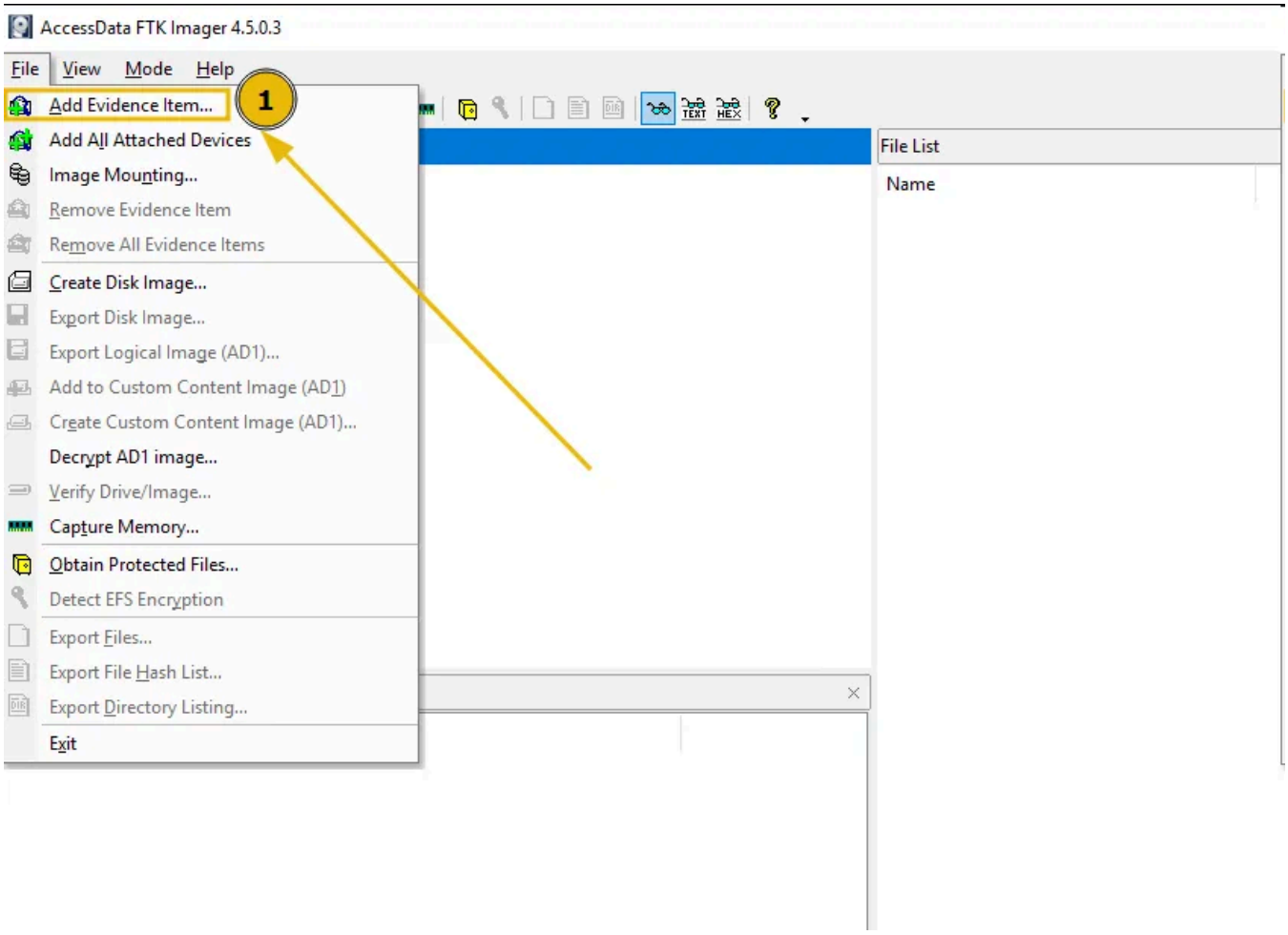
Alternate Data Stream

Alternate Data Streams (ADS) are a feature of NTFS that allows multiple data streams to be associated with a single file or directory. While the primary data stream contains the main content of a file, ADS can store additional metadata or hidden content without altering the size or appearance of the main file.

Identifying the Artifacts

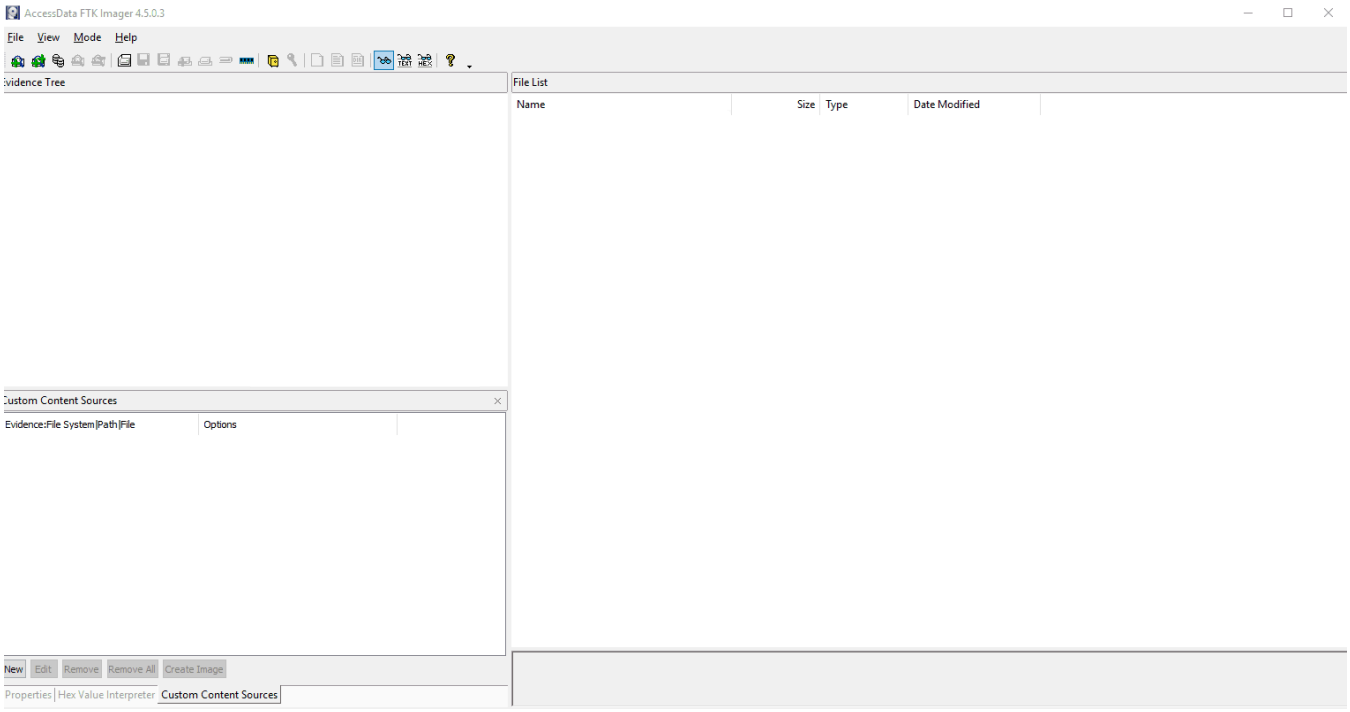
To extract and examine the NTFS disk and its components, we will use the powerful forensics tool called FTK Imager, which can be found on the taskbar.

First things, open FTK Imager, which can be found in the taskbar, and click on File -> Add Evidence Item, as shown below:



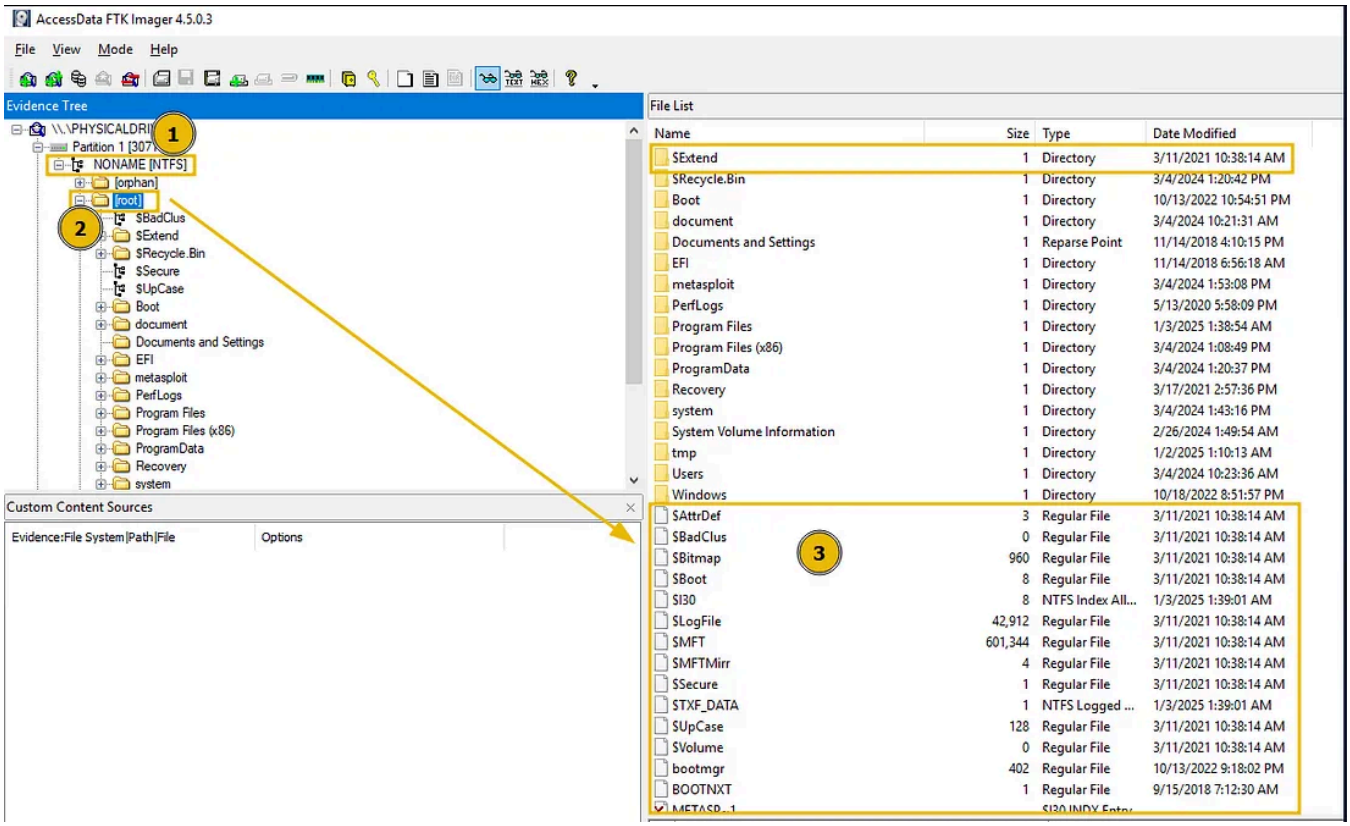
This image and all the theoretical content of the present article is TryHackMe´s property.

Click on File -> Add Evidence Item -> Select Physical Drive -> Select the first option. This will load the live disk as the evidence, which we will be examining, as shown below:



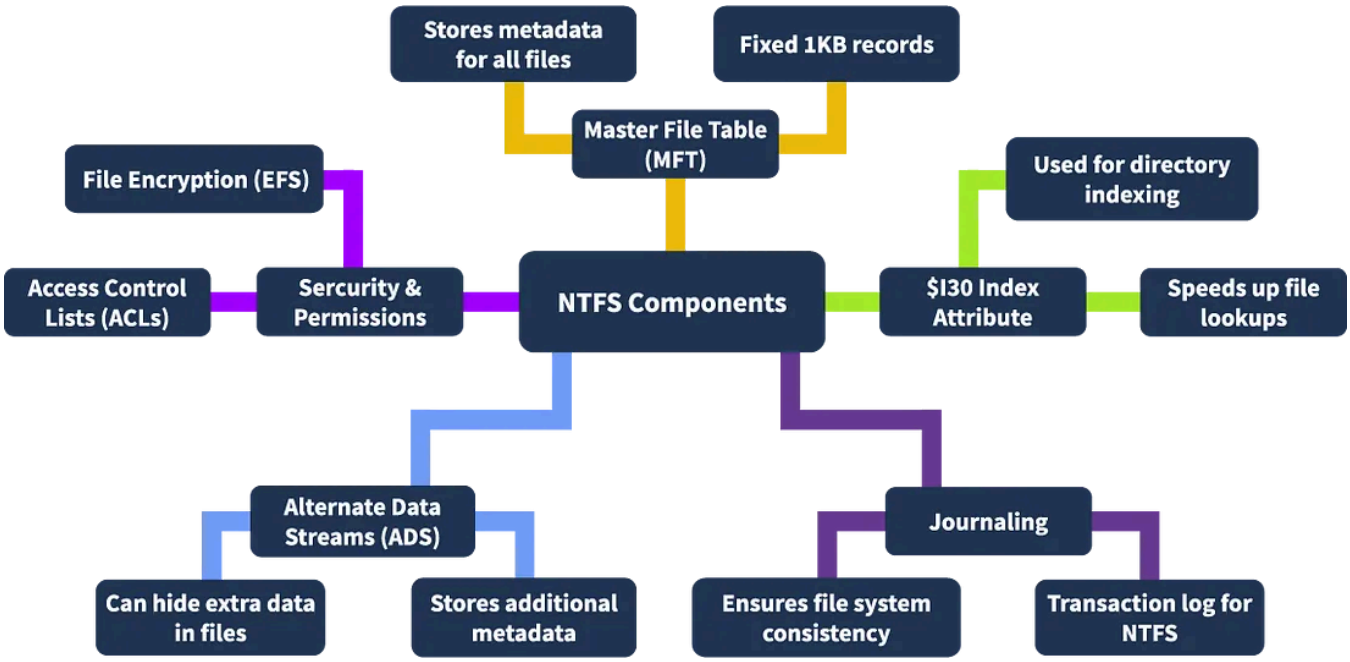
This image and all the theoretical content of the present article is TryHackMe´s property.

Once the disk is loaded, we will click on the **Partition -> NONAME [NTFS] -> [root]** to navigate to the main content of the disk, as shown below:



This image and all the theoretical content of the present article is TryHackMe´s property.

We can see all the major key components of NTFS. In the coming tasks, we will explore some of the most important components.



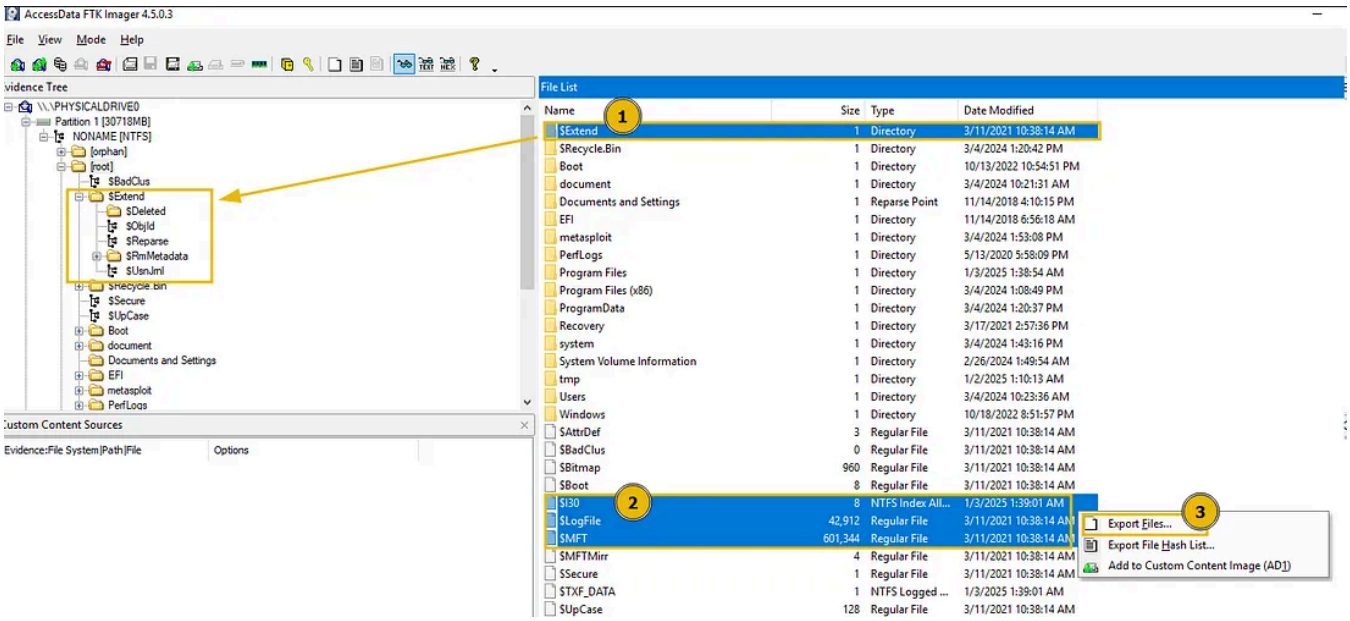
This image and all the theoretical content of the present article is TryHackMe´s property.

Collecting the Artifacts

Earlier, we explored the key components of the NTFS. Let's use the FTK Imager to collect these logs and save the offline copy on the Desktop for further analysis. The components which we will need for the investigation are:

— **$MFT** (Master File Table)
— **$LogFile**
— **$I30** (NTFS Index Attribute)
— **$Extend**
— **$UsnJrnl**

Select, Right Click and Export Files into the Evidence folder on the Desktop, as shown below:

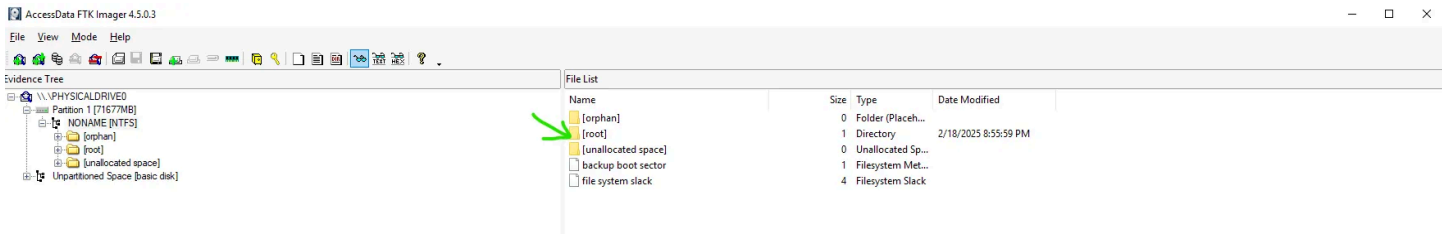


Answer the question below

4.1. Double-click on the $UsnJrnl file in the $Extend folder; what is the first evidence file you find?

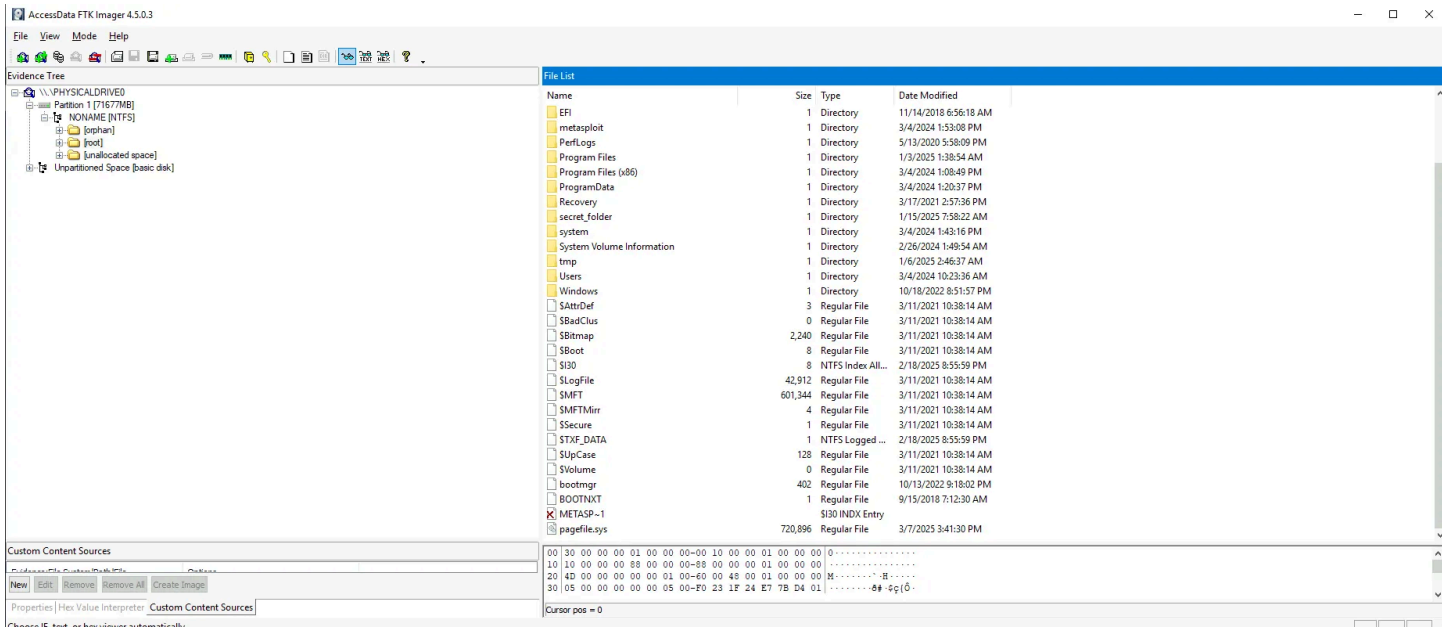
`$J`

— Launched `FTK Imager`.
— Clicked `File`.
— Clicked `Add Evidence Item ...`.
— Selected `Physical Drive`.
— Clicked `Next`.
— Clicked `Finish`.
— Expanded `\\.\PHYSICALDRIVE0`.
— Expanded `Partition 1 [71677MB]`.
— Expanded `NONAME [NTFS]`.
— Double-clicked `[root]` to navigate to the main content of the disk = key componentes of the NTFS.



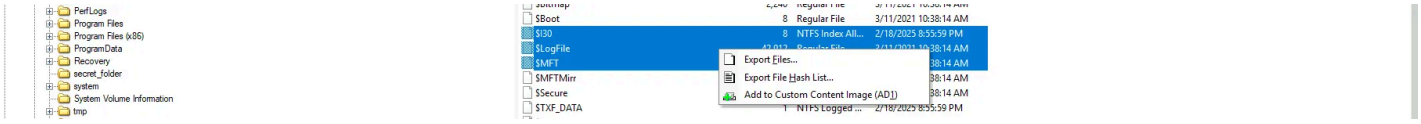
FTK Imager

— Visualized the offline copy of the logs saved in the Desktop.



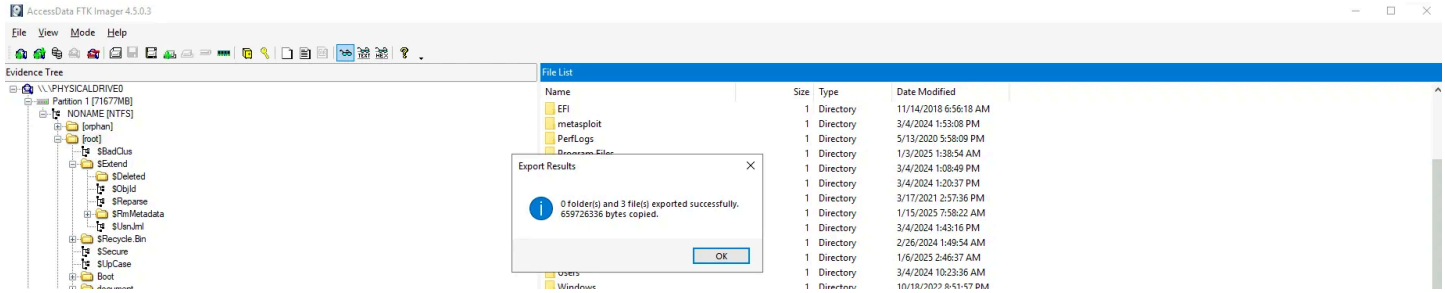
FTK Imager

— Right-clicked and selected `$I30`, `$LogFile`, and `$MFT`.
— Right-clicked, and selected `Export Files`.



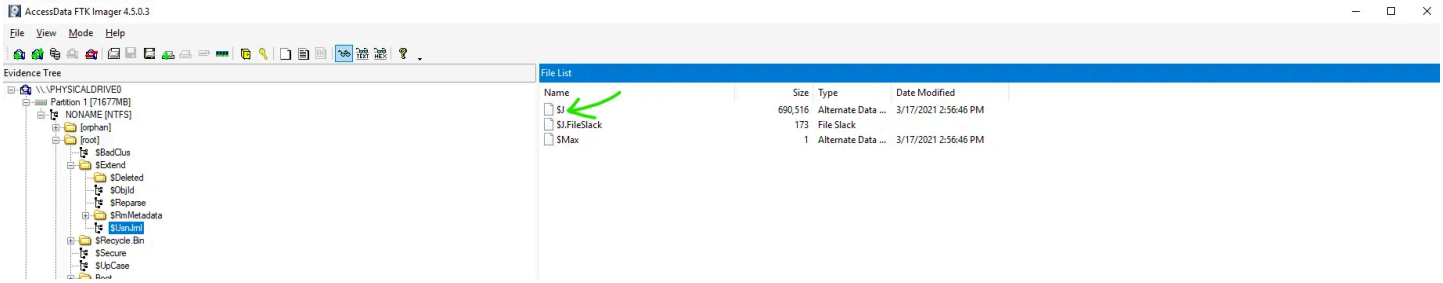
FTK Imager

— Selected the `Evidence` folder on the `Desktop`.
— Clicked `OK`.
— Received a message that `0 folder(s) and 3 file(s) exported successfully`.



FTK Imager

— Clicked on the `$UsnJrnl` file in the `$Extend` folder.
— Discovered `$J` as the first evidence.



FTK Imager

Task 5 . MFT Record Analysis

The Master File Table (MFT) Record is considered the most important file for the forensics investigator because the MFT handles any file created/modified/deleted on the file system. The Windows OS creates an entry for each individual file or folder within the MFT record.

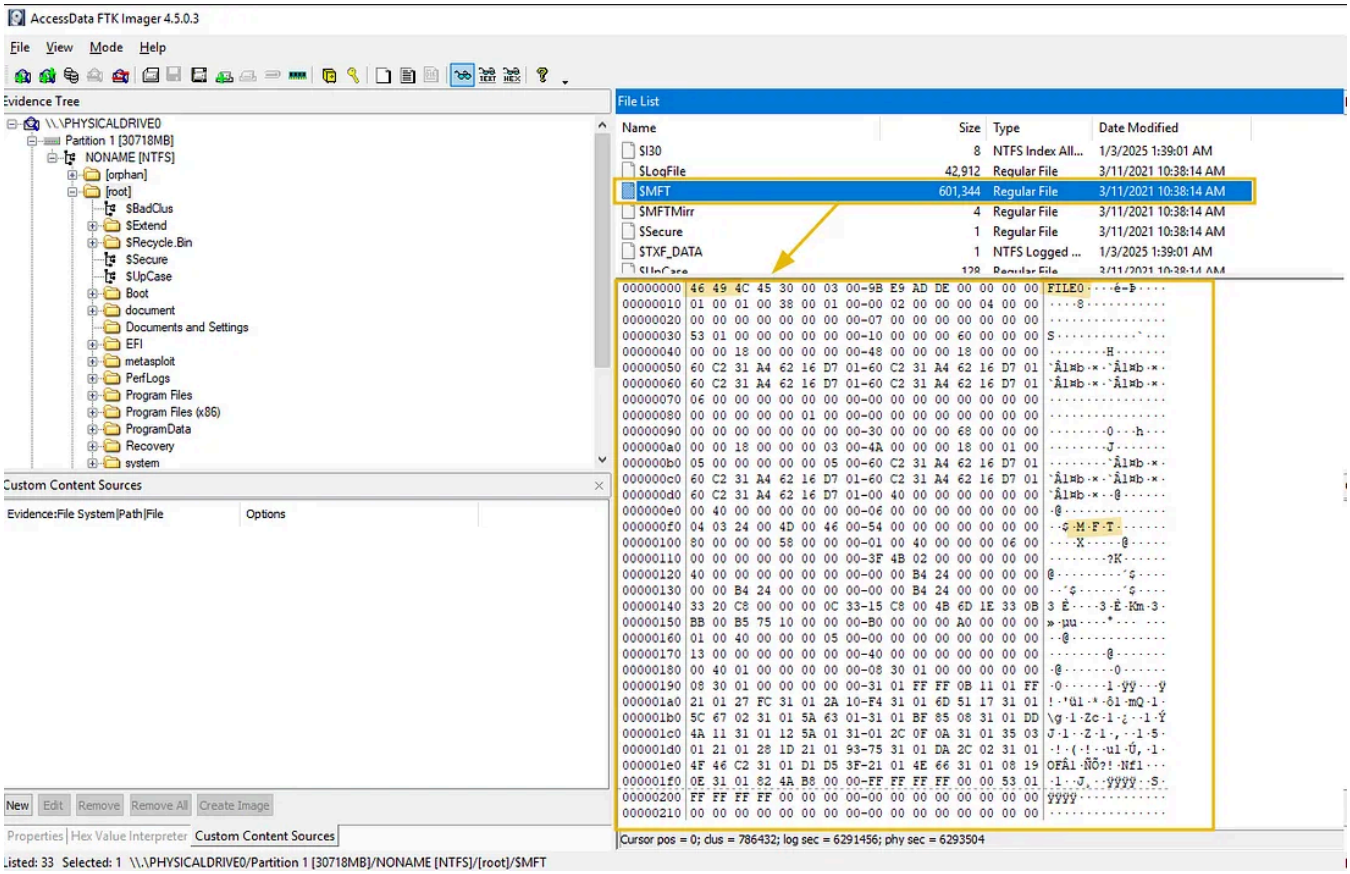
Let's start our investigation by exploring the MFT record, a very important component of NTFS.

Master File Table

The MFT record is a core component of the NTFS, which serves as the database that tracks all files and directories. Each file or directory on the disk has a corresponding MFT record, which acts as a detailed metadata repository for that object.

Contents of an MFT Record

The MFT contains very important information about the records related to the files. Each MFT record contains attributes describing the file or the directory. The Hex representation of a particular record within the MFT File is shown below:



This image and all the theoretical content of the present article is TryHackMe´s property.

The MFT record contains the following information:

— **File Name:** The file's name or the directory contained within.

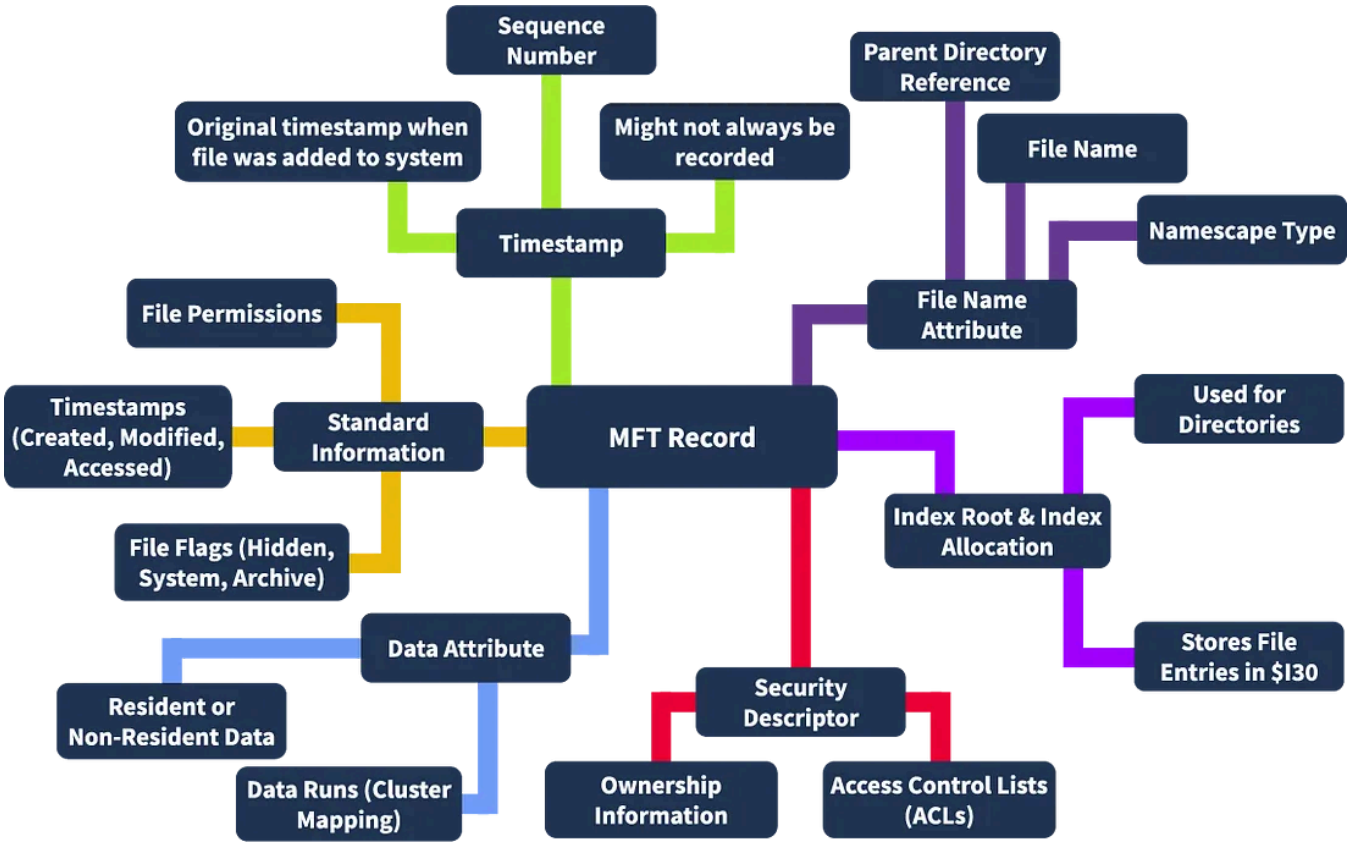
— **Standard Information:** Contains key information about the link count, timestamps, file attributes, etc.

— **Attributes:** Metadata like MACB timestamp, attributes like Read-only, hidden, etc.

— **File Data:** This contains the actual data if the file is small enough to fit within the MFT record; otherwise, it points to the clusters where the data is stored.

— **File Index:** Index information that points to other files or directories.

— **Security Information:** ACL that defines the security permissions for the file or directory.



This image and all the theoretical content of the present article is TryHackMe´s property.

From an OS perspective, MFT File manages the files and directories within the disk. It is used to locate and interpret the files' metadata, such as timestamps, security permissions, attributes, etc. The OS also uses MFT File to effectively access/find the files by storing pointers to the file or data.

Forensics Value

MFT records play a vital role in investigations, as they are considered an invaluable source of evidence because they provide a comprehensive map of the file system activity and the timeline. Some of the key aspects of MFT records, from the forensics point of view, are:

— It provides evidence of the presence of a certain file on the disk, even if the file is deleted.

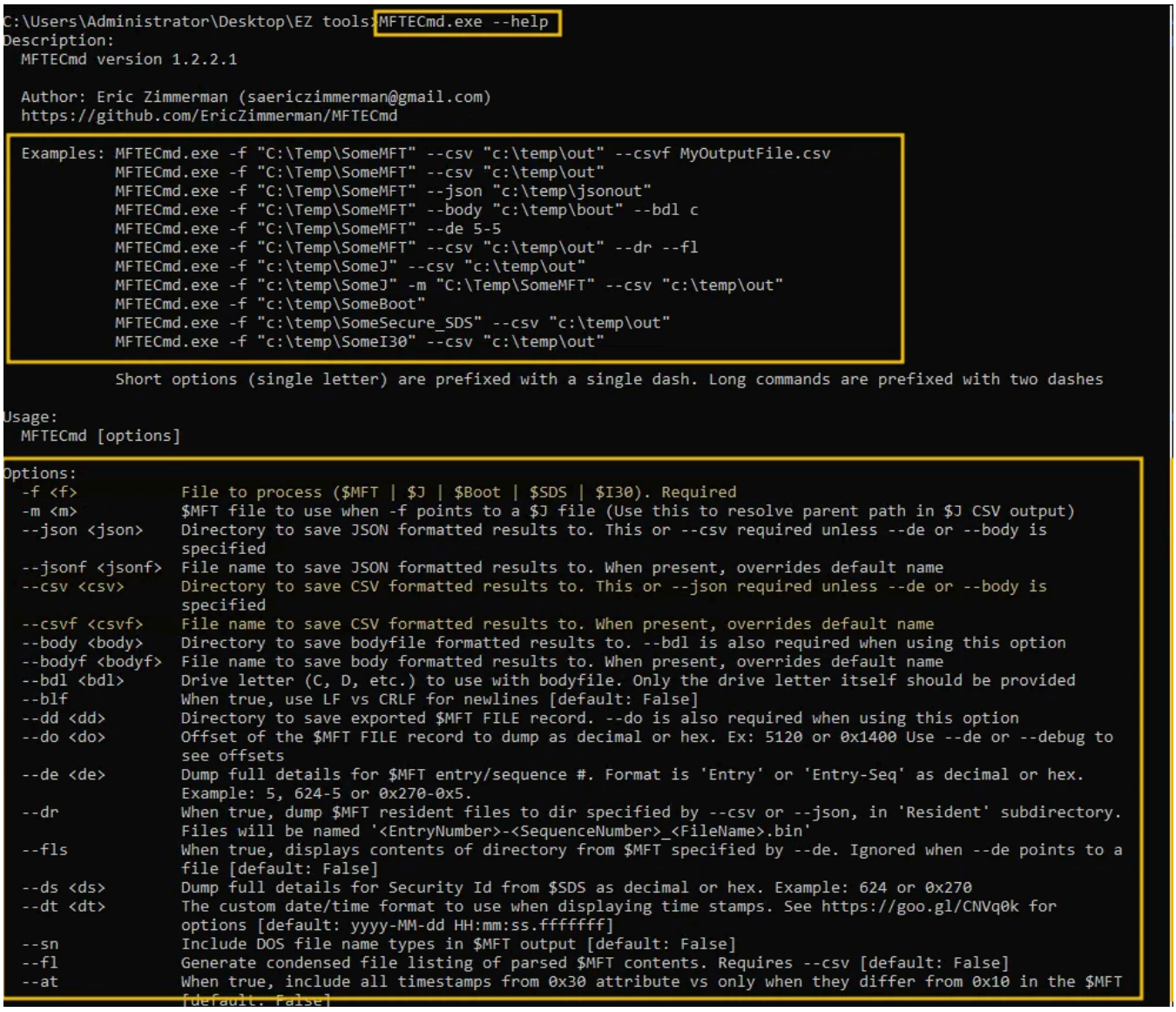
— It can also provide evidence of file modification, which could be useful in certain forensics use cases.

Examining the MFT Record

Now that we have understood the importance of the MFT record, it is time to extract the juicy information from the MFT File we extracted in the previous task and validate all the key findings that we discussed earlier.

To extract the information from the MFT records, we will use the **MFTECmd.exe** tool, which is placed in the EZ Tools directory on the Desktop. Open the terminal and run the following command to check the available options, as shown below:

Command: **MFTECmd.exe — help**

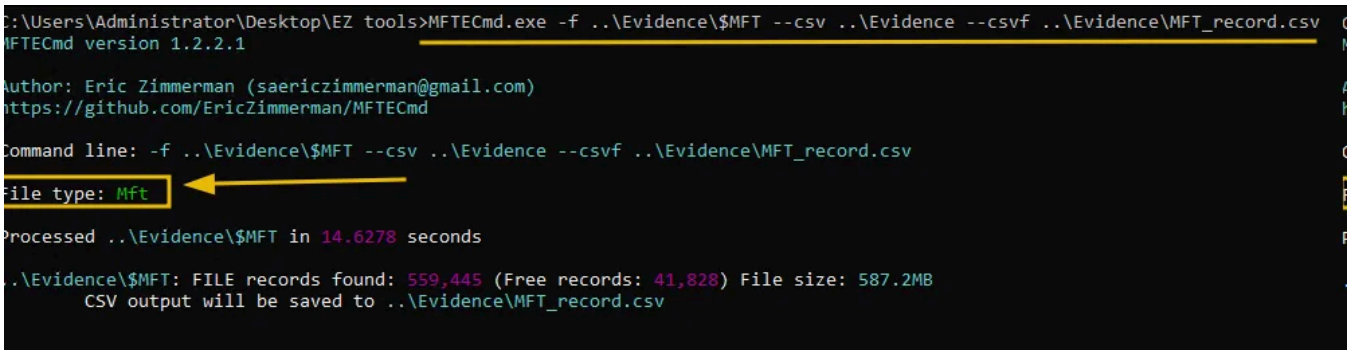


This image and all the theoretical content of the present article is TryHackMe´s property.

As mentioned in the example commands, we will be using the following three flags to create our query:

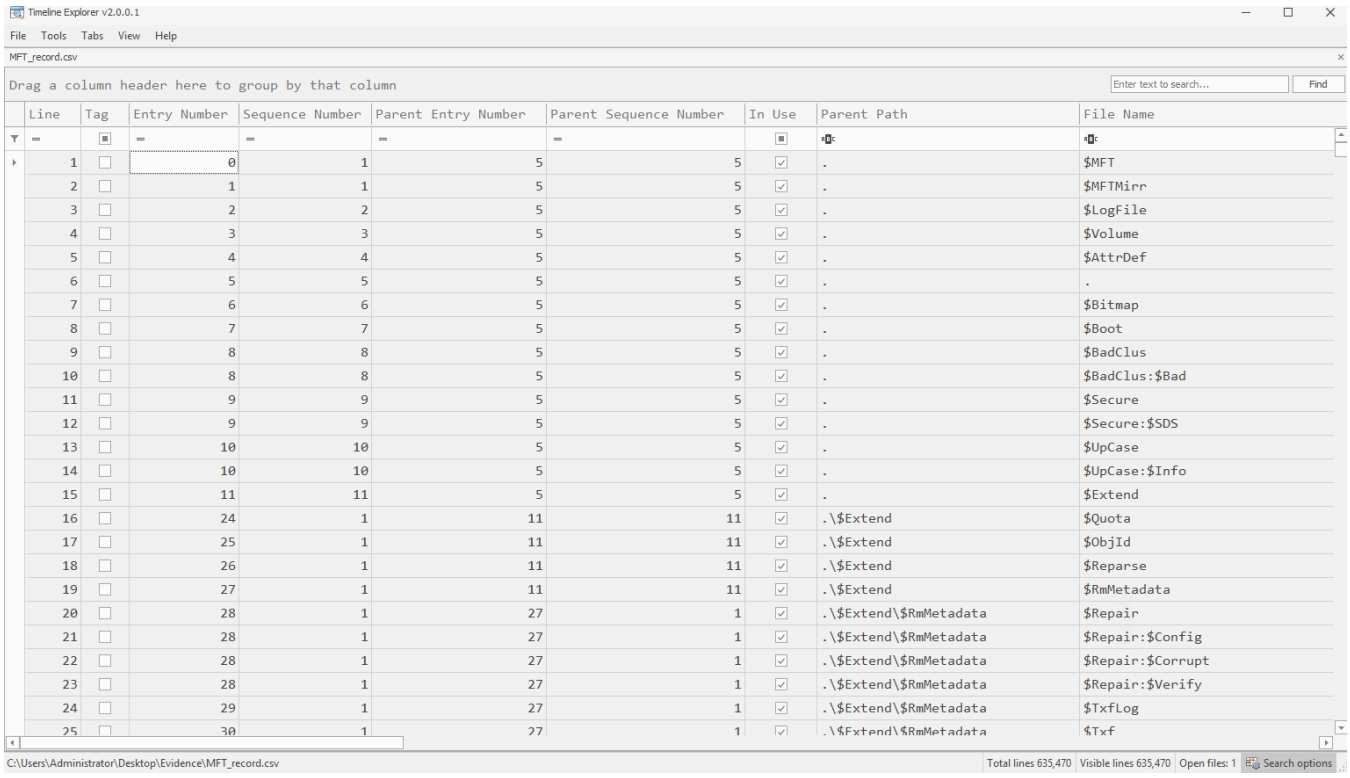
— **-f**: To provide the MFT File Record
— **— csv**: Directory to save the CSV formatted result to.
— **— csvf**: File name to save CSV formatted result to.
Let's use the flags mentioned above to extract the records from the MFT File present in the **~\Desktop\Evidence** and save it in the same folder.

Command: **MFTECmd.exe -f ..\Evidence\$MFT — csv ..\Evidence — csvf ..\Evidence\MFT_record.csv**



This image and all the theoretical content of the present article is TryHackMe´s property.

If we look at the output, we can see that the tool has correctly identified the file type as MFT. Also, the output is saved in the Evidence folder as MFT_record.csv. We can click on the output file, and it will automatically load the file into another useful tool called Timeline Explorer, as shown below:



This image and all the theoretical content of the present article is TryHackMe´s property.

Important Columns in the MFT Record

If we examine the output, we can see that various columns contain important information. Let's break down these columns to better understand the output.

Entry Number
— Description: A unique identifier for the MFT record.
— Importance: Helps map the record to its specific file or directory and reuse track records.

Parent Entry Number
— **Description**: The entry number of the parent directory.
— **Importance**: Shows the file or directory location in the file system hierarchy, aiding in reconstructing directory structures.

Sequence Number
— **Description**: A counter incremented when the MFT record is reused.
— **Importance**: Helps identify reused records and differentiate between old and new files occupying the same entry.

File Name
— **Description**: The name of the file or directory.
— **Importance**: Essential for identifying the file and matching it with user activities or processes.

Timestamps
— **Creation Time**: When the file or directory was created.
— **Modification Time**: When the content of the file was last modified.
— **Access Time**: When the file was last accessed.
— **MFT record Modification Time**: When the metadata was last updated.
— **Importance**: Crucial for timeline analysis, helping to reconstruct the s — equence of events involving the file.

Flags
— **Description**: Indicates whether the record represents a file, directory, or unused record.
— **Importance**: Determines the type of object and its activity status (e.g., deleted or active).

Entry Flags
— **Description:** Flags like Read-only, Hidden, System, etc.
— **Importance**: Useful for identifying hidden or system files often used by malware.

In Use
— **Description:** Indicates whether the current record is associated with the active file or directory.
— **Importance:** It can indicate which files are present on the disk and which are deleted or no longer in use. MFT File still maintains the record even after the deletion.

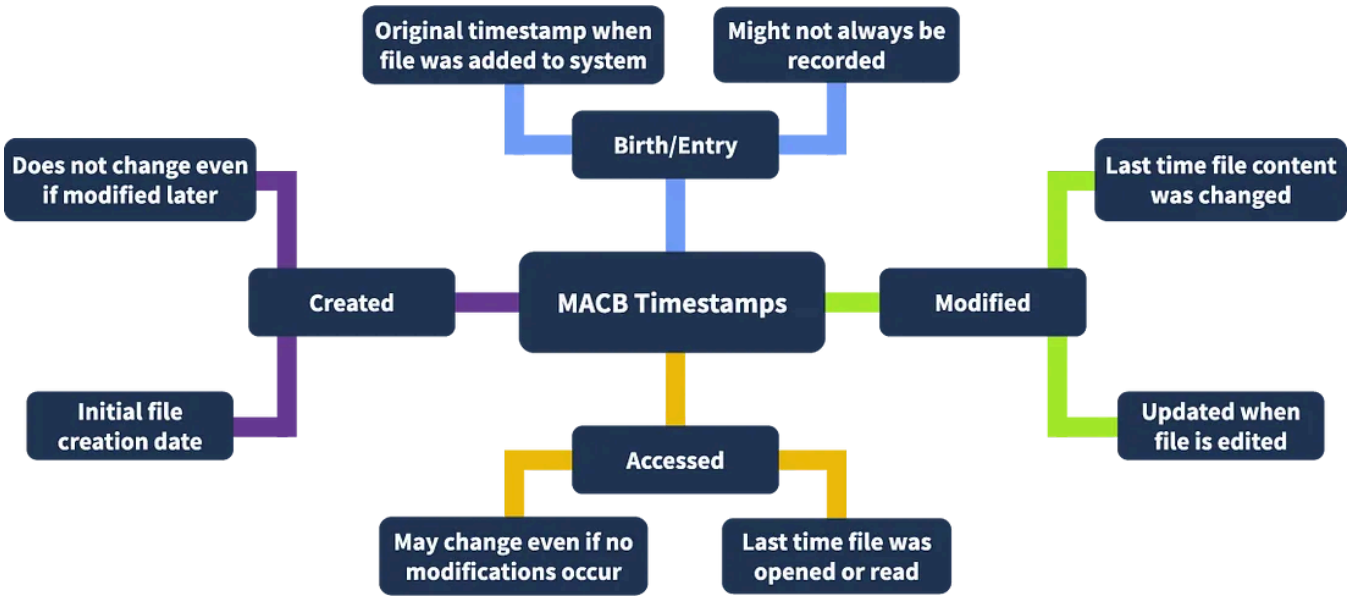
Logical Size
— **Description**: The size of the file's actual data.
— **Importance**: Provides insights into the file's content size, useful for verifying anomalies (e.g., files marked empty but containing data).

Physical Size
— **Description**: The amount of disk space allocated to the file.
— **Importance**: Helps identify slack space or fragmented files.

Now that we understand what these columns mean and what key values they provide, explore the MFT record and see if you can find footprints of any suspicious file or the tool.

MACB Time

In the records, we see different timestamps associated with each record. What are they, and how are they different from each other? Well, they are classified as MACB time. **MACB refers to the timestamps associated with file system metadata, which provide crucial information about a file's history**. These timestamps are:
— **M — Modified:** The time when the file's content was last changed
— **A — Accessed:** The time when the file was last read or opened.
— **C — Changed** (Metadata): The time when the file's metadata (e.g., permissions, ownership) was last altered.
— **B — Birth** (Creation): The time when the file was first created (available in NTFS and other advanced file systems).



This image and all the theoretical content of the present article is TryHackMe´s property.

Why is the MACB Timestamp Important?

Some of the key points to understand the importance of MACB timestamp are:

- By analyzing MACB times, we can reconstruct events, such as when a file was created, accessed, modified, or when its metadata changed. This is critical in building a timeline of user or attacker activities.
- Timestamps can reveal unusual behavior, such as files modified after a system compromise or files accessed during odd hours.
- Investigators can match timestamps to log entries, user activity, or external events to identify patterns or an attacker's presence.
- Attackers often modify timestamps to cover their tracks. One such attack is classified as a timestomping attack. We can examine the timestamp to identify the manipulation of the timestamps of files or directories.

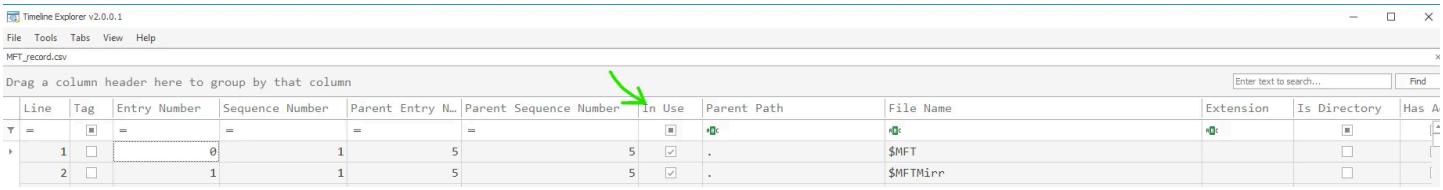
Answer the questions below

5.1. Which column indicates that the file is no longer present on the disk?

`In Use`

- **In Use**
  - **Description**: Indicates whether the current record is associated with the active file or directory.
  - **Importance**: It can indicate which files are present on the disk and which are deleted or no longer in use. MFT File still maintains the record even after the deletion.

This image and all the theoretical content of the present article is TryHackMe´s property.

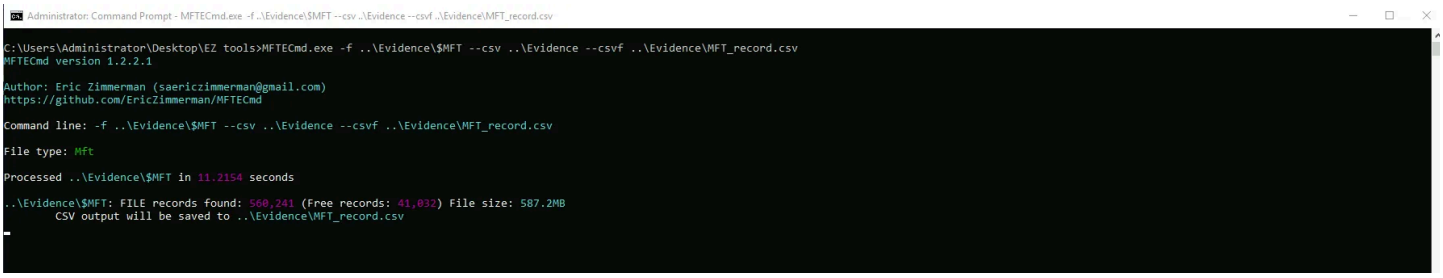


Timeline Explorer

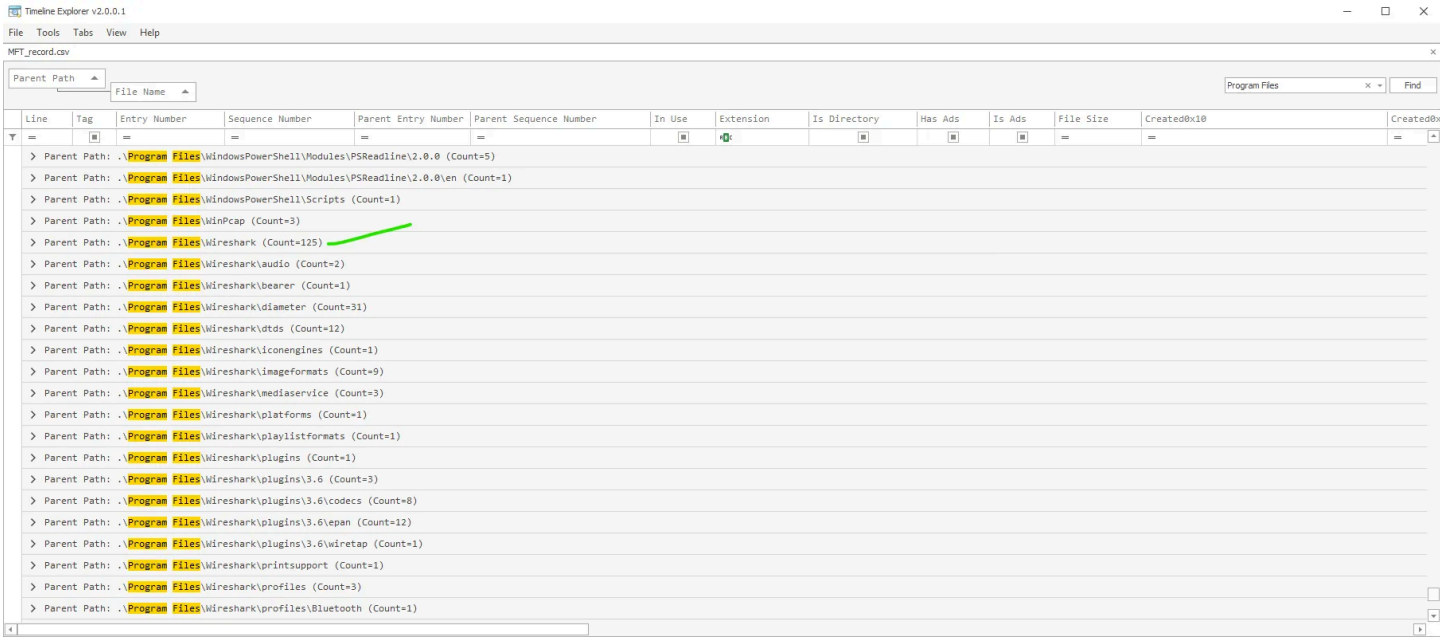
5.2. Examine the MFT record; what is the network sniffer installed on this system in the \Program Files\ directory? *Hint* : *Apply filter on the given path in the Parent Path Column.*

`wireshark`

— Ran the provided command line in this task, and a `MFT_record.csv` file was generated in the `Evidence` folder.



— Launched `Timeline Explorer`.
— Clicked `File` , than `Open`.
— Selected `MFT_record.csv` file, and clicked `Open`.
— Typed `Program Files` , and clicked `Find`.
— Dragged the column header `Parent Path` , than the column header `File Name` to group.

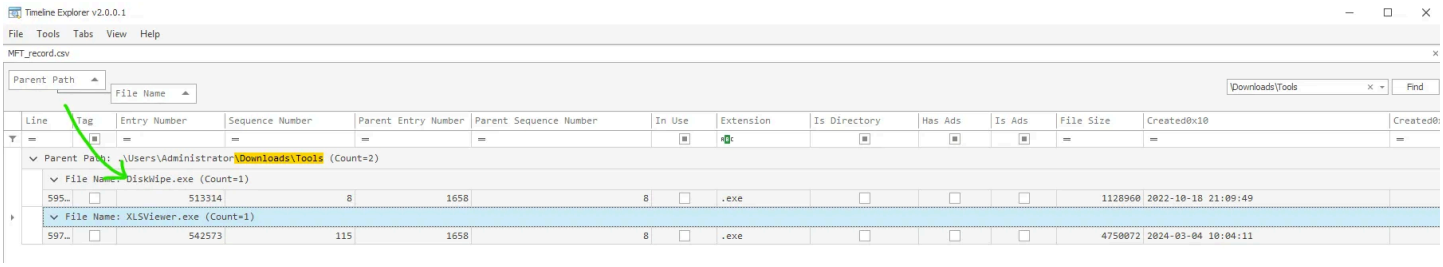


Timeline Explorer

5.3. An anti-forensics tool responsible for wiping out an attacker's traces was installed in the \Downloads\Tools folder. What is the name of the tool? *Hint* : *Apply filter on the given path in the Parent Path Column.*

`DiskWipe.exe`

— Typed `\Downloads\Tools` , and clicked `Find` .

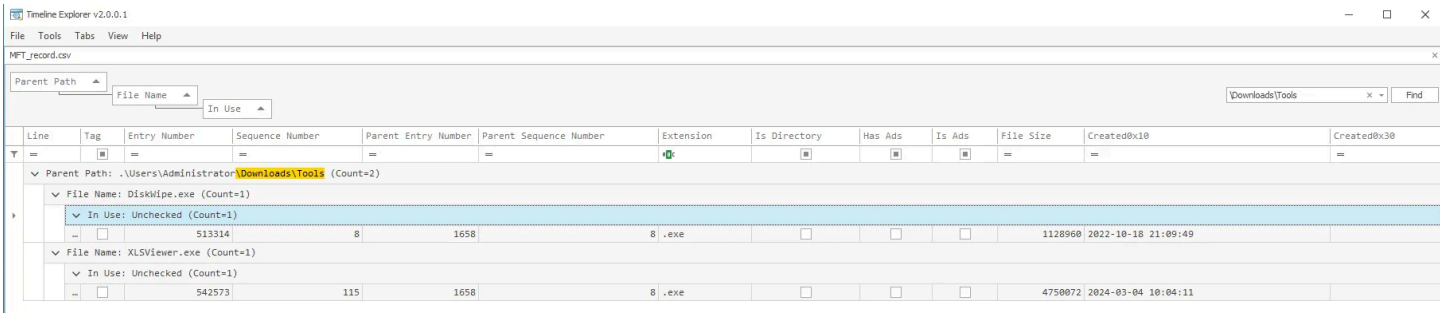


Timeline Explorer

5.4. According to the MFT record, is the anti-forensics tool currently present on the disk? (yay or nay)

`nay`

— It is `nay` because the `In Use` field is unchecked.

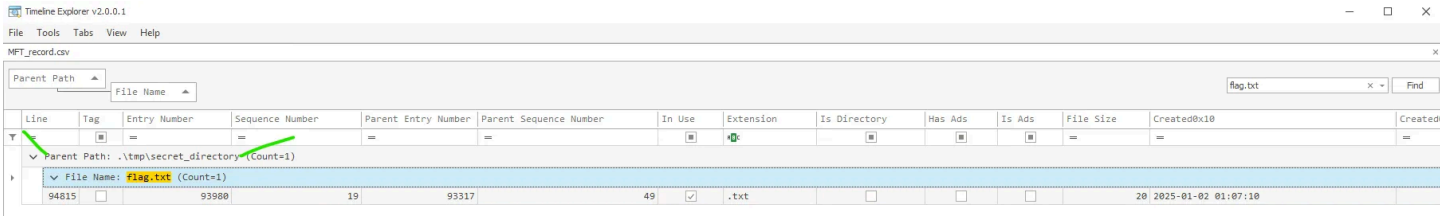


Timeline Explorer

5.5. Examining the MFT record, it seems there is a record of a flag.txt file. What is the parent path of the file?

`.\tmp\secret_directory`

— Typed `flag.txt` , and clicked `Find` .

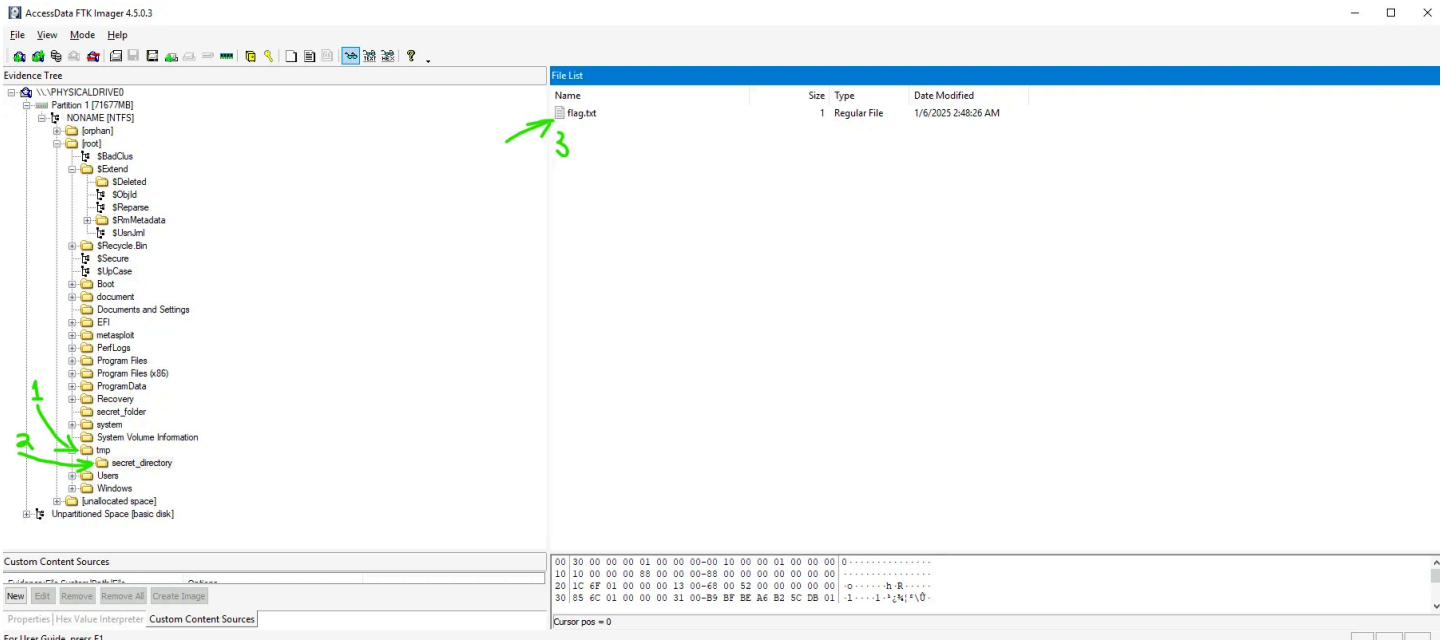


Timeline Explorer

5.6. What is the content of the **flag.txt** file?

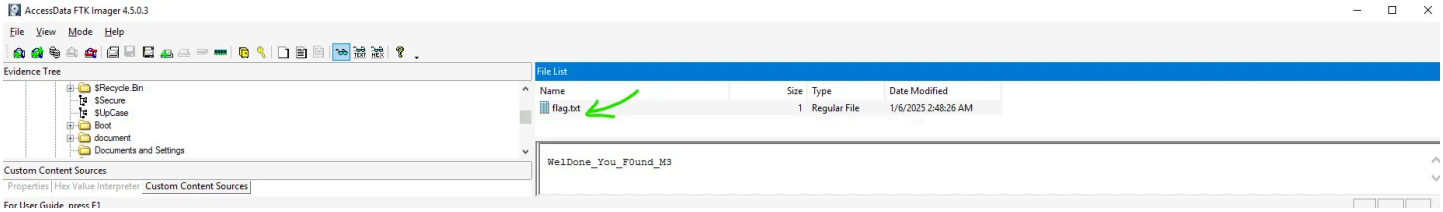
`WelDone_You_F0und_M3`

— Used `FTK IMager` to expand `tmp` folder, than `secret_directory` .



FTK Imager

— Clicked over `flag.txt`.

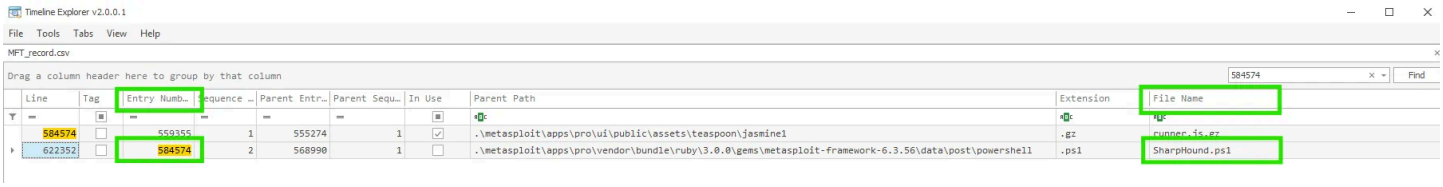


FTK Imager

5.7. What is the file name associated with the MFT entry number “584574”?

`SharpHound.ps1`

— Typed `584574`, and clicked `Find`.



Timeline Explorer

Task 6 . NFTS Journaling

NTFS includes a very important feature called file system journaling. This technology allows the OS to maintain a transactional record of all changes made to a volume, so in the event of a crash or power failure, the system can roll back the changes or continue where it left off.

The goal is to maintain file system integrity and to prevent catastrophic events from occurring. Some of the key features of NTFS Journaling are:

- **File system Integrity**: NTFS journaling ensures that the file system can recover even after crashes, minimizing data loss and corruption risks. This is essential for both performance and reliability.
- **Deleted Files**: The ability to track when files were deleted through the USN journal allows forensic experts to establish timelines critical in investigations.
- **USN vs. Log File**: Understanding the difference between the USN journal (file changes) and the log file (MFT metadata changes) is vital for accurate forensic analysis, as they provide different insights.
- **Historical Data Access**: Volume shadow copies provide a way to retrieve older versions of journals, allowing forensic teams to analyze data from different points in time, which might reveal crucial evidence.

Types Of NTFS Journals

There are two journals in the NTFS which we will explore:

$LogFile

[…]

Universal Sequence Number (USN) Journal ($USNJrnl)

[…]

Structure of USNJrnl

[…]

Extract the $J File

[…]

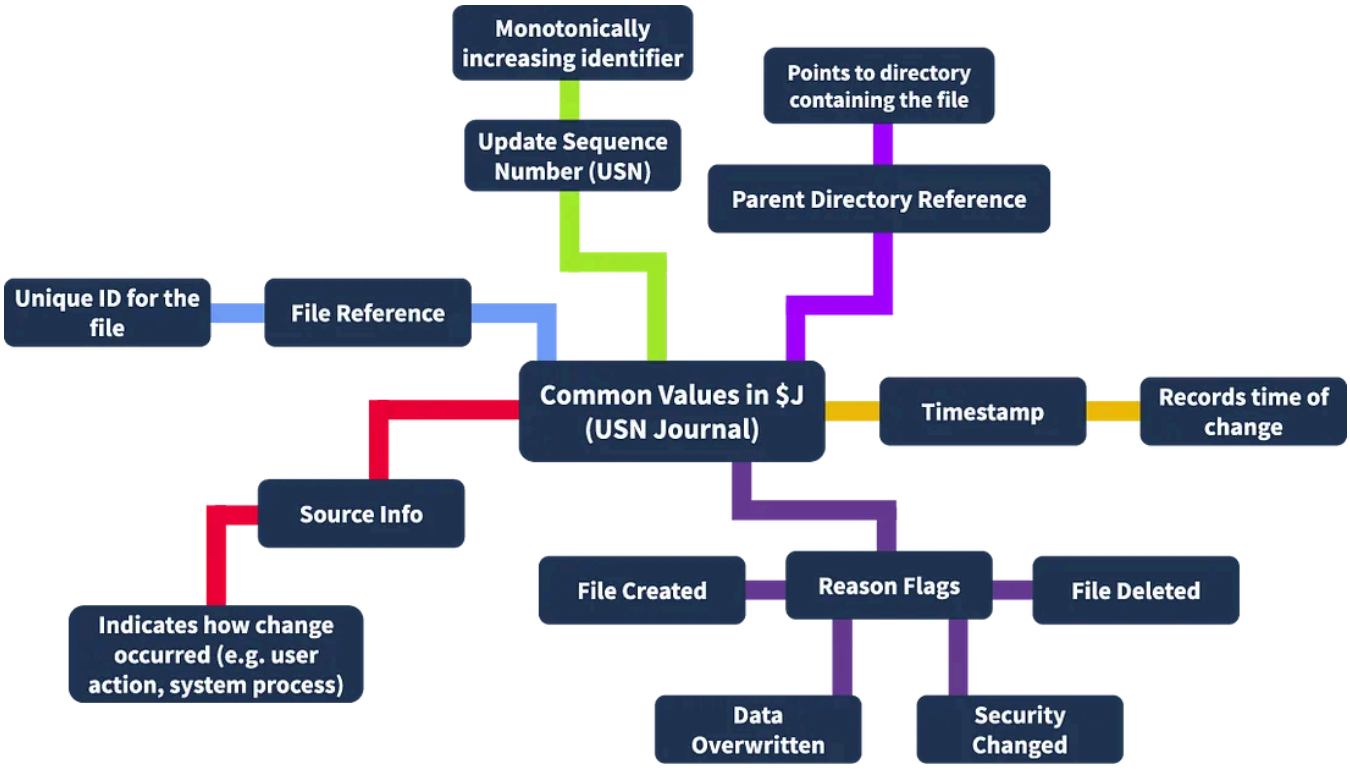
Examining the Updated Reason

If we click on the Update Reason column, we can see various reasons, indicating the type of the file system change. Some of the common Opcodes associated with the file changes are explained below:

Opcode	Description
USN_REASON_DATA_OVERWRITE	File or directory data was overwritten.
USN_REASON_DATA_EXTEND	File or directory data was extended (e.g., file size increased).
USN_REASON_DATA_TRUNCATION	File or directory data was truncated (e.g., file size decreased).
USN_REASON_NAMED_DATA_OVERWRITE	The alternate data stream.
USN_REASON_NAMED_DATA_EXTEND	An alternate data stream was extended.
USN_REASON_FILE_CREATE	A new file or directory was created.
USN_REASON_FILE_DELETE	A file or directory was deleted.
USN_REASON_RENAME_OLD_NAME	The file or directory was renamed (old name recorded).
USN_REASON_CLOSE	The file or directory handle was closed after changes.

This image and all the theoretical content of the present article is TryHackMe´s property.

We can also filter on a certain value to display results related to that particular value.



This image and all the theoretical content of the present article is TryHackMe´s property.

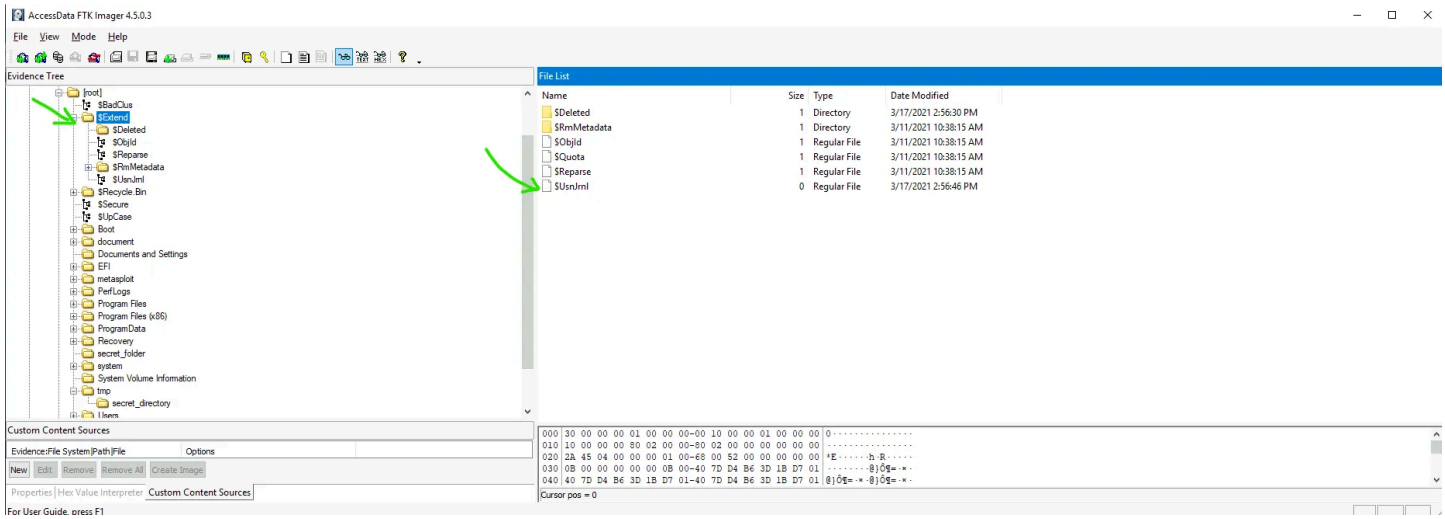
Now that we have extracted and parsed the evidence file, explore the output and see if you can find anything suspicious.

Answer the questions below

6.1. What is the text file name associated with **entry number 95071 before renaming it**?

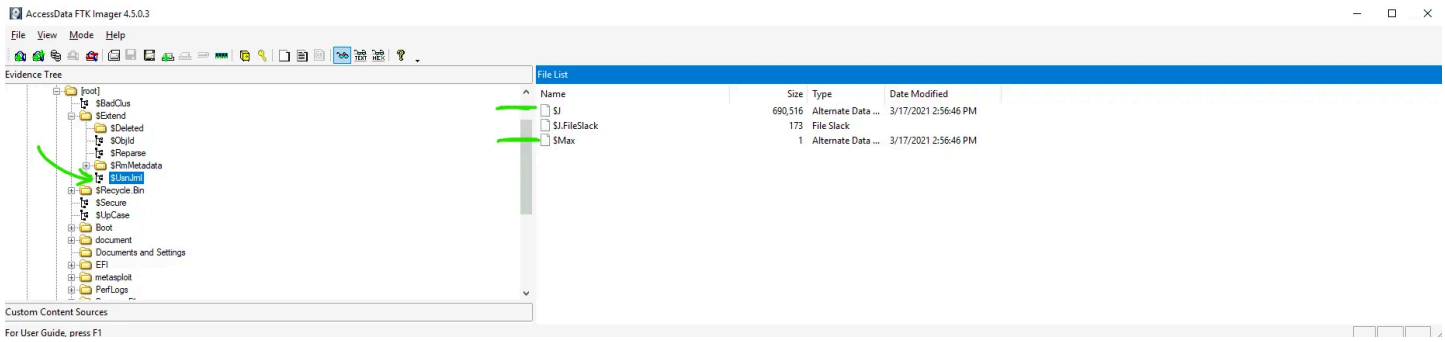
`New Text Document.txt`

— Expanded `$Extend`, and discovered the location of `$UsnJrnl`.



FTK Imager

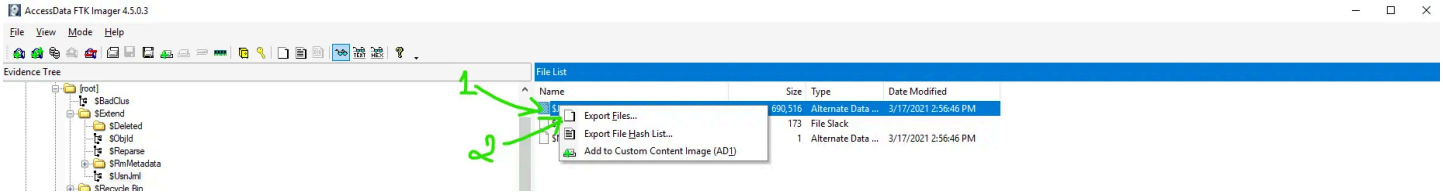
— Double-clicked `$UsnJrnl`, and discovered its components: `$J` , and `$Max`.



FTK Imager

— Right-clicked `$J`.

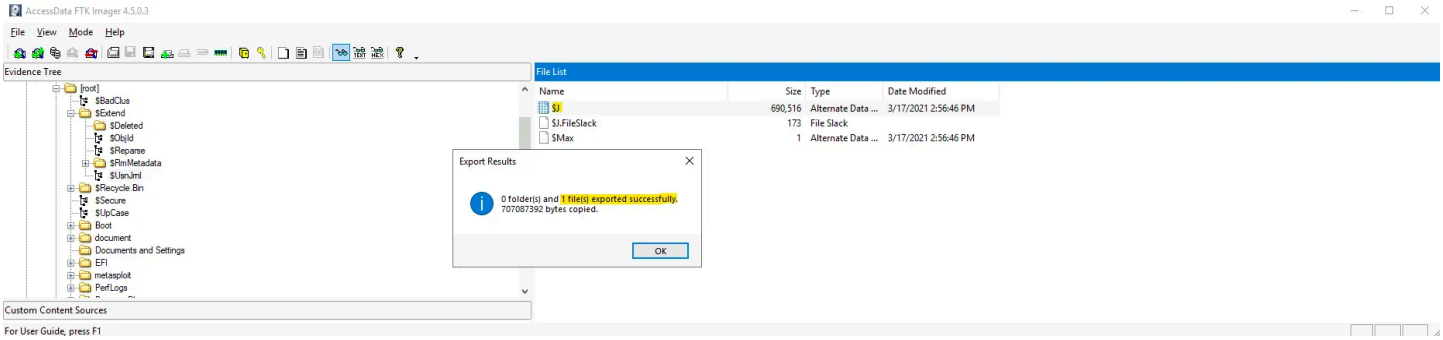
— Clicked `Export Files ...`.



FTK Imager

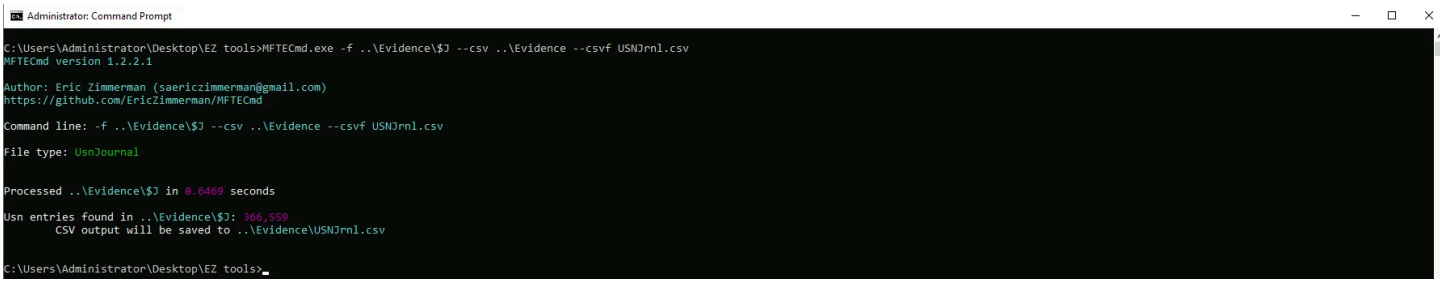
— Selected `C:\Users\Administrator\Desktop\Evidence\$J`.

— Clicked `OK`.



FTK IMager

— Ran the provided command line in this task, and a `USNJrnl.csv` file was generated in the `Evidence` folder.

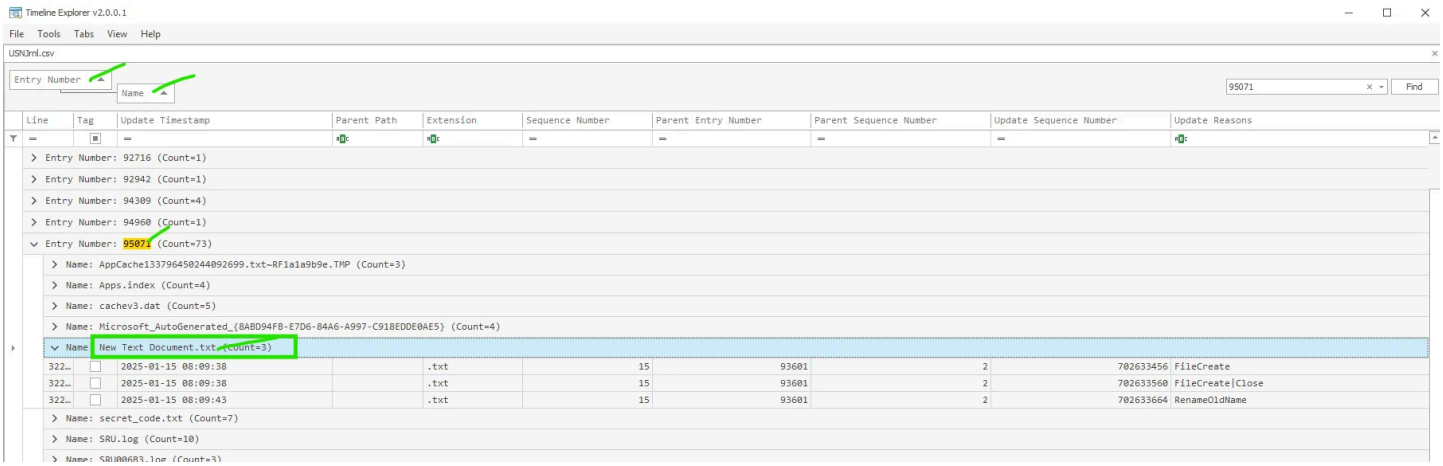


Command Prompt

— Opened `USNJrnl.csv` using `Timeline Explorer`.

— Typed `Program Files`, and clicked `Find`.

— Dragged the `Entry Number` column header, than the `Name` column header to group.

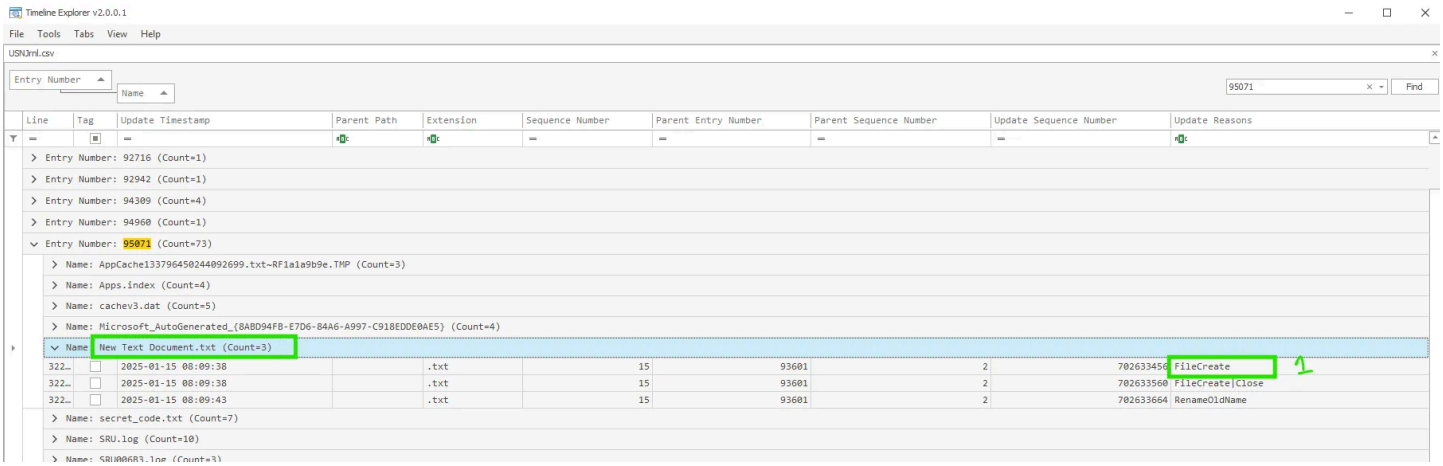


Timeline Explorer

6.2. According to the record, what is the first operation performed on the file in the question above?

`FileCreate`

— Discovered the answer based on **6.1.**



Timeline Explorer

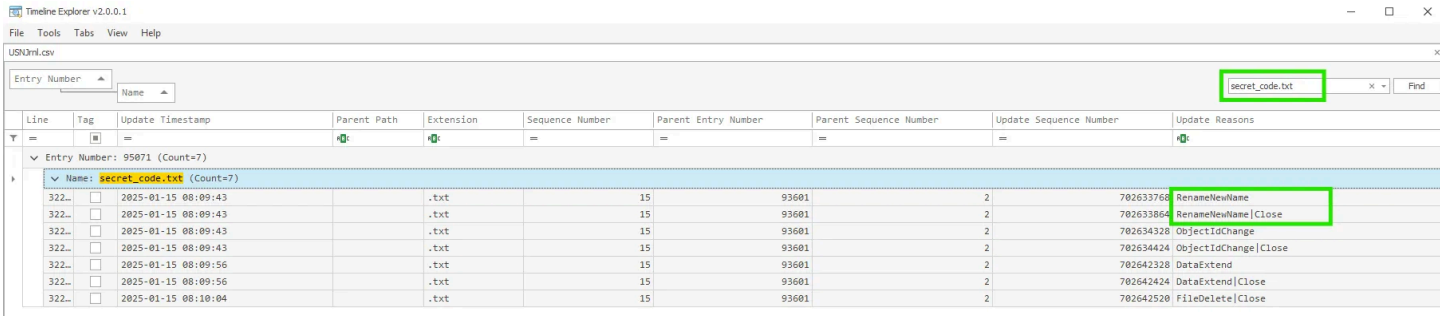
6.3. According to the record in $J, what is the count of the rename operation found against secret_code.txt? *Hint* : *Apply Filter on the given File and examine the Update Reasons.*

2

— Typed `secret_code.txt` , and clicked `Find` .

Timeline Explorer

— Clicked expanding `secret_code.txt` ´s results, and analyzed the `Update Reasons`.

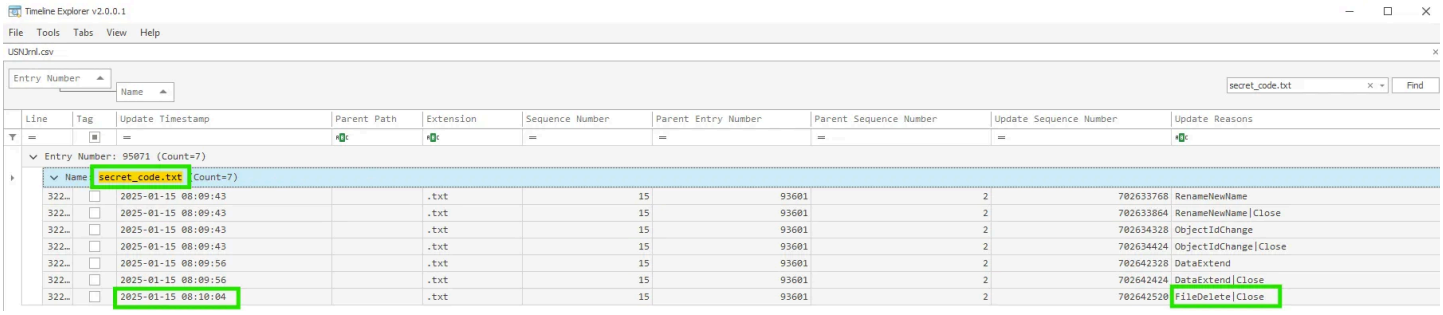


Timeline Explorer

6.4. According to the record, when was the **secret_code.txt file deleted**?

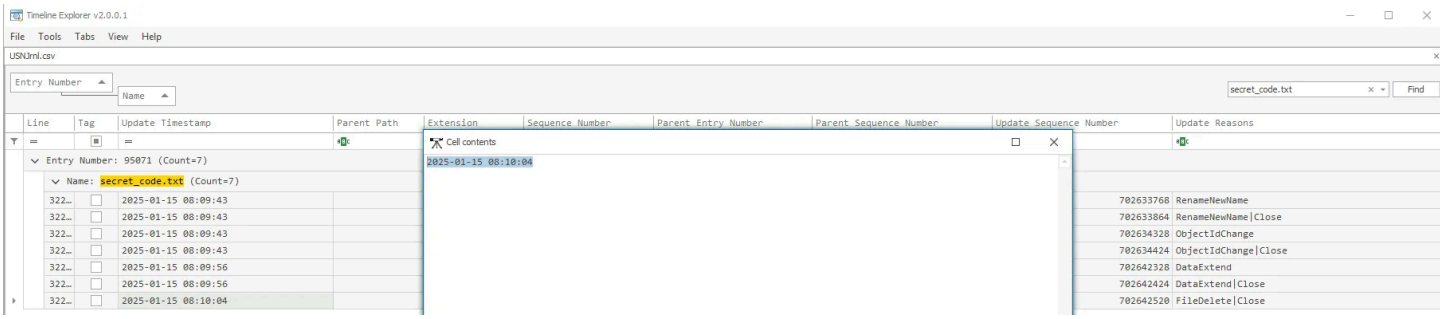
`2025-01-15 08:10:04`

— Discovered the answer based on **6.3.**



Timeline Explorer

— Doubled-cliked over `Update Timestamp` to copy it.



Timeline Explorer

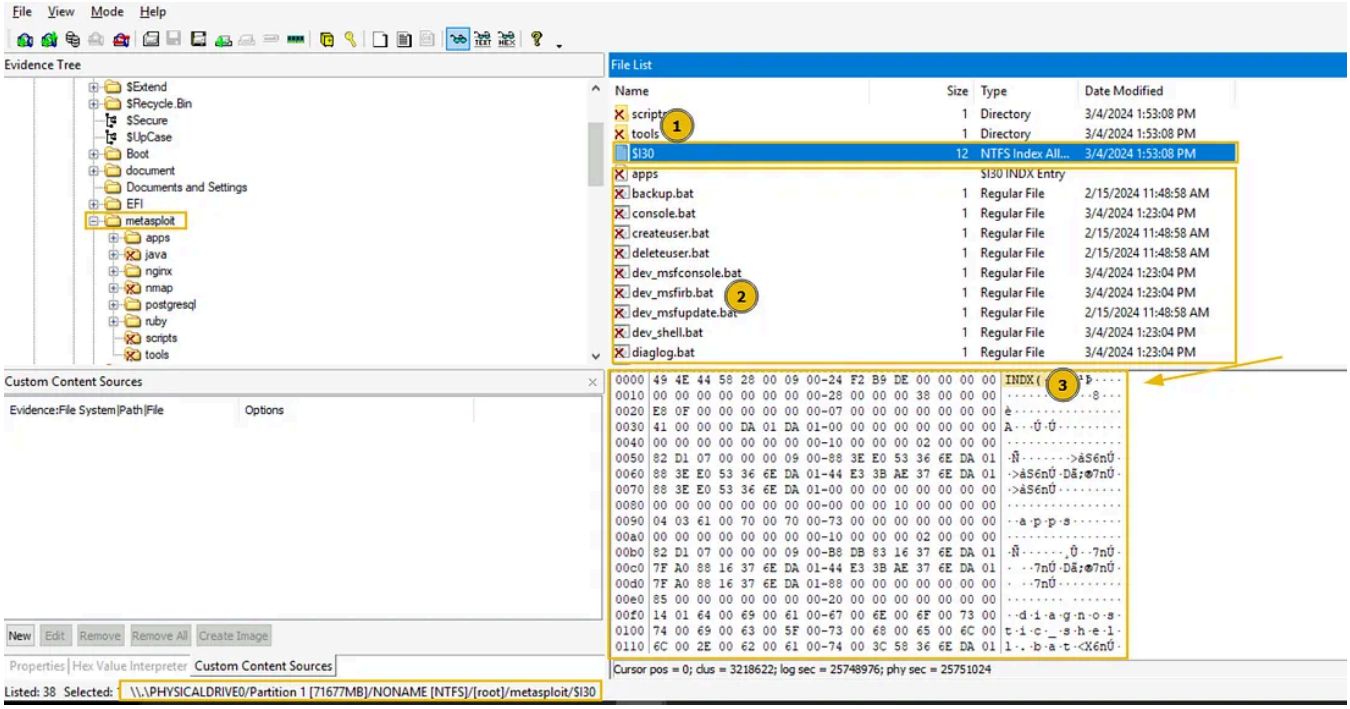
Task 7 . Index Allocation Attribute ($I30) Overview

The $I30 file is an NTFS index attribute, which can also be classified as the directory index that maintains a structured layout of files and directories within an NTFS volume. The Windows OS uses it to store and retrieve information about directory entries. As we examined the directories in FTK Imager, we observed that $I30 is present in some directories containing information about the files and folders within the directory.

Simply put, it is mainly used to store a directory's index entries and contains metadata about the files and subdirectories within it. This metadata includes filenames, timestamps, and other details about files, even those that have been deleted or modified.

Index Allocation Attribute ($I30)

Before explaining what the $I30 file contains, let's explore the [ROOT]/metasploit folder within the FTK Imager. We will find the $I30 file in the directory, classified as NTFS Index Allocation, as shown below:



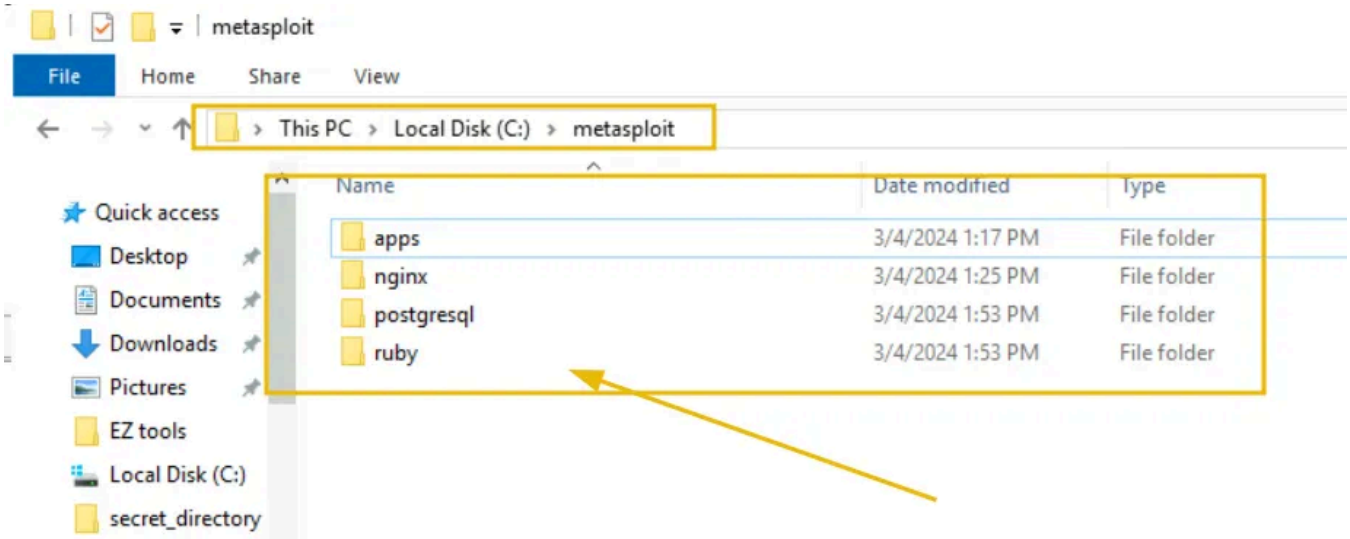
This image and all the theoretical content of the present article is TryHackMe´s property.

Here, we found the $I30 file, along with a few files marked with a red cross, indicating that they are no longer present, which is called the Slack Space.

Slack Space

Slack space in $I30 refers to files that are deleted, renamed, or moved within a directory. Even though a file entry is removed from the active index, residual data may still be present in the slack space, which may be helpful for forensics investigations when looking for deleted files.

To verify the content of this directory, we will navigate to the C:\metasploit directory, which shows only four directories, as shown below:



This image and all the theoretical content of the present article is TryHackMe´s property.

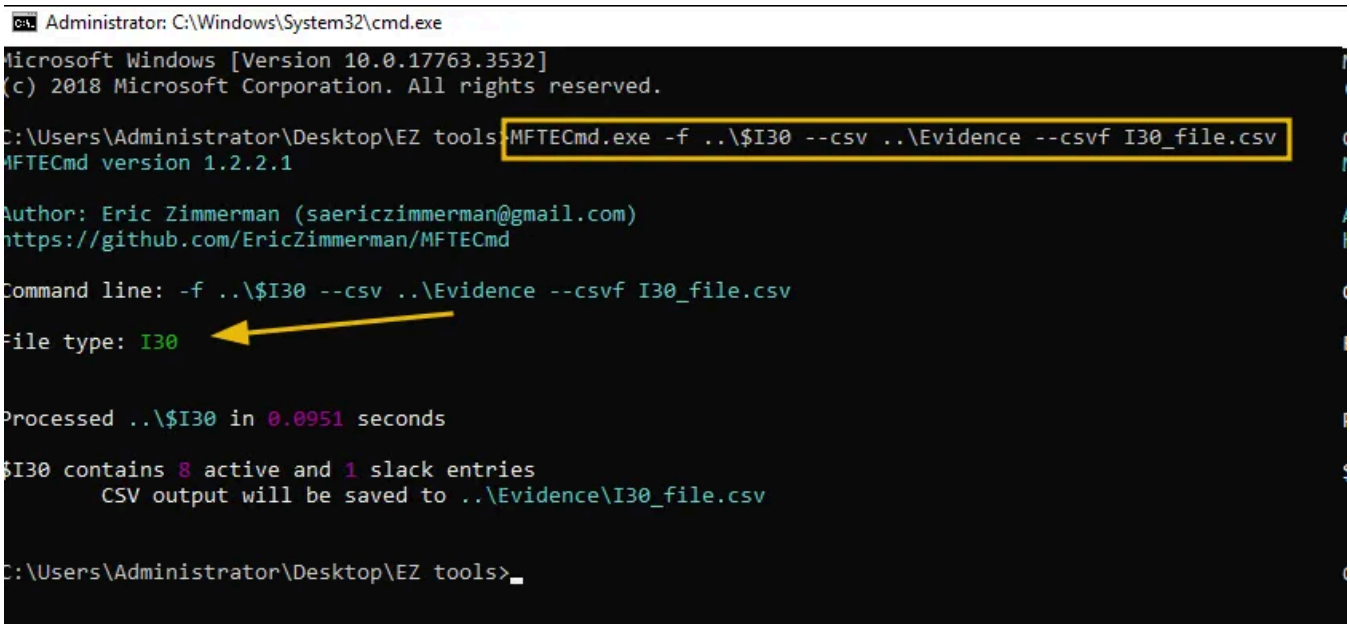
Forensics Value

Now that we understand the $I30 file and its purpose, it's important to understand its forensics value. Some of the key forensics values we can get from this are:
— **Deleted Files:** When a file is created, its entry with attributes is stored in the $I30. However, when the files are deleted, their metadata might still be preset in the $I30, which could help track deleting files until they are overwritten.
— **Renamed / Moved File:** Actions like renaming or moving files are also reflected in the $I30, which helps track file movement.
— **Uncover Hidden Files:** File Attributes can help identify the hidden files.

Analyzing $I30

Let's export this $I30 attribute file from the /[root]/metasploit directory using FTK Imager for the analysis. Once the $I30 index file is exported, we will use the following command to extract the information contained within the $I30 file, as shown below:

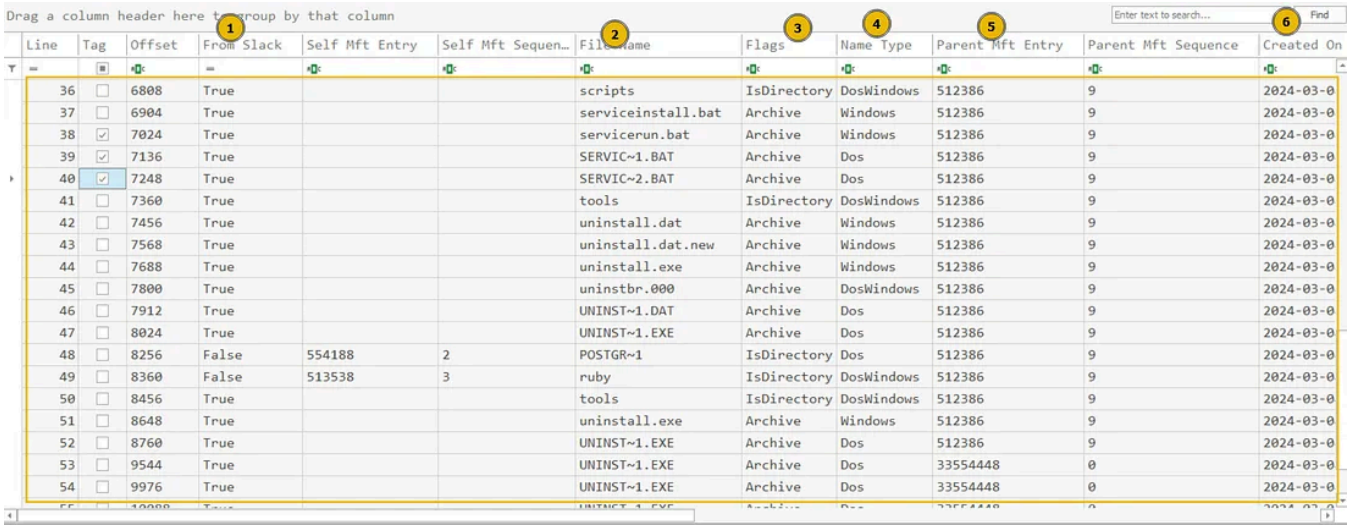
Command: **MFTECmd.exe -f ..\Evidence\$I30 — csv ..\Evidence\ — csvf i30.csv**



This image and all the theoretical content of the present article is TryHackMe´s property.

This command will parse and extract information from the $I30 attribute file.

Open the output file in the Timeline Explorer, as shown below:



Drag a column header here to group by that column

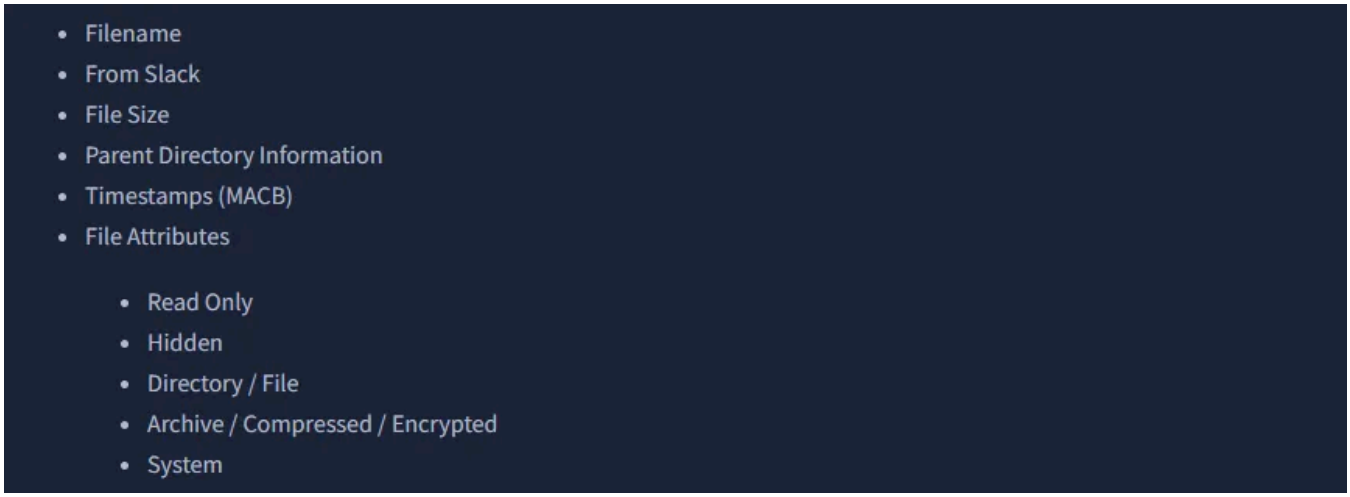
Enter text to search... Find

Line	Tag	Offset	From Slack	Self Mft Entry	Self Mft Sequen…	File Name	Flags	Name Type	Parent Mft Entry	Parent Mft Sequence	Created On
36	☐	6808	True			scripts	IsDirectory	DosWindows	512386	9	2024-03-0
37	☐	6904	True			serviceinstall.bat	Archive	Windows	512386	9	2024-03-0
38	☑	7024	True			servicerun.bat	Archive	Windows	512386	9	2024-03-0
39	☑	7136	True			SERVIC~1.BAT	Archive	Dos	512386	9	2024-03-0
40	☑	7248	True			SERVIC~2.BAT	Archive	Dos	512386	9	2024-03-0
41	☐	7360	True			tools	IsDirectory	DosWindows	512386	9	2024-03-0
42	☐	7456	True			uninstall.dat	Archive	Windows	512386	9	2024-03-0
43	☐	7568	True			uninstall.dat.new	Archive	Windows	512386	9	2024-03-0
44	☐	7688	True			uninstall.exe	Archive	Windows	512386	9	2024-03-0
45	☐	7800	True			uninstbr.000	Archive	DosWindows	512386	9	2024-03-0
46	☐	7912	True			UNINST~1.DAT	Archive	Dos	512386	9	2024-03-0
47	☐	8024	True			UNINST~1.EXE	Archive	Dos	512386	9	2024-03-0
48	☐	8256	False	554188	2	POSTGR~1	IsDirectory	Dos	512386	9	2024-03-0
49	☐	8360	False	513538	3	ruby	IsDirectory	DosWindows	512386	9	2024-03-0
50	☐	8456	True			tools	IsDirectory	DosWindows	512386	9	2024-03-0
51	☐	8648	True			uninstall.exe	Archive	Windows	512386	9	2024-03-0
52	☐	8760	True			UNINST~1.EXE	Archive	Dos	512386	9	2024-03-0
53	☐	9544	True			UNINST~1.EXE	Archive	Dos	33554448	0	2024-03-0
54	☐	9976	True			UNINST~1.EXE	Archive	Dos	33554448	0	2024-03-0

This image and all the theoretical content of the present article is TryHackMe´s property.

File Attributes in the $I30

Some of the key information/metadata we can find within the $I30 about files and directories are listed below:



- Filename
- From Slack
- File Size
- Parent Directory Information
- Timestamps (MACB)
- File Attributes
  - Read Only
  - Hidden
  - Directory / File
  - Archive / Compressed / Encrypted
  - System

This image and all the theoretical content of the present article is TryHackMe´s property.

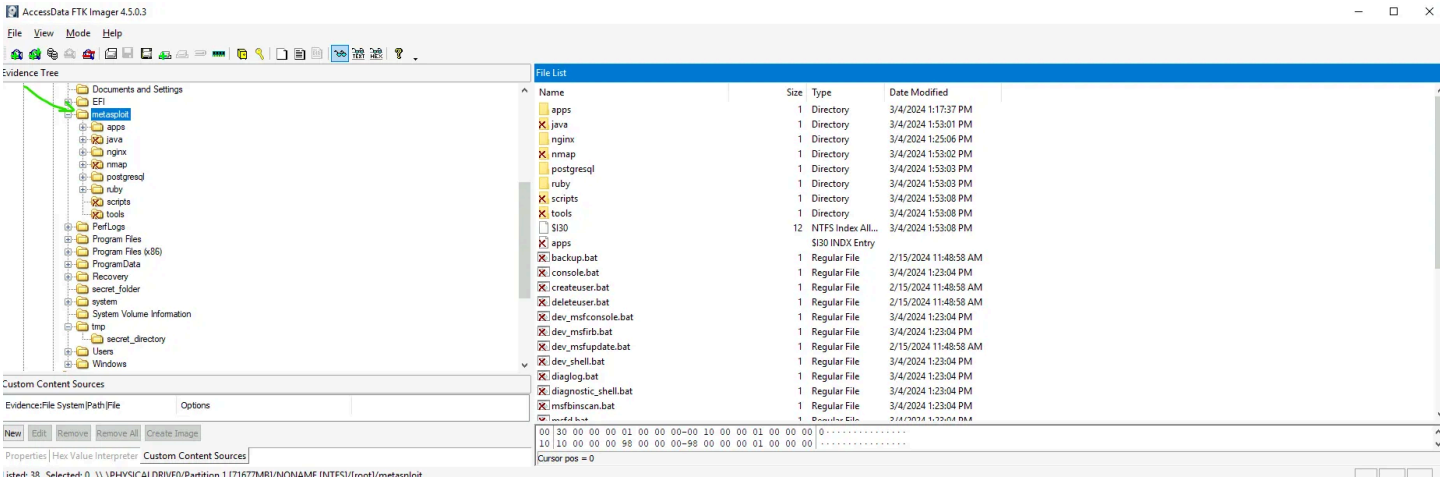
The output will show details about the files or directories present or present within the metasploit directory. We can apply a filter on the “From Slack” column to list down the files or directories that used to be present in this folder and are no longer present, hence being referred to in Slack Space.

Answer the questions below

7.1. How many deleted files or folders are present in the $I30 attribute file that was extracted in this task?

52

— Clicked `metasploit` folder which is part of `root` .

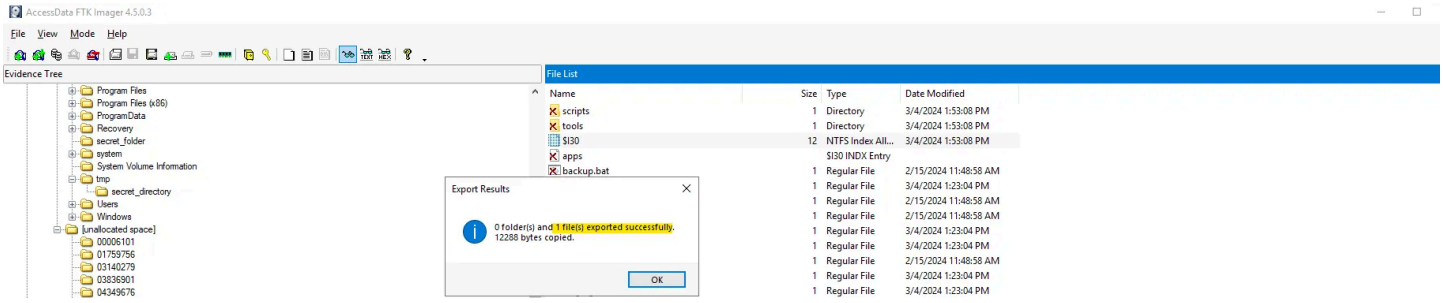


— Right-clicked `$I30` .

— Clicked `Export Files …` .

— Selected `Evidence` .

— Clicked `OK` .



— Ran the provided command line in this task, and an `outcome.csv` file was generated in the `Evidence` folder.

Command Prompt

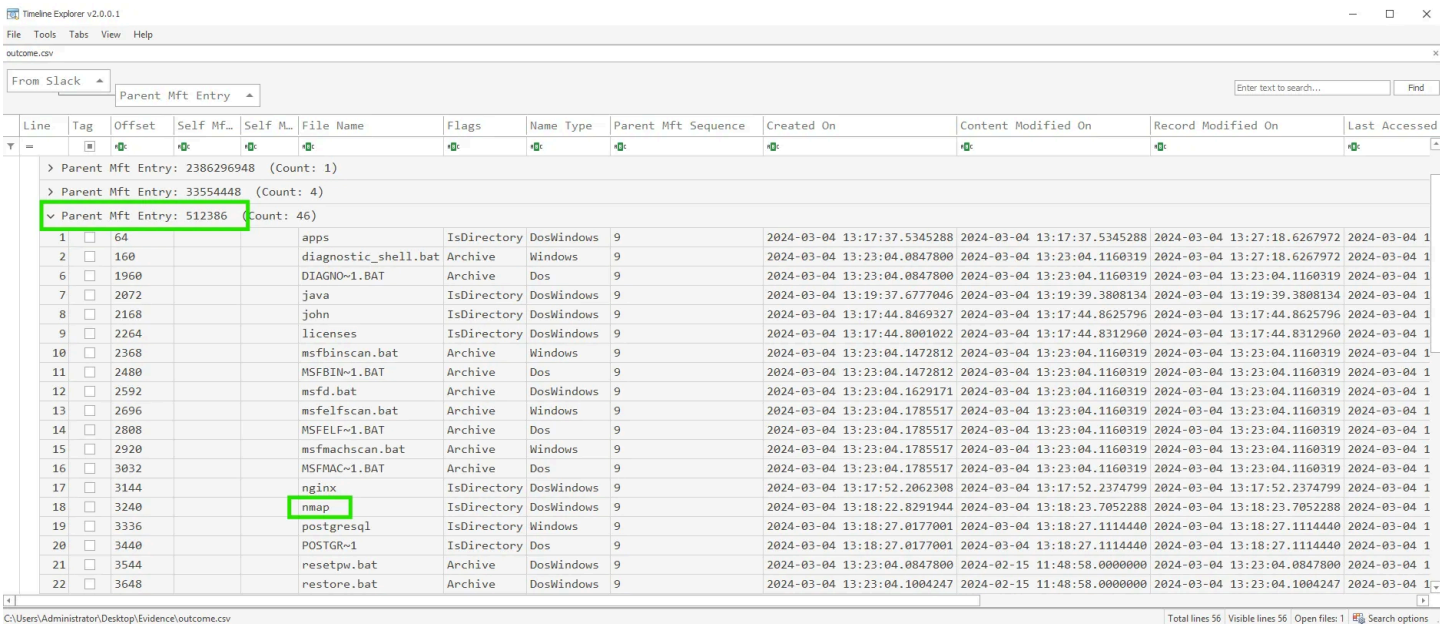
— Opened `outcome.csv` using `Timeline Explorer`.
— Dragged the `From Slack` column header to group.
— Dragged the `Entry Number` column header, than the `Name` column header to group.
— Confirmed the quantity of files deleted.

Timeline Explorer

7.2. What is the **parent MFT** entry of the **nmap** directory?

`512386`

— Dragged the `Parent MfT Entry` column header to group.
— Expanded the groups.



Timeline Explorer

Task 8. Conclusion

That's it for the room.

In this room, we dived deep into the details of the NTFS and how it is structured.
Some of the key points we learned are:
— How the NTFS is structured.
— Examine key components of the NTFS.
— Analyzed MFT table to explore information about the files on the disk.
— Learned about key journaling files: USN and LogFile
— Understand the importance of timestamps.

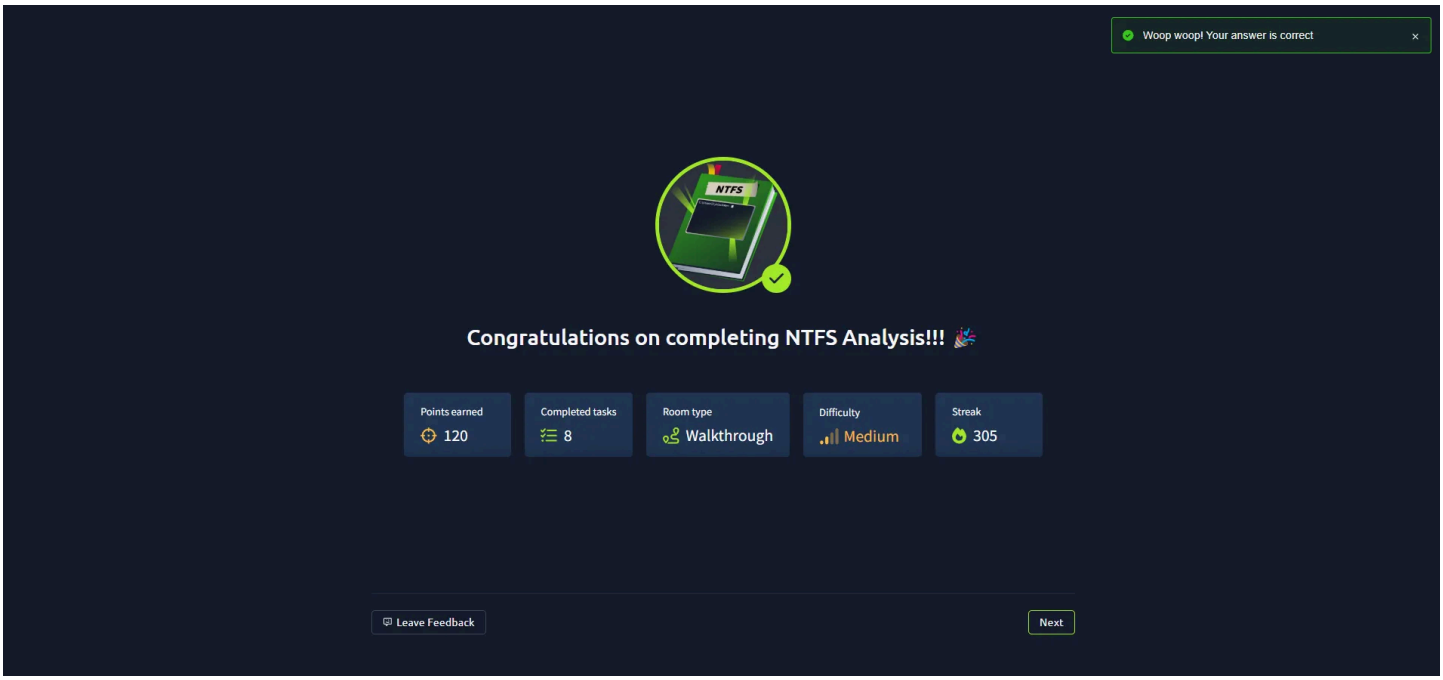
More walkthroughs and challenge rooms on File system Analysis are in the pipeline. Keep an eye on the socials for more updates.

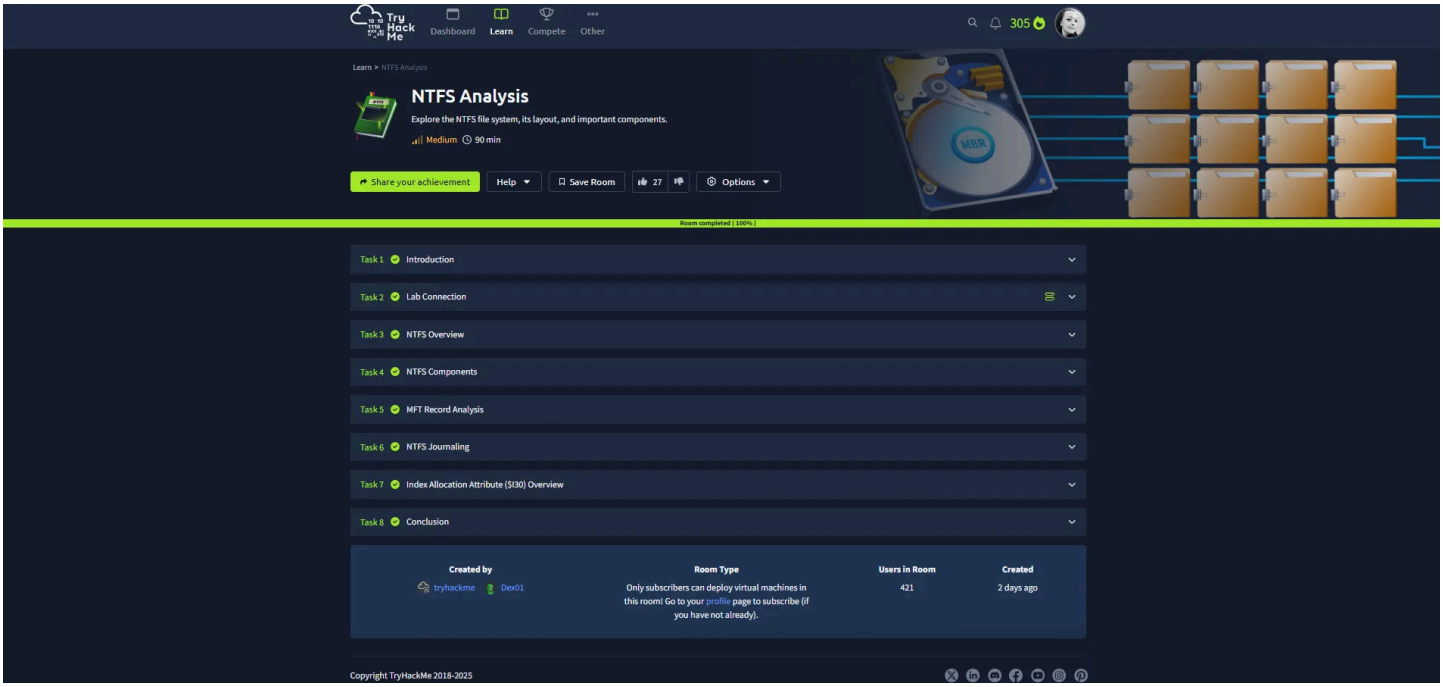
Answer the question below

8.1. Continue to complete the room.

```
No answer needed
```

Room Completed





This image and all the theoretical content of the present article is TryHackMe´s property.

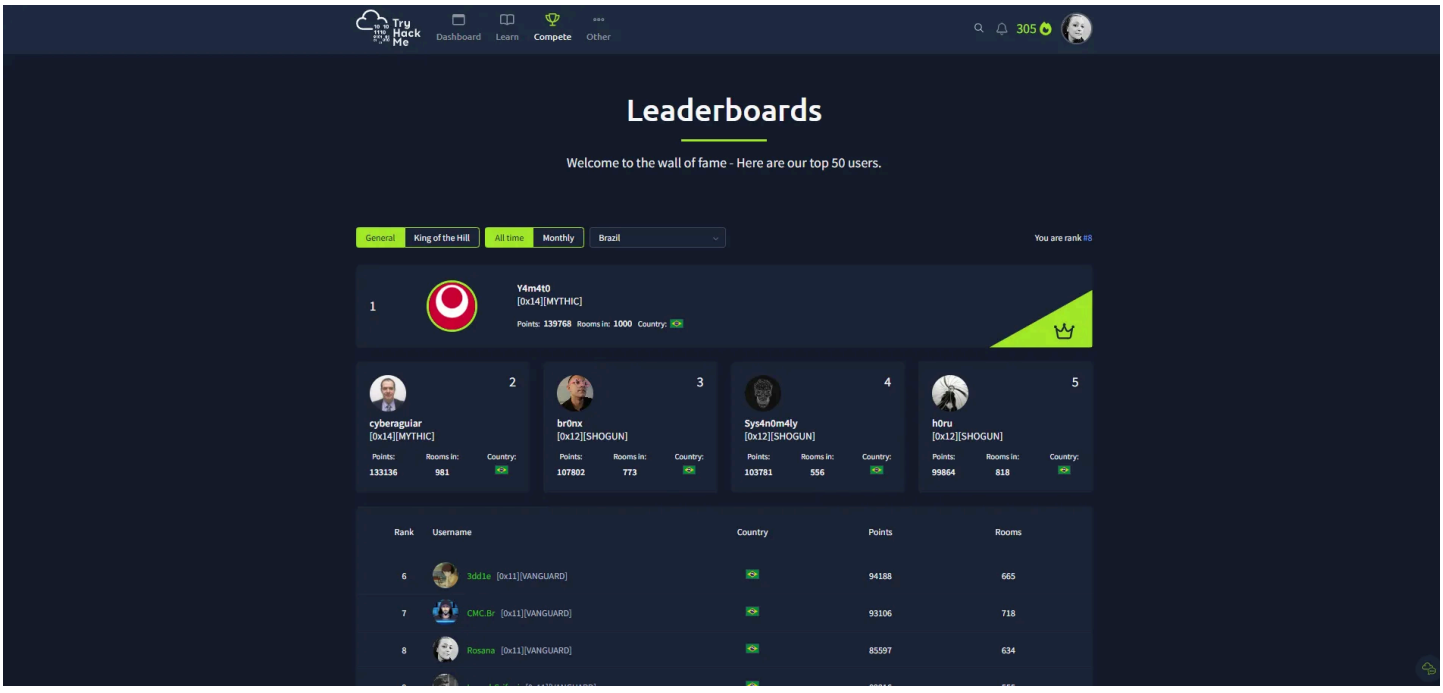
My journey on TryHackMe

```
305 days streak 🎉 ▪ 85,597 points ▪ 606 rooms completed ▪ 59 badges 🎖️
Global ranking:    368th all time    ▪    414th monthly
Brazil ranking:      8th all time    ▪      7th monthly
```

Global all time ranking: 368th

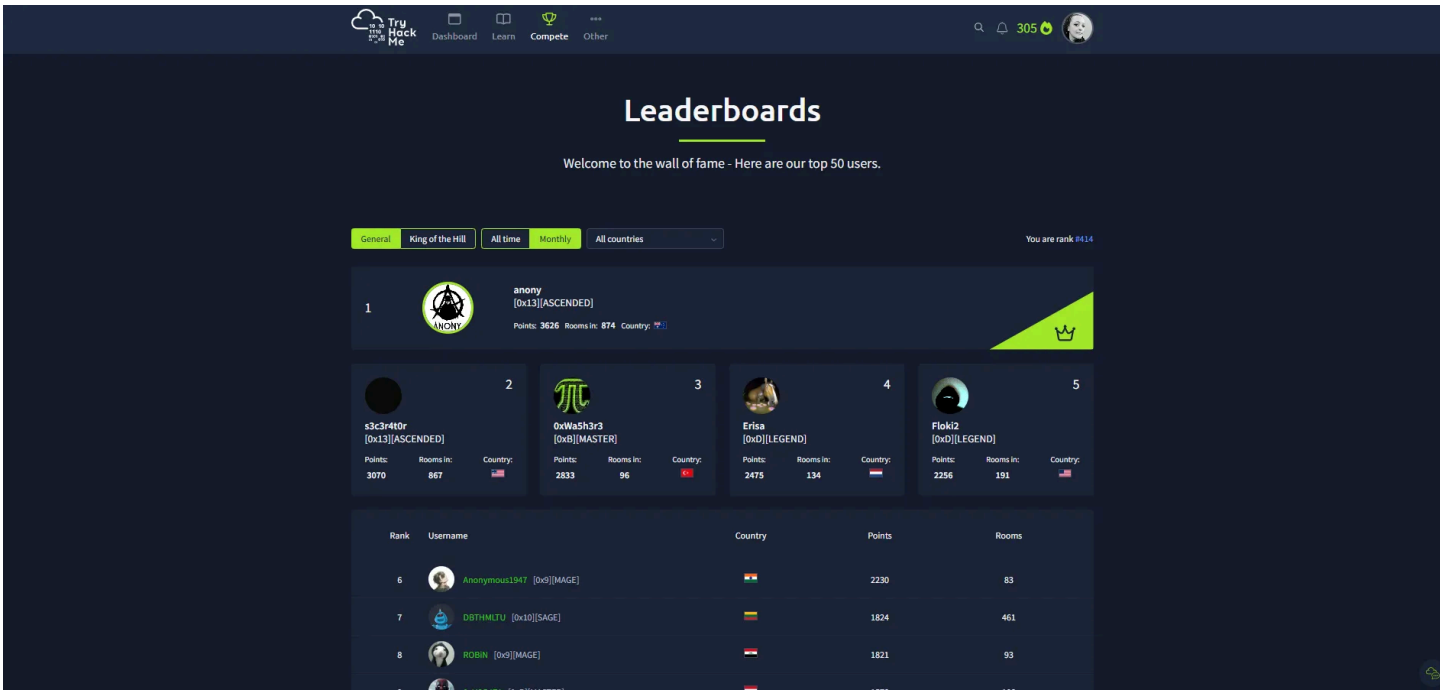
This image and all the theoretical content of the present article is TryHackMe´s property.

Brazil all time ranking: 8th



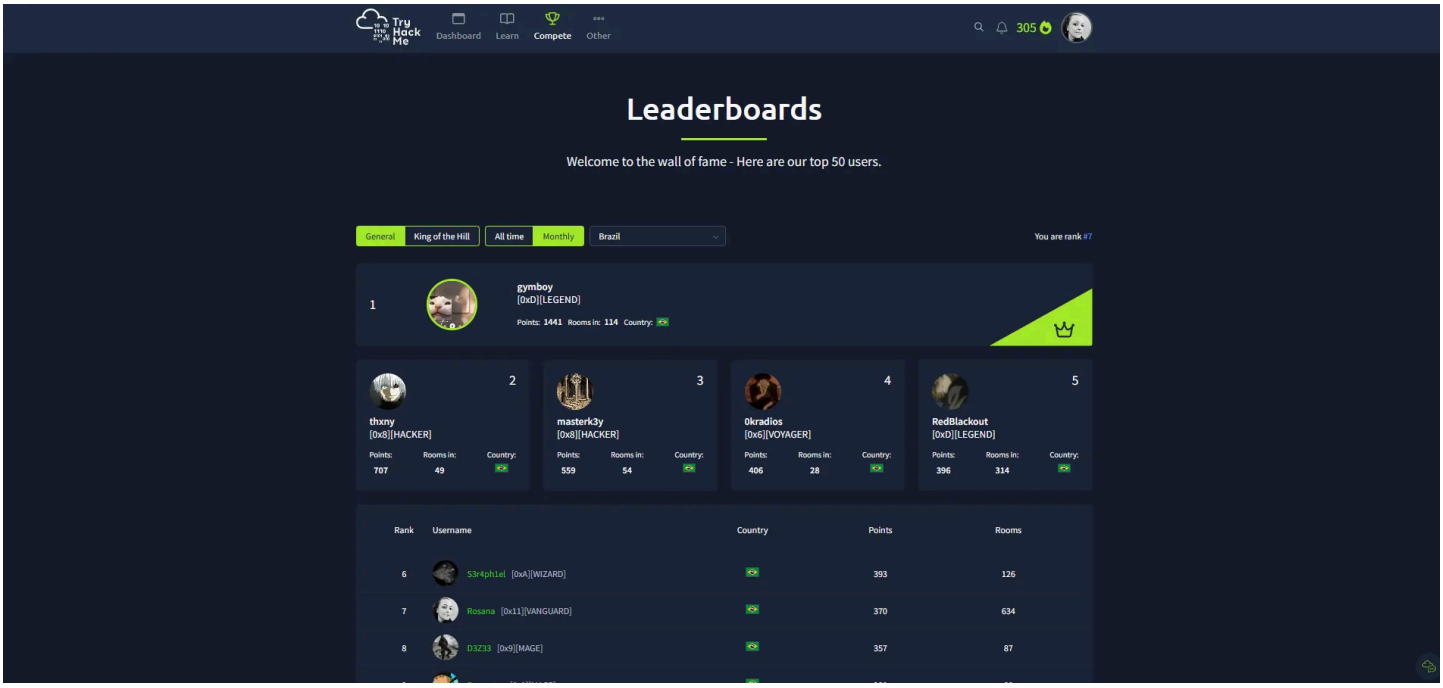
This image and all the theoretical content of the present article is TryHackMe´s property.

Global monthly ranking: 414th



This image and all the theoretical content of the present article is TryHackMe´s property.

Brazil monthly ranking: 7th



This image and all the theoretical content of the present article is TryHackMe´s property.

Thanks for coming! Happy Ethical Hacking!

Follow me for more about cybersecurity: Medium . GitHub . LinkedIN.

This challenge was created by tryhackme, and Dex01.
Thank you for investing your time and effort to develop it so that I can sharpen my skills!

Tryhackme | Tryhackme Walkthrough | Tryhackme Writeup | Cybersecurity | Ethical Hacking

Follow

Written by RosanaFSS

281 followers · 130 following

Information Security . Cybersecurity . Cloud Security . Penetration Testing

No responses yet

Write a response

What are your thoughts?

More from RosanaFSS

RosanaFSS

Windows Logging for SOC . TryHackMe Walkthrough

Start your Windows monitoring journey by learning how to use key system logs to detect threats.

Jun 28 54 1

RosanaFSS

Burp Suite: The Basics . TryHackMe Walkthrough

Web Fundamentals Learning Path > Burp Suite > An introduction to using Burp Suite for web application pentesting.

Oct 10, 2024 9

RosanaFSS

Lookup . TryHackMe Walkthrough . Enumeration

Test your enumeration skills on this boot-to-root machine.

Nov 24, 2024

RosanaFSS

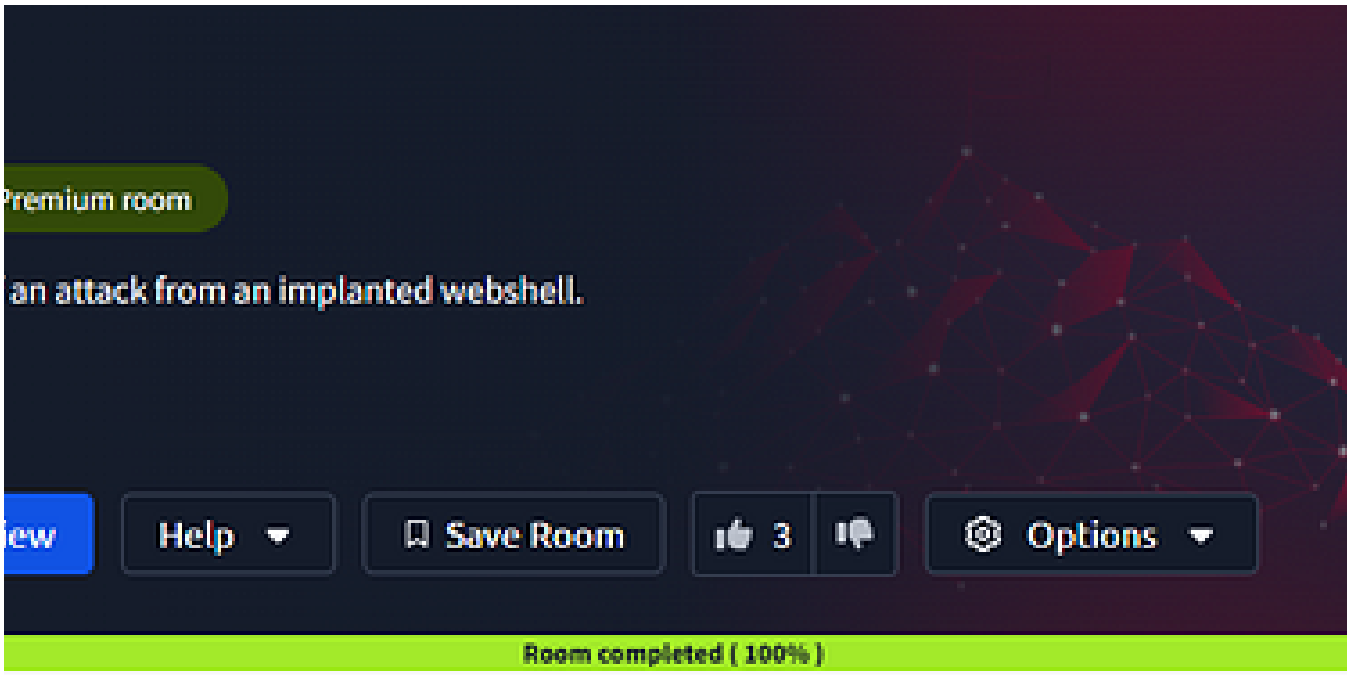
Offensive Pentesting . Vulnversity: TryHackMe Walkthrough

Offensive Pentesting learning path → Getting Started module → 02 of 04, Vulnversity. Practice Active Reconnaissance, Web Appplication...

Jan 7 4

See all from RosanaFSS

Recommended from Medium

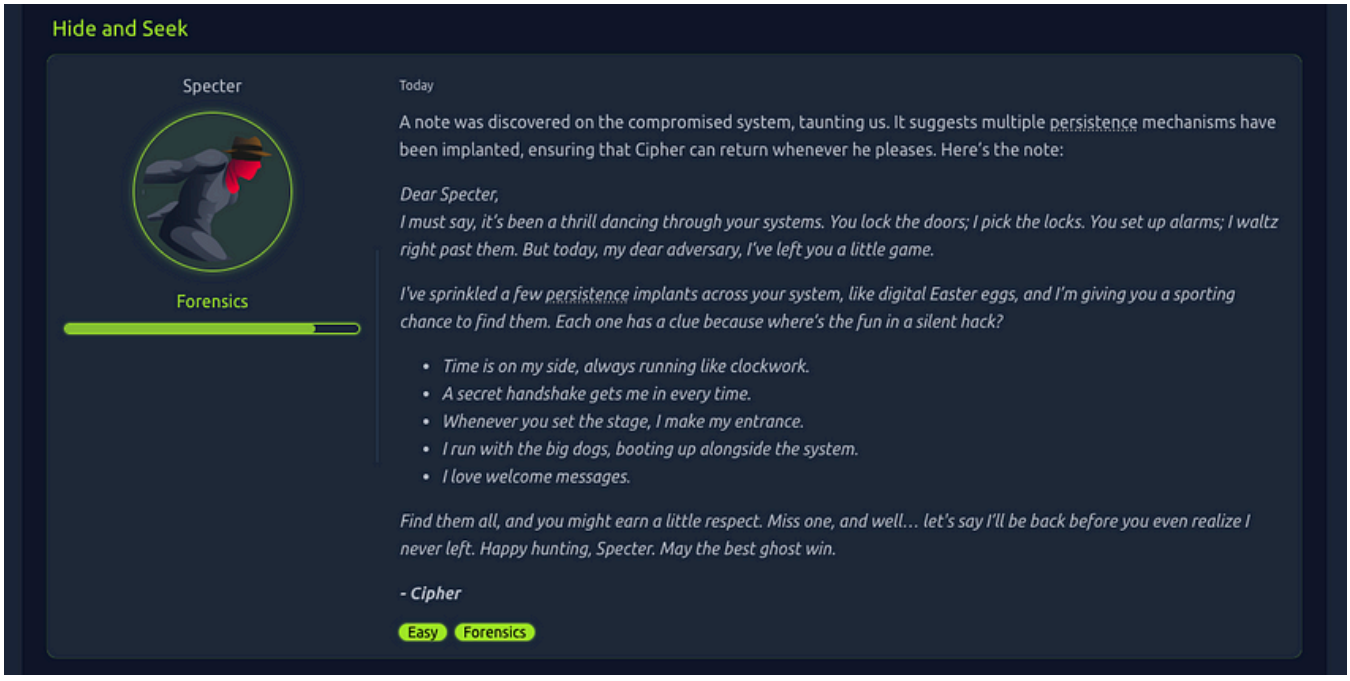


Sle3pyHead

Infinity Shell CTF Notes | TryHackMe

Quick reference notes for TryHackMe's Infinity Shell challenge.

Jul 11 15



Rahul Kumar

Hide and Seek | Tryhackme Walkthrough

Conduct a live system analysis to uncover post-compromise activity related to persistence mechanisms.

Jul 14

In T3CH by Ansul Kotadia

🔧 TryHackMe: Industrial Intrusion—Writeup

Solo-ing a team-based ICS challenge like a boss 🧠💥

Jun 27 119 1

VALKYRIE

Stolen Mount | TryHackMe Writeup | By VALKYRI3

Analyse network traffic related to an unauthenticated file share access attempt, focusing on potential signs of data exfiltration.

Jul 12 110

RosanaFSS

Security Operations, DFIR -Investigating Windows 3.x : a TryHackMe Walkthrough - 900 points

In the Investigating Windows 1.0 challenge we performed a brief analysis. Within Investigating Windows 2.0 we took a deeper dive into the...

Feb 5 43 1

In InfoSec Write-ups by Visir

Volt Typhoon APT Walkthrough — TryHackMe Room Investigation Using Splunk & Threat Hunting...

Step-by-Step Guide to Detect and Analyse Volt Typhoon Intrusion | Splunk Logs, Threat Intelligence, and SOC Workflow

Jun 10 164 2