

# NTFS Analysis - Digital Forensics : TryHackMe Walkthrough

22 min read · Mar 7, 2025



RosanaFSS

Follow

Listen

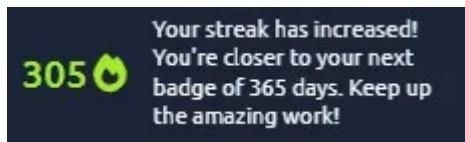
Share

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 305<sup>th</sup> 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!

## TryHackMe | Cyber Security Training

TryHackMe is a free online platform for learning cyber security, using hands-on exercises and labs, all through your...

[tryhackme.com](http://tryhackme.com)

## 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.

Open in app ↗

Sign up

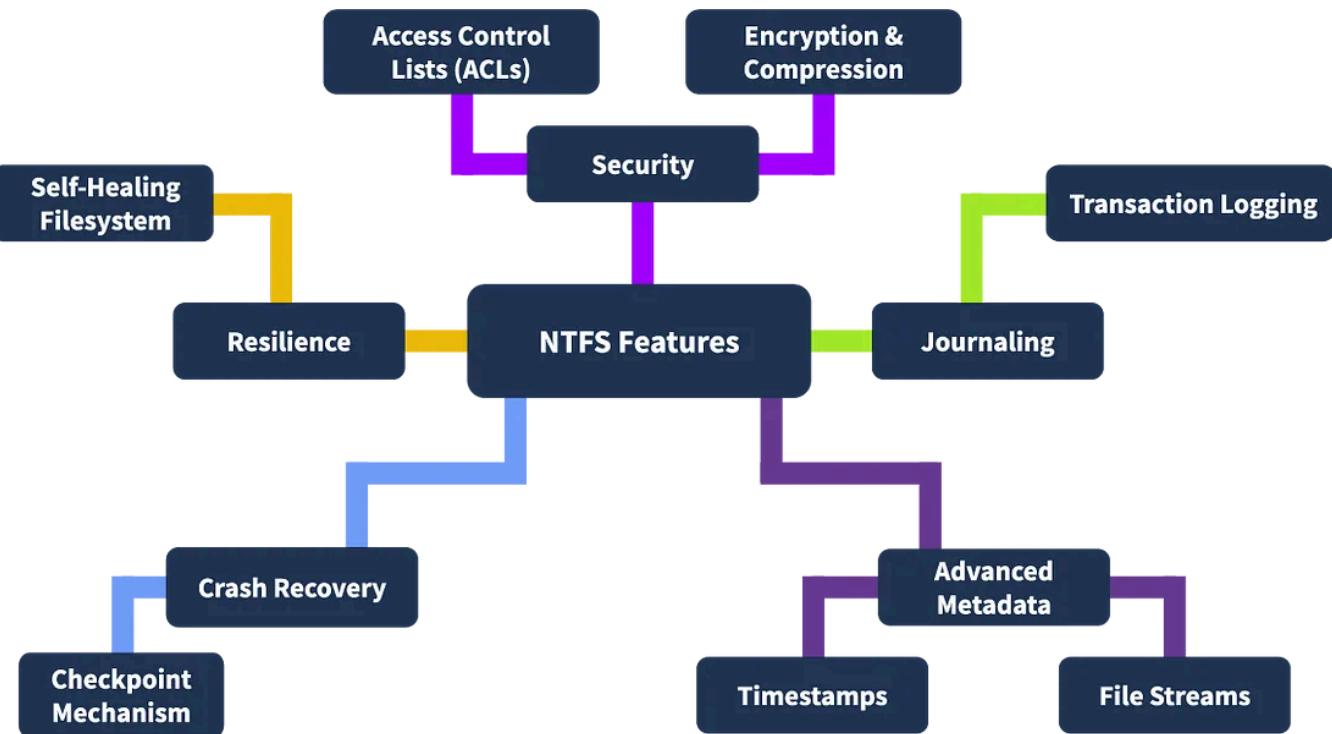
Sign in

Medium



Search





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

File
1      \$Mft - MFT
2      \$MftMirr - Log file
3      \$LogFile - Log file
4      \$Volume - Volume file
5      \$AttrDef - Attribute definition table
6      \ - Root directory
7      \$Bitmap - Volume duster allocation file
8      \$Boot - Boot sector
9      \$BadClus - Bad-duster file
10     \$Secure - Security settings file
11     \$Extend - Extended metadata directory
12     Unused
15     Unused
16     User files and directories

Reserved for NTFS  
metadata files

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).

Name	Size	Type	Date Modified
System Volume Information	1	Directory	2/26/2024 1:49:54 AM
tmp	1	Directory	3/4/2024 1:17:05 PM
Users	1	Directory	3/4/2024 10:23:36 AM
Windows	1	Directory	10/18/2022 9:51:57 PM
\$AttrDef	3	Regular File	3/11/2021 10:38:14 AM
\$BadClus	0	Regular File	3/11/2021 10:38:14 AM
\$Extend	960	Regular File	3/11/2021 10:38:14 AM
\$LogFile	8	Regular File	3/11/2021 10:38:14 AM
\$MFT	8	NTFS Index All...	12/6/2024 11:27:35 AM
\$MFTMirr	42,912	Regular File	3/11/2021 10:38:14 AM
\$Secure	601,344	Regular File	3/11/2021 10:38:14 AM
\$STXF_DATA	1	Regular File	3/11/2021 10:38:14 AM
\$UpCase	128	Regular File	3/11/2021 10:38:14 AM
\$Volume	0	Regular File	3/11/2021 10:38:14 AM
bootmgr	402	Regular File	10/13/2022 9:18:02 PM
BOOTNXT	1	Regular File	9/15/2018 7:12:30 AM
pagefile.sys	720,896	Regular File	12/6/2024 11:24:37 AM
S130	SI30 INDX Entry		

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.

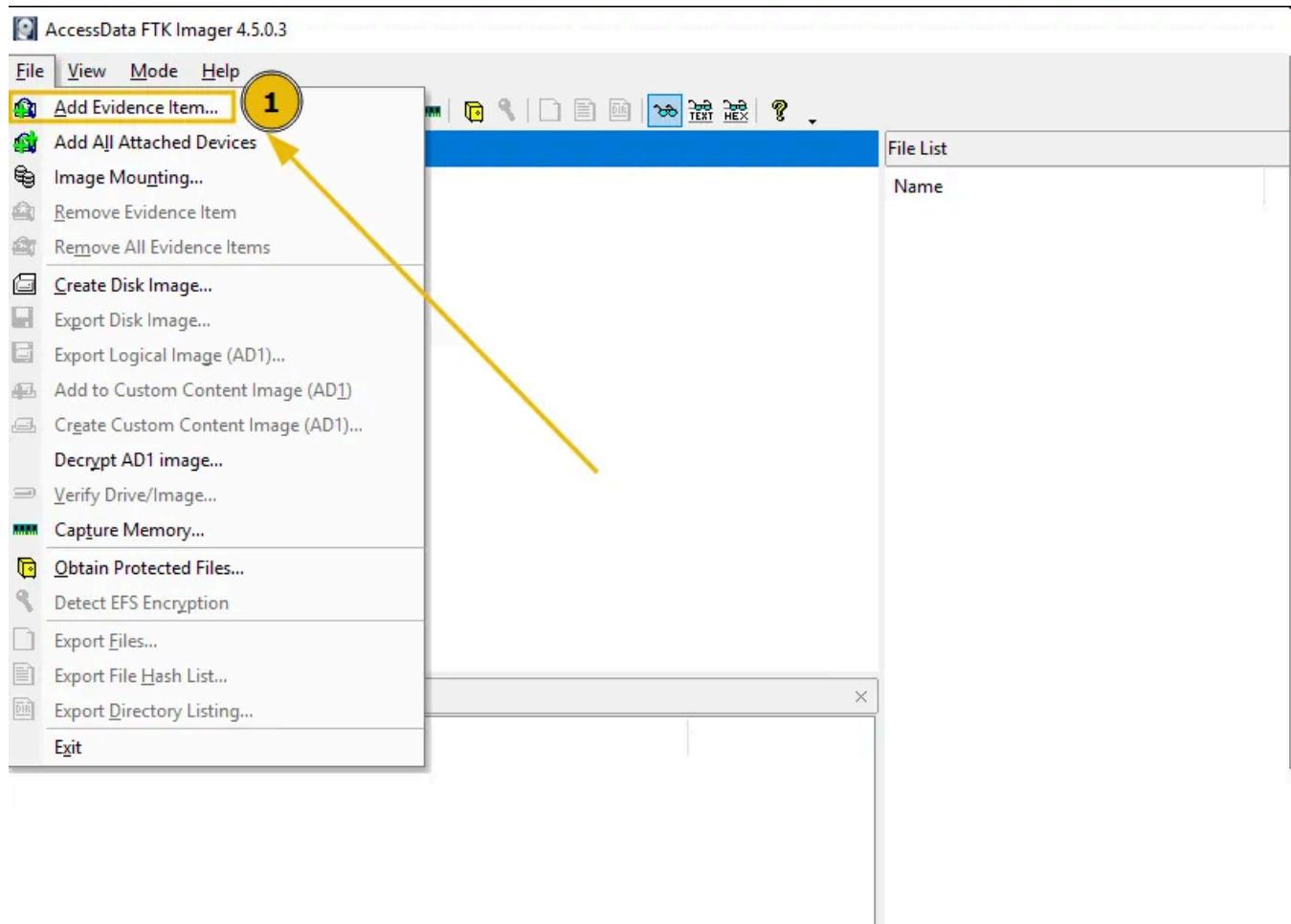
## Alternate Data Stream

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

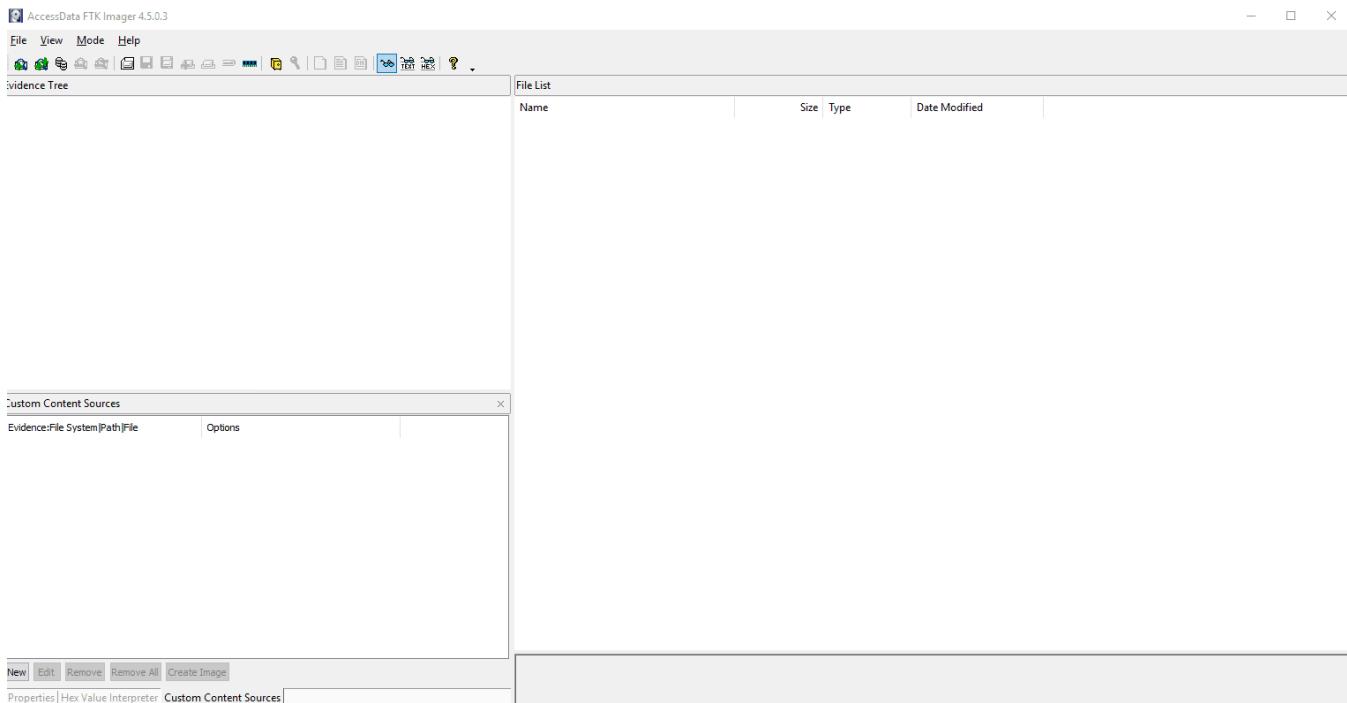
To extract and examine the NTFS disk and its components, we will use the powerful forensics tool called FTK Imager, which can be found on the taskbar.

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



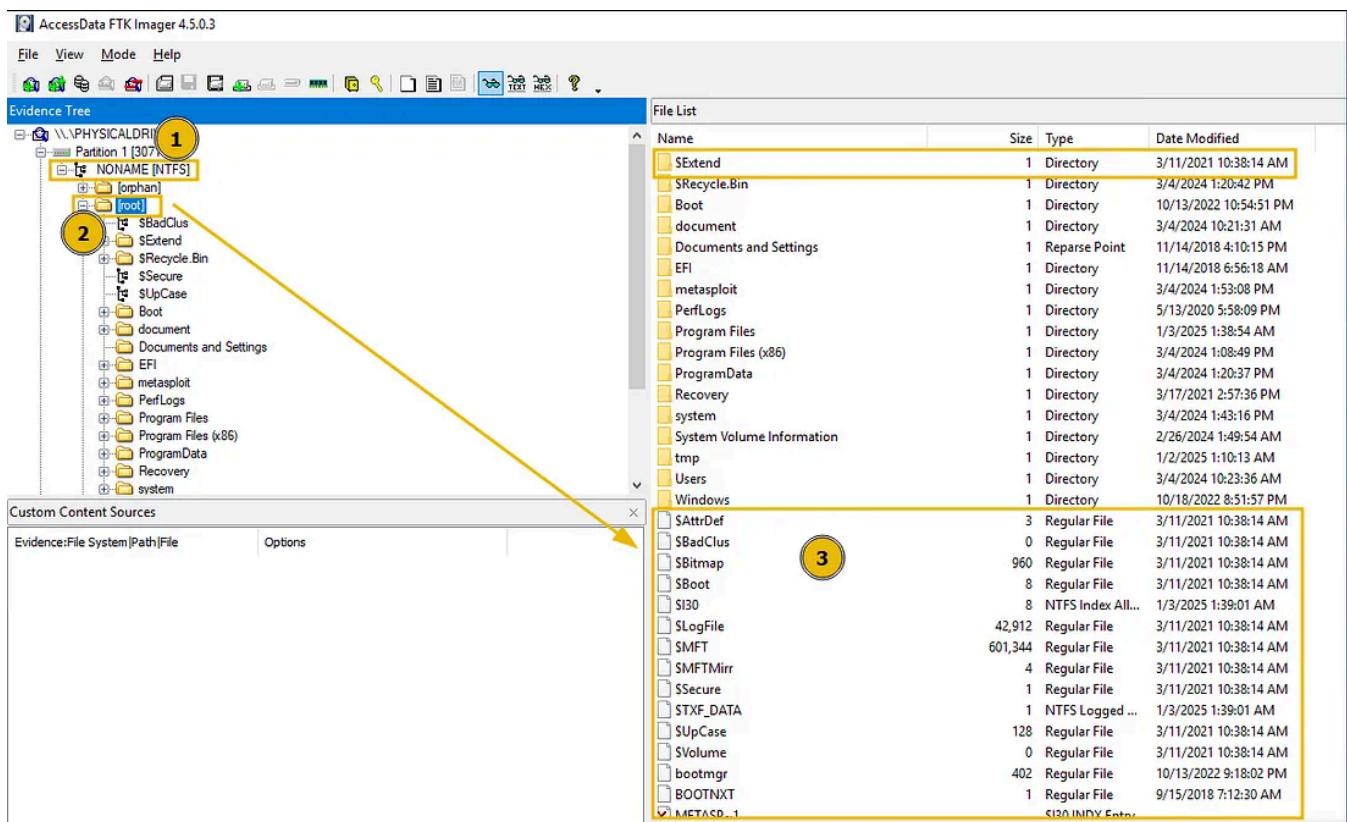
This image and all the theoretical content of the present article is [TryHackMe](#)'s property.

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:



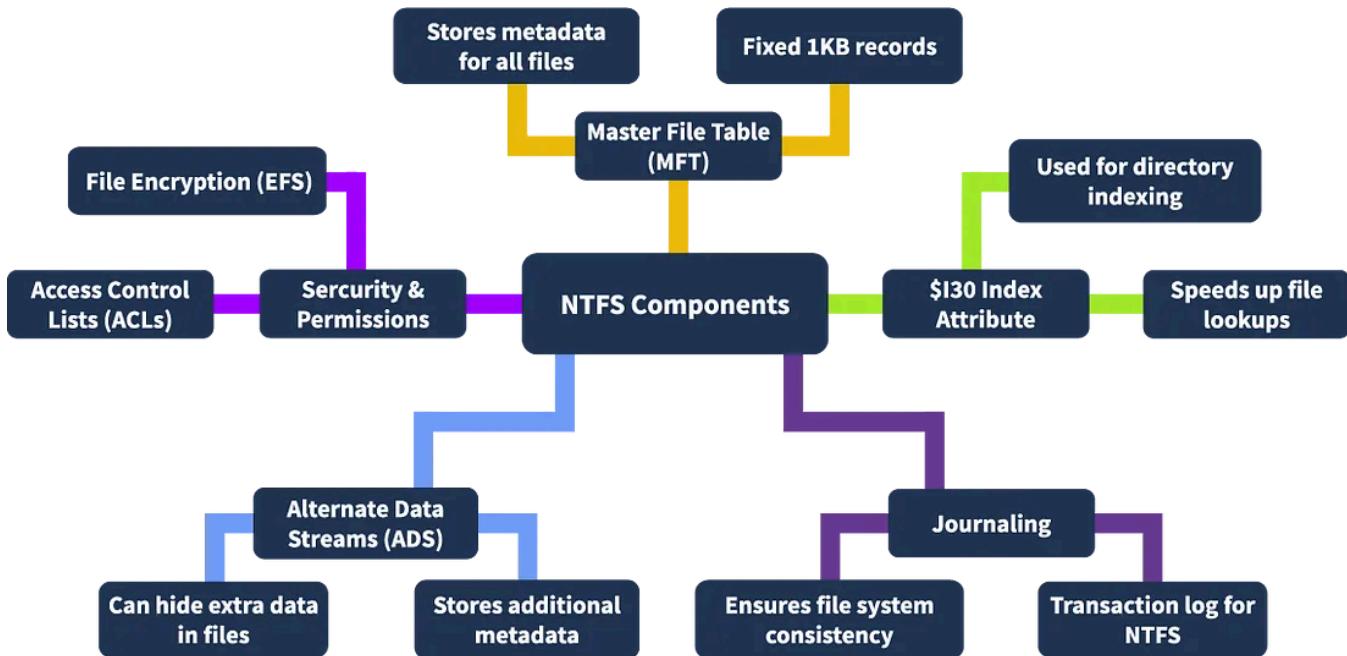
This image and all the theoretical content of the present article is [TryHackMe](#)'s property.

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.



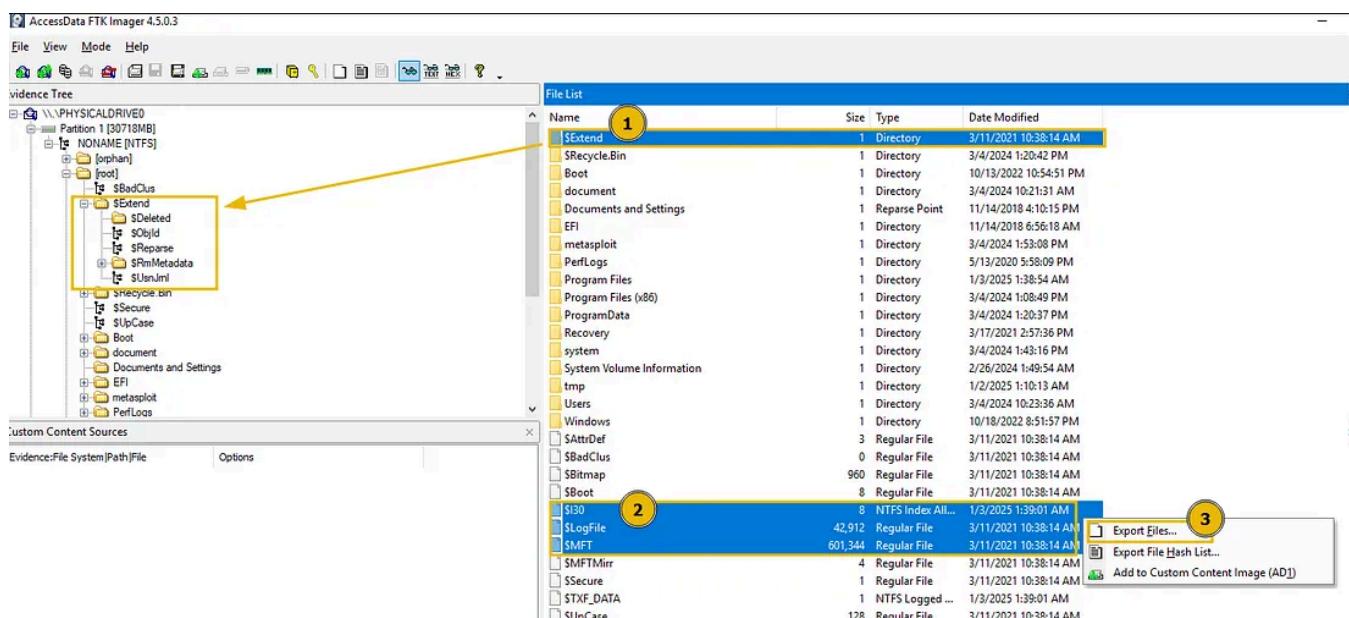
This image and all the theoretical content of the present article is [TryHackMe](#)'s property.

## 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:



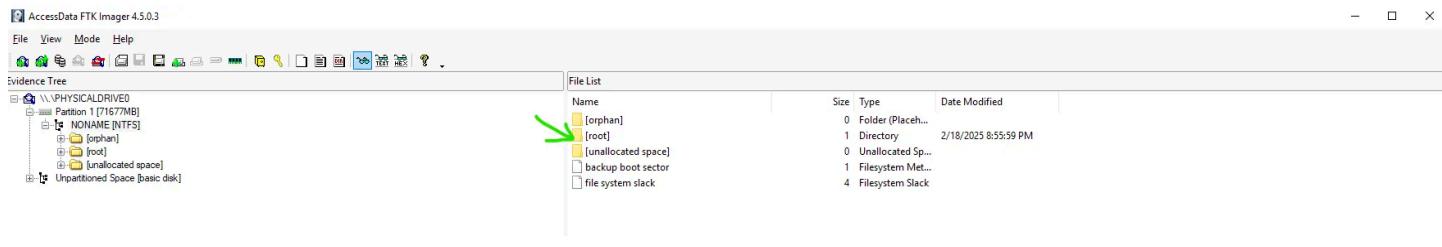
This image and all the theoretical content of the present article is [TryHackMe](#)'s property.

*Answer the question below*

**4.1.** Double-click on the \$UsnJrn1 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.

AccessData FTK Imager 4.5.0.3

Evidence Tree

File List

Name	Size	Type	Date Modified
EFI	1	Directory	11/14/2018 6:56:18 AM
metasploit	1	Directory	3/4/2024 1:53:08 PM
PerfLogs	1	Directory	5/13/2020 5:58:09 PM
Program Files	1	Directory	1/3/2025 1:38:54 AM
Program Files (x86)	1	Directory	3/4/2024 1:08:49 PM
ProgramData	1	Directory	3/4/2024 1:20:37 PM
Recovery	1	Directory	3/17/2021 2:57:36 PM
secret_folder	1	Directory	1/15/2025 7:58:22 AM
System	1	Directory	3/4/2024 1:43:16 PM
System Volume Information	1	Directory	2/26/2024 1:49:54 AM
tmp	1	Directory	1/6/2025 2:46:37 AM
Users	1	Directory	3/4/2024 10:23:36 AM
Windows	1	Directory	10/18/2022 8:51:57 PM
\$AttrDef	3	Regular File	3/11/2021 10:38:14 AM
\$BadClus	0	Regular File	3/11/2021 10:38:14 AM
\$Bitmap	2,240	Regular File	3/11/2021 10:38:14 AM
\$Boot	8	Regular File	3/11/2021 10:38:14 AM
\$I30	8	NTFS Index All...	2/18/2025 8:55:59 PM
\$LogFile	42,912	Regular File	3/11/2021 10:38:14 AM
\$MFT	601,344	Regular File	3/11/2021 10:38:14 AM
\$MFTMirr	4	Regular File	3/11/2021 10:38:14 AM
\$Secure	1	Regular File	3/11/2021 10:38:14 AM
\$TxF_DATA	1	NTFS.Logged...	2/18/2025 8:55:59 PM
\$UpCase	128	Regular File	3/11/2021 10:38:14 AM
\$Volume	0	Regular File	3/11/2021 10:38:14 AM
bootmgr	402	Regular File	10/13/2022 9:18:02 PM
BOOTINVT	1	Regular File	9/15/2018 7:12:30 AM
METASP-1	0	\$I30.INDX Entry	
pagefile.sys	720,896	Regular File	3/7/2025 3:41:30 PM

Custom Content Sources

New Edit Remove Remove All Create Image

Properties Hex Value Interpreter Custom Content Sources

Cursor pos = 0

FTK Imager

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

Evidence Tree

File List

ContextMenu

- Export Files...
- Export File Hash List...
- Add to Custom Content Image (ADI)

FTK Imager

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

AccessData FTK Imager 4.5.0.3

Evidence Tree

File List

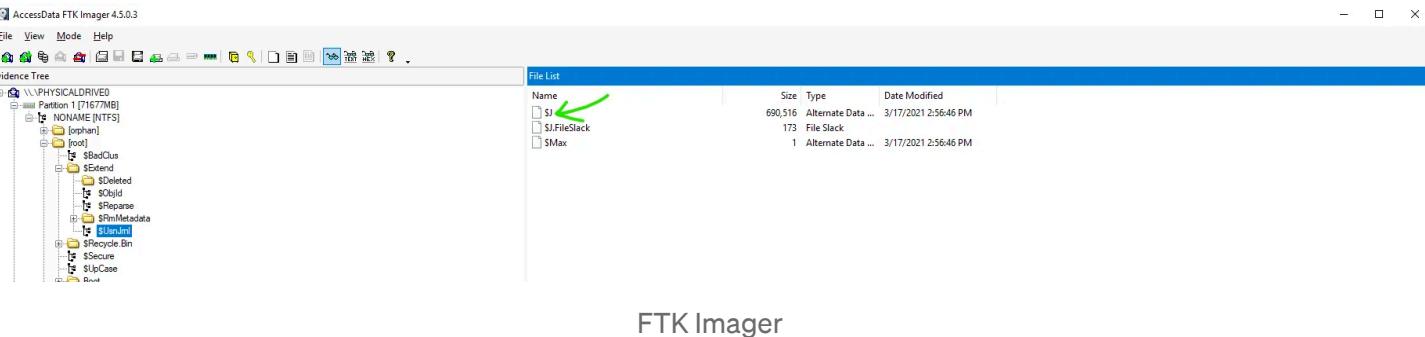
Export Results

0 folder(s) and 3 file(s) exported successfully.  
659726336 bytes copied.

OK

FTK Imager

- Clicked on the \$usnJrn1 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:

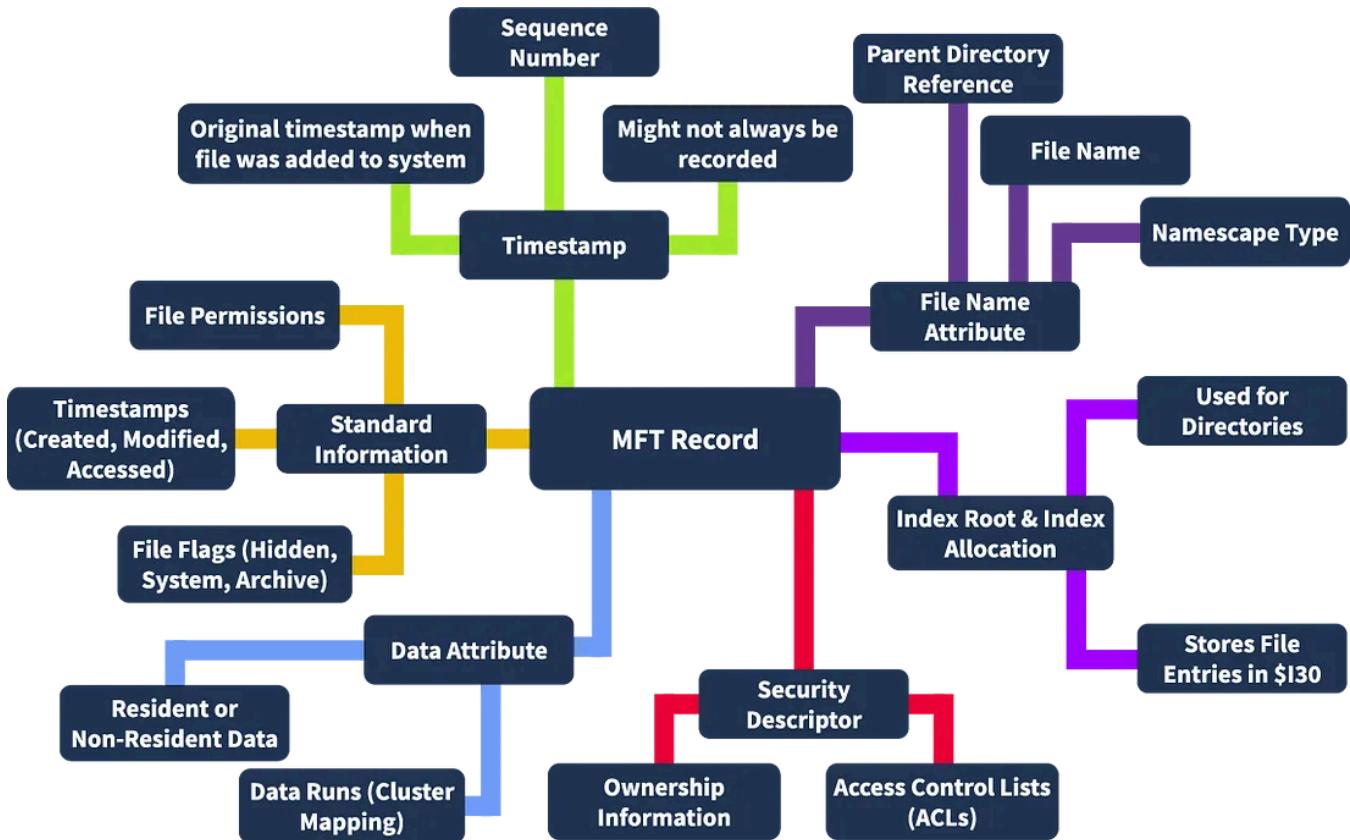
The screenshot shows the AccessData FTK Imager interface. The Evidence Tree pane on the left shows a tree structure of files and folders on a physical drive. The File List pane on the right shows a table of files with columns for Name, Size, Type, and Date Modified. The row for 'SMFT' is highlighted. Below the table is a large preview window showing the raw binary data of the file, with a yellow arrow pointing to the start of the data. The preview window has a scroll bar.

Name	Size	Type	Date Modified
SI30	8	NTFS Index All...	1/3/2025 1:39:01 AM
\$LogFile	42,912	Regular File	3/11/2021 10:38:14 AM
<b>SMFT</b>	601,344	Regular File	3/11/2021 10:38:14 AM
\$MFTMirr	4	Regular File	3/11/2021 10:38:14 AM
SSecure	1	Regular File	3/11/2021 10:38:14 AM
\$TxF_DATA	1	NTFS Logged ...	1/3/2025 1:39:01 AM
\$ImFree	179	Regular File	3/11/2021 10:38:14 AM

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

```
C:\Users\Administrator\Desktop\EZ tools>MFTECmd.exe --help
Description:
MFTECmd version 1.2.2.1

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

Examples: MFTECmd.exe -f "C:\Temp\SomeMFT" --csv "c:\temp\out" --csvf MyOutputFile.csv
          MFTECmd.exe -f "C:\Temp\SomeMFT" --csv "c:\temp\out"
          MFTECmd.exe -f "C:\Temp\SomeMFT" --json "c:\temp\jsonout"
          MFTECmd.exe -f "C:\Temp\SomeMFT" --body "c:\temp\bout" --bdl c
          MFTECmd.exe -f "C:\Temp\SomeMFT" --de 5-5
          MFTECmd.exe -f "C:\Temp\SomeMFT" --csv "c:\temp\out" --dr --fl
          MFTECmd.exe -f "c:\temp\SomeJ" --csv "c:\temp\out"
          MFTECmd.exe -f "c:\temp\SomeJ" -m "C:\Temp\SomeMFT" --csv "c:\temp\out"
          MFTECmd.exe -f "c:\temp\SomeBoot"
          MFTECmd.exe -f "c:\temp\SomeSecure_SDS" --csv "c:\temp\out"
          MFTECmd.exe -f "c:\temp\SomeI30" --csv "c:\temp\out"

Short options (single letter) are prefixed with a single dash. Long commands are prefixed with two dashes

Usage:
MFTECmd [options]

Options:
-f <f>           File to process ($MFT | $J | $Boot | $SDS | $I30). Required
-m <m>           $MFT file to use when -f points to a $J file (Use this to resolve parent path in $J CSV output)
--json <json>     Directory to save JSON formatted results to. This or --csv required unless --de or --body is specified
--jsonf <jsonf>   File name to save JSON formatted results to. When present, overrides default name
--csv <csv>       Directory to save CSV formatted results to. This or --json required unless --de or --body is specified
--csvf <csvf>    File name to save CSV formatted results to. When present, overrides default name
--body <body>     Directory to save bodyfile formatted results to. --bdl is also required when using this option
--bodyf <bodyf>   File name to save body formatted results to. When present, overrides default name
--bdl <bdl>       Drive letter (C, D, etc.) to use with bodyfile. Only the drive letter itself should be provided
--blf             When true, use LF vs CRLF for newlines [default: False]
--dd <dd>         Directory to save exported $MFT FILE record. --do is also required when using this option
--do <do>         Offset of the $MFT FILE record to dump as decimal or hex. Ex: 5120 or 0x1400 Use --de or --debug to see offsets
--de <de>         Dump full details for $MFT entry/sequence #. Format is 'Entry' or 'Entry-Seq' as decimal or hex.
                  Example: 5, 624-5 or 0x270-0x5.
--dr              When true, dump $MFT resident files to dir specified by --csv or --json, in 'Resident' subdirectory.
                  Files will be named '<EntryNumber>-<SequenceNumber>_<FileName>.bin'
--fls             When true, displays contents of directory from $MFT specified by --de. Ignored when --de points to a file [default: False]
--ds <ds>         Dump full details for Security Id from $SDS as decimal or hex. Example: 624 or 0x270
--dt <dt>         The custom date/time format to use when displaying time stamps. See https://goo.gl/CNVq0k for options [default: yyyy-MM-dd HH:mm:ss.fffffffff]
--sn              Include DOS file name types in $MFT output [default: False]
--fl              Generate condensed file listing of parsed $MFT contents. Requires --csv [default: False]
--at              When true, include all timestamps from 0x30 attribute vs only when they differ from 0x10 in the $MFT [default: False]
```

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**

```
C:\Users\Administrator\Desktop\EZ tools>MFTECmd.exe -f ..\Evidence\$MFT --csv ..\Evidence --csvf ..\Evidence\MFT_record.csv
MFTECmd version 1.2.2.1
author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/MFTECmd

Command line: -f ..\Evidence\$MFT --csv ..\Evidence --csvf ..\Evidence\MFT_record.csv
file type: Mft
Processed ..\Evidence\$MFT in 14.6278 seconds

..\Evidence\$MFT: FILE records found: 559,445 (Free records: 41,828) File size: 587.2MB
CSV output will be saved to ..\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:

Line	Tag	Entry Number	Sequence Number	Parent Entry Number	Parent Sequence Number	In Use	Parent Path	File Name
1		0	1	5	5	<input checked="" type="checkbox"/>	.	\$MFT
2		1	1	5	5	<input checked="" type="checkbox"/>	.	\$MFTMirr
3		2	2	5	5	<input checked="" type="checkbox"/>	.	\$LogFile
4		3	3	5	5	<input checked="" type="checkbox"/>	.	\$Volume
5		4	4	5	5	<input checked="" type="checkbox"/>	.	\$AttrDef
6		5	5	5	5	<input checked="" type="checkbox"/>	.	.
7		6	6	5	5	<input checked="" type="checkbox"/>	.	\$Bitmap
8		7	7	5	5	<input checked="" type="checkbox"/>	.	\$Boot
9		8	8	5	5	<input checked="" type="checkbox"/>	.	\$BadClus
10		8	8	5	5	<input checked="" type="checkbox"/>	.	\$BadClus:\$Bad
11		9	9	5	5	<input checked="" type="checkbox"/>	.	\$Secure
12		9	9	5	5	<input checked="" type="checkbox"/>	.	\$Secure:\$SDS
13		10	10	5	5	<input checked="" type="checkbox"/>	.	\$UpCase
14		10	10	5	5	<input checked="" type="checkbox"/>	.	\$UpCase:\$Info
15		11	11	5	5	<input checked="" type="checkbox"/>	.	\$Extend
16		24	1	11	11	<input checked="" type="checkbox"/>	.	\$Quota
17		25	1	11	11	<input checked="" type="checkbox"/>	.	\$ObjId
18		26	1	11	11	<input checked="" type="checkbox"/>	.	\$Reparse
19		27	1	11	11	<input checked="" type="checkbox"/>	.	\$RmMetadata
20		28	1	27	1	<input checked="" type="checkbox"/>	.	\$Repair
21		28	1	27	1	<input checked="" type="checkbox"/>	.	\$Repair:\$Config
22		28	1	27	1	<input checked="" type="checkbox"/>	.	\$Repair:\$Corrupt
23		28	1	27	1	<input checked="" type="checkbox"/>	.	\$Repair:\$Verify
24		29	1	27	1	<input checked="" type="checkbox"/>	.	\$TxfLog
25		30	1	27	1	<input checked="" type="checkbox"/>	.	\$Txf

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 sequence 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

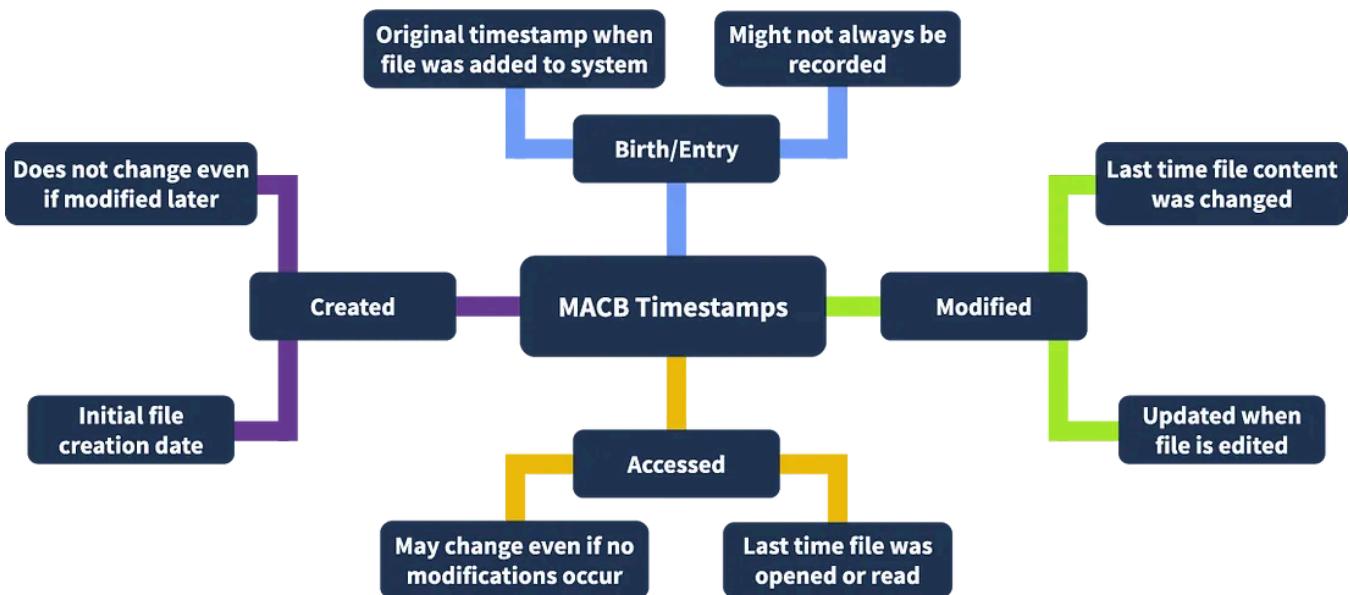
- **Description:** The amount of disk space allocated to the file.
- **Importance:** Helps identify slack space or fragmented files.

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 timestamping attack. We can examine the timestamp to identify the manipulation of the timestamps of files or directories.

*Answer the questions below*

Get RosanaFSS's stories in your inbox

Join Medium for free to get updates from this writer.

Enter your email

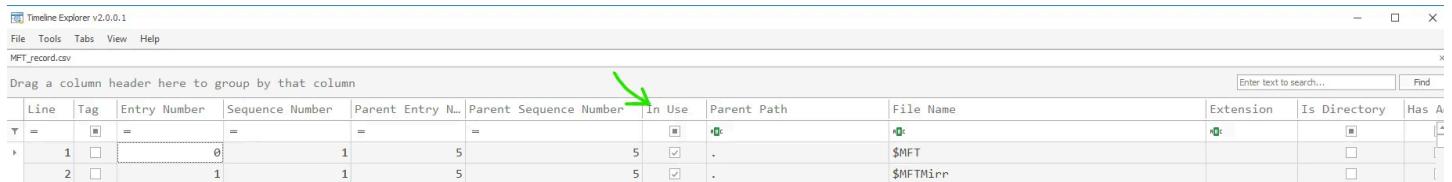
Subscribe

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

In Use

In Use														
<ul style="list-style-type: none"> <li><b>Description:</b> Indicates whether the current record is associated with the active file or directory.</li> <li><b>Importance:</b> 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.</li> </ul>														

This image and all the theoretical content of the present article is [TryHackMe](#)'s property.



Line	Tag	Entry Number	Sequence Number	Parent Entry N...	Parent Sequence Number	In Use	Parent Path	File Name	Extension	Is Directory	Has A...
Y =	=	=	=	=	=						
1	□	0	1	5	5	✓	.	\$MFT	▪	□	▲
2	□	1	1	5	5	✓	.	\$MFTMirr	□	□	□

Timeline Explorer

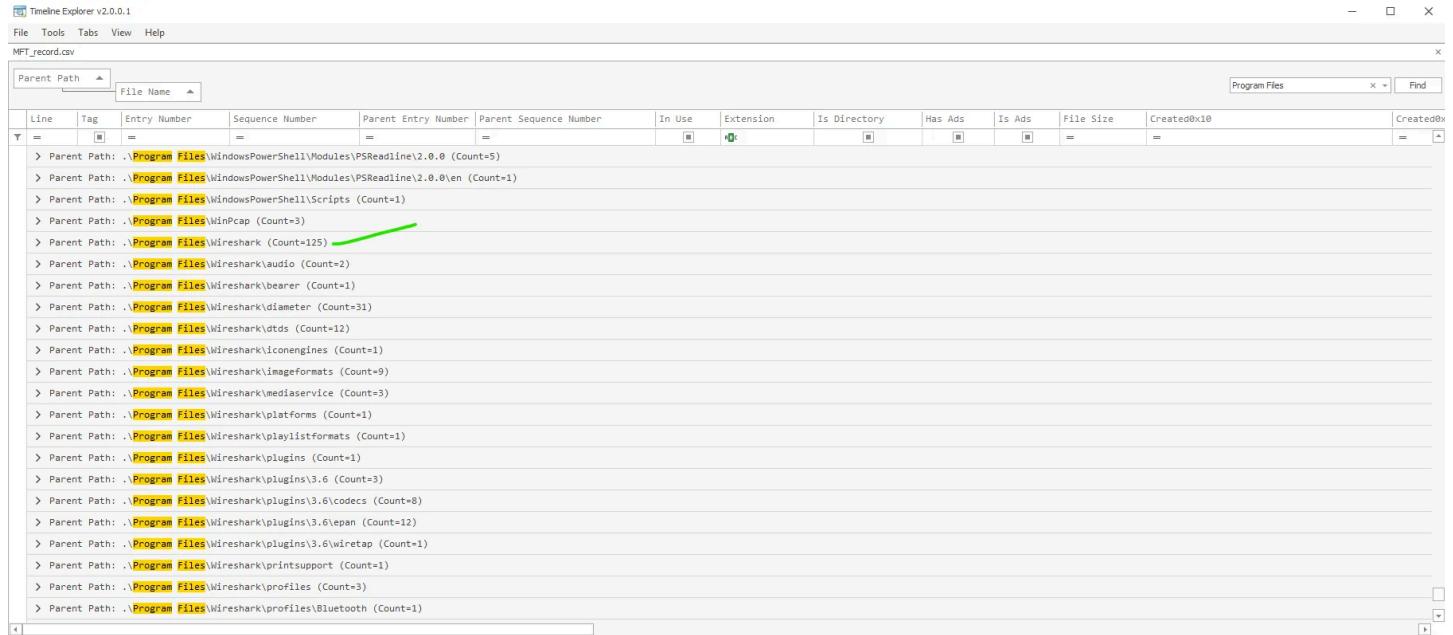
## 5.2. Examine the MFT record; what is the network sniffer installed on this system in the \Program Files\ directory? Hint : Apply filter on the given path in the Parent Path Column.

wireshark

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

```
Administrator: Command Prompt - MFTECmd.exe -f ..\Evidence\$MFT --csv ..\Evidence --csvf ..\Evidence\MFT_record.csv
C:\Users\Administrator\Desktop\EZ tools>MFTECmd.exe -f ..\Evidence\$MFT --csv ..\Evidence --csvf ..\Evidence\MFT_record.csv
MFTECmd version 1.2.2.1
Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/MFTECmd
Command line: -f ..\Evidence\$MFT --csv ..\Evidence --csvf ..\Evidence\MFT_record.csv
File type: Mft
Processed ..\Evidence\$MFT in 11.2154 seconds
..\Evidence\$MFT: FILE records found: 560,241 (Free records: 41,032) File size: 587.2MB
CSV output will be saved to ..\Evidence\MFT_record.csv
```

- 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.



The screenshot shows the Timeline Explorer interface with a search filter applied to the 'Parent Path' column. The results list various Windows system paths under 'Program Files'. A green arrow points from the search bar to the 'Parent Path' column header.

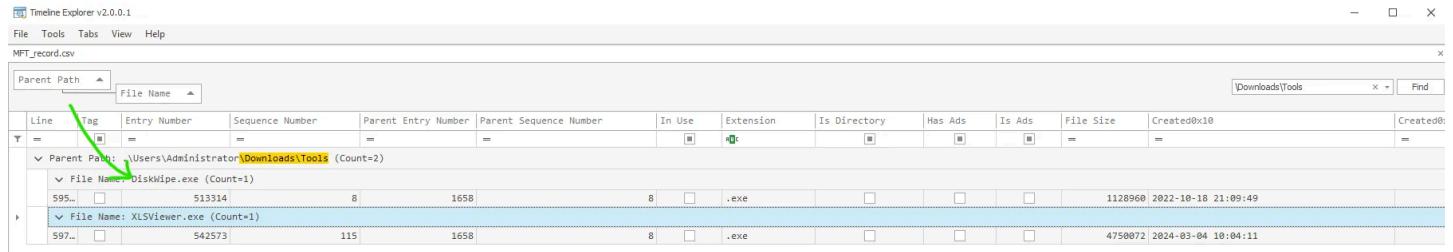
Line	Tag	Entry Number	Sequence Number	Parent Entry Number	Parent Sequence Number	In Use	Extension	Is Directory	Has Ads	Is Ads	File Size	Created0x10	Created0x
Y =	[ ]	=	=	=	=	[ ]	[ ]	[ ]	[ ]	[ ]	=	=	=
> Parent Path: .\Program Files\WindowsPowerShell\Modules\PSReadline\2.0.0 (Count=5)													
> Parent Path: .\Program Files\WindowsPowerShell\Modules\PSReadline\2.0.0\en (Count=1)													
> Parent Path: .\Program Files\WindowsPowerShell\Scripts (Count=1)													
> Parent Path: .\Program Files\WinPcap (Count=3)													
> Parent Path: .\Program Files\Wireshark (Count=125)													
> Parent Path: .\Program Files\Wireshark\audio (Count=2)													
> Parent Path: .\Program Files\Wireshark\bearer (Count=1)													
> Parent Path: .\Program Files\Wireshark\diameter (Count=31)													
> Parent Path: .\Program Files\Wireshark\ddts (Count=12)													
> Parent Path: .\Program Files\Wireshark\lconengines (Count=1)													
> Parent Path: .\Program Files\Wireshark\imageformats (Count=9)													
> Parent Path: .\Program Files\Wireshark\mediaservice (Count=3)													
> Parent Path: .\Program Files\Wireshark\platforms (Count=1)													
> Parent Path: .\Program Files\Wireshark\playlistformats (Count=1)													
> Parent Path: .\Program Files\Wireshark\plugins (Count=1)													
> Parent Path: .\Program Files\Wireshark\plugins3.6 (Count=3)													
> Parent Path: .\Program Files\Wireshark\plugins3.6\codecs (Count=8)													
> Parent Path: .\Program Files\Wireshark\plugins3.6\epan (Count=12)													
> Parent Path: .\Program Files\Wireshark\plugins3.6\wiretap (Count=1)													
> Parent Path: .\Program Files\Wireshark\printsupport (Count=1)													
> Parent Path: .\Program Files\Wireshark\profiles (Count=3)													
> Parent Path: .\Program Files\Wireshark\profiles\Bluetooth (Count=1)													

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 .



The screenshot shows the Timeline Explorer interface with a search filter applied to the 'Parent Path' column. The results list files in the '\Downloads\Tools' directory. A green arrow points from the search bar to the 'Parent Path' column header.

Line	Tag	Entry Number	Sequence Number	Parent Entry Number	Parent Sequence Number	In Use	Extension	Is Directory	Has Ads	Is Ads	File Size	Created0x10	Created0x
Y =	[ ]	=	=	=	=	[ ]	[ ]	[ ]	[ ]	[ ]	=	=	=
> Parent Path: .\Users\Administrator\Downloads\Tools (Count=2)													
> File Name: DiskWipe.exe (Count=1)													
595.. [ ] 513314 8 1658 [ ] 8 [ ] .exe [ ] [ ] [ ] 1128960 2022-10-18 21:09:49													
> File Name: XLSViewer.exe (Count=1)													
597.. [ ] 542573 115 1658 [ ] 8 [ ] .exe [ ] [ ] [ ] 4750072 2024-03-04 10:04:11													

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.

The screenshot shows the Timeline Explorer interface with the file 'MFT\_record.csv' loaded. The main pane displays a table of MFT records. A green arrow points to the 'File Name' column header. The table has columns for Line, Tag, Entry Number, Sequence Number, Parent Entry Number, Parent Sequence Number, In Use, Extension, Is Directory, Has Ads, Is Ads, File Size, Created@x10, and Created@x30. Two entries are visible:

Line	Tag	Entry Number	Sequence Number	Parent Entry Number	Parent Sequence Number	In Use	Extension	Is Directory	Has Ads	Is Ads	File Size	Created@x10	Created@x30
▼	=	=	=	=	=	<input checked="" type="checkbox"/>	.exe	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1128960	2022-10-18 21:09:49	
▼ Parent Path:	File Name:	DiskWipe.exe (Count=2)											
In Use: Unchecked (Count=1)		513314	8	1658		<input type="checkbox"/>	.exe	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
File Name: XLSViewer.exe (Count=1)		542573	115	1658		<input type="checkbox"/>	.exe	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4750072	2024-03-04 10:04:11	

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 .

The screenshot shows the Timeline Explorer interface with the file 'MFT\_record.csv' loaded. A green arrow points to the search bar at the top right, which contains 'flag.txt'. The main pane displays a table of MFT records. The table has the same columns as the previous screenshot. One entry is visible, with a green arrow pointing to the 'File Name' column:

Line	Tag	Entry Number	Sequence Number	Parent Entry Number	Parent Sequence Number	In Use	Extension	Is Directory	Has Ads	Is Ads	File Size	Created@x10	Created@x30
▼	=	=	=	=	=	<input checked="" type="checkbox"/>	.txt	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	20	2025-01-02 01:07:10	
▼ Parent Path:	File Name:	.\tmp\secret_directory\flag.txt (Count=1)											
94815	<input type="checkbox"/>	93980	19	93317		<input checked="" type="checkbox"/>	.txt	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			

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 .

AccessData FTK Imager 4.5.0.3

Evidence Tree

- \PhysicalDrives\Partition 1 (71677MB)
  - NONAME (NTFS)
    - [orphan]
    - [lost]
    - \$BadClus
    - \$Extent
    - \$Deleted
    - \$ObjId
    - \$Reparse
    - \$RMMetadata
    - \$TLogFile
    - \$SecureBin
    - \$Secure
    - \$SecureCase
    - Boot
    - document
    - Documents and Settings
    - EFI
    - metasploit
    - PerfLogs
    - Program Files
    - Program Files (x86)
    - ProgramData
    - Programs
    - secret\_folder
    - System
    - System Volume Information
    - tmp
    - secret\_directory
    - Users
    - Windows
    - [unallocated space]
- Unpartitioned Space [basic disk]

File List

Name	Type	Date Modified
flag.txt	Regular File	1/6/2025 2:48:26 AM

Custom Content Sources

New Edit Remove Remove All Create Image

Properties Hex Value Interpreter Custom Content Sources

Cursor pos = 0

FTK Imager

– Clicked over flag.txt .

AccessData FTK Imager 4.5.0.3

Evidence Tree

- \PhysicalDrives\Partition 1 (71677MB)
  - NONAME (NTFS)
    - [orphan]
    - [lost]
    - \$BadClus
    - \$Extent
    - \$Deleted
    - \$ObjId
    - \$Reparse
    - \$RMMetadata
    - \$TLogFile
    - \$SecureBin
    - \$Secure
    - \$SecureCase
    - Boot
    - document
    - Documents and Settings
    - EFI
    - metasploit
    - PerfLogs
    - Program Files
    - Program Files (x86)
    - ProgramData
    - Programs
    - secret\_folder
    - System
    - System Volume Information
    - tmp
    - secret\_directory
    - Users
    - Windows
    - [unallocated space]
- Unpartitioned Space [basic disk]

File List

Name	Type	Date Modified
flag.txt	Regular File	1/6/2025 2:48:26 AM

Custom Content Sources

Properties Hex Value Interpreter Custom Content Sources

We1Done\_You\_Found\_M3

Cursor pos = 0

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 v2.0.0.1

MFT\_record.csv

Drag a column header here to group by that column

Line	Tag	Entry Number	Sequence	Parent Entr...	Parent Sequ...	In Use	Parent Path	Extension	File Name
Y		584574		550355	1	555274	1	.gz	runner_is.exe
>		622352		584574	2	568990	1	.ps1	SharpHound.ps1

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

[ ... ]

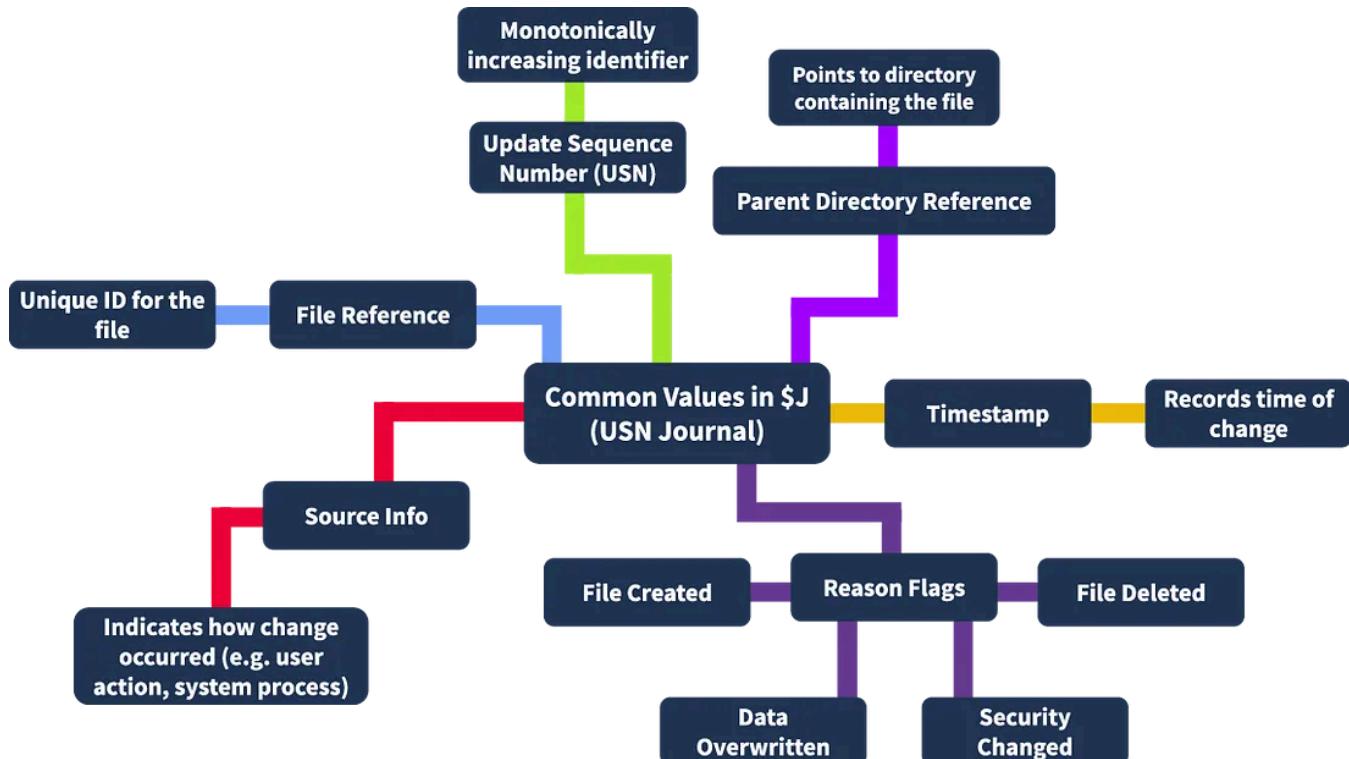
## Examining the Updated Reason

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.

Evidence Tree

- [root]
  - \$BadClus
  - \$Extend
  - \$Deleted
  - \$Old
  - \$Reparse
  - \$RmMetadata
  - \$Recycle.Bin
  - \$Secure
  - \$UCase
  - Boot
  - document
  - Documents and Settings
  - EFI
  - metasploit
  - PerfLogs
  - Program Files
  - Program Files (x86)
  - ProgramData
  - Recovery
  - secret\_folder
  - System
  - System Volume Information
  - tmp
  - secret\_directory
  - Icons

File List

Name	Size	Type	Date Modified
\$Deleted	1	Directory	3/17/2021 2:56:30 PM
\$Old	1	Directory	3/11/2021 10:38:15 AM
\$RmMetadata	1	Regular File	3/11/2021 10:38:15 AM
\$ObjId	1	Regular File	3/11/2021 10:38:15 AM
\$Quota	1	Regular File	3/11/2021 10:38:15 AM
\$Reparse	1	Regular File	3/11/2021 10:38:15 AM
\$UsnJrnl	0	Regular File	3/17/2021 2:56:46 PM

Custom Content Sources

Evidence:File System|Path|File Options

New Edit Remove Remove All Create Image

Properties | Hex Value Interpreter | Custom Content Sources

For User Guide, press F1

FTK Imager

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

Evidence Tree

- [root]
  - \$BadClus
  - \$Extend
  - \$Deleted
  - \$Old
  - \$Reparse
  - \$RmMetadata
  - \$UsnJrnl
  - \$Recycle.Bin
  - \$Secure
  - \$UCase
  - Boot
  - document
  - Documents and Settings
  - EFI
  - metasploit
  - PerfLogs

File List

Name	Size	Type	Date Modified
\$J	690,516	Alternate Data Stream	3/17/2021 2:56:46 PM
\$J.FileSlack	173	File Slack	
\$Max	1	Alternate Data Stream	3/17/2021 2:56:46 PM

Custom Content Sources

Evidence:File System|Path|File Options

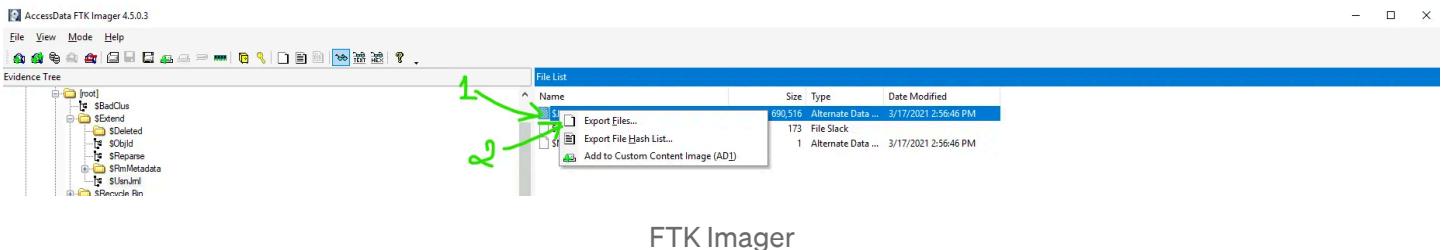
New Edit Remove Remove All Create Image

Properties | Hex Value Interpreter | Custom Content Sources

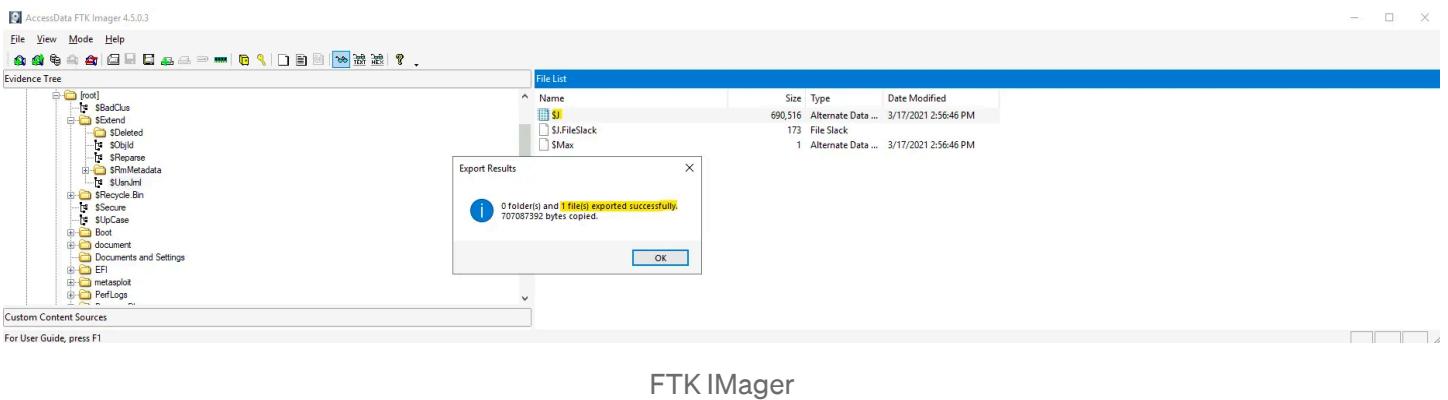
For User Guide, press F1

FTK Imager

- Right-clicked \$J.
- Clicked Export Files ... .



- Selected c:\Users\Administrator\Desktop\Evidence\\$\\$J .
- Clicked OK .



- Ran the provided command line in this task, and a USNJournal.csv file was generated in the Evidence folder.

```
Administrator: Command Prompt
C:\Users\Administrator\Desktop\EZ tools\MFTECmd.exe -f ..\Evidence\$\$J --csv ..\Evidence --csvf USNJournal.csv
MFTECmd version 1.2.2.1
Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/MFTECmd
Command line: -f ..\Evidence\$\$J --csv ..\Evidence --csvf USNJournal.csv
File type: UsnJournal
Processed ..\Evidence\$\$J in 0.6469 seconds
Usn entries found in ..\Evidence\$\$J: 366,559
CSV output will be saved to ..\Evidence\USNJournal.csv
C:\Users\Administrator\Desktop\EZ tools>
```

Command Prompt

- Opened USNJournal.csv using Timeline Explorer .
- Typed Program Files , and clicked Find .
- Dragged the Entry Number column header, than the Name column header to group.

Line	Tag	Update Timestamp	Parent Path	Extension	Sequence Number	Parent Entry Number	Parent Sequence Number	Update Sequence Number	Update Reasons
>	Entry Number:	92716 (Count=1)							
>	Entry Number:	92942 (Count=1)							
>	Entry Number:	94309 (Count=4)							
>	Entry Number:	94968 (Count=1)							
>	Entry Number:	95071 (Count=73)							
>	Name:	AppCache133796450244092699.txt~RF1a1a9b9e.TMP (Count=3)							
>	Name:	Apps.index (Count=4)							
>	Name:	cachev3.dat (Count=5)							
>	Name:	Microsoft_AutoGenerated_(8ABD94FB-E7D6-84A6-A997-C918EDDE0AE5) (Count=4)							
>	Name:	New Text Document.txt (Count=3)							
322..		2025-01-15 08:09:38		.txt	15	93601		2	702633456 FileCreate
322..		2025-01-15 08:09:38		.txt	15	93601		2	702633560 FileCreate Close
322..		2025-01-15 08:09:43		.txt	15	93601		2	702633664 RenameOldName
>	Name:	secret_code.txt (Count=7)							
>	Name:	SRU.log (Count=10)							
>	Name:	SRU006R3.lne (Count=3)							

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.

Line	Tag	Update Timestamp	Parent Path	Extension	Sequence Number	Parent Entry Number	Parent Sequence Number	Update Sequence Number	Update Reasons
>	Entry Number:	92716 (Count=1)							
>	Entry Number:	92942 (Count=1)							
>	Entry Number:	94309 (Count=4)							
>	Entry Number:	94968 (Count=1)							
>	Entry Number:	95071 (Count=73)							
>	Name:	AppCache133796450244092699.txt~RF1a1a9b9e.TMP (Count=3)							
>	Name:	Apps.index (Count=4)							
>	Name:	cachev3.dat (Count=5)							
>	Name:	Microsoft_AutoGenerated_(8ABD94FB-E7D6-84A6-A997-C918EDDE0AE5) (Count=4)							
>	Name:	New Text Document.txt (Count=3)							
322..		2025-01-15 08:09:38		.txt	15	93601		2	702633456 FileCreate
322..		2025-01-15 08:09:38		.txt	15	93601		2	702633560 FileCreate Close
322..		2025-01-15 08:09:43		.txt	15	93601		2	702633664 RenameOldName
>	Name:	secret_code.txt (Count=7)							
>	Name:	SRU.log (Count=10)							
>	Name:	SRU006R3.lne (Count=3)							

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 v2.0.0.1									
File Tools Tabs View Help									
USN0nI.csv									
Entry Number ▲ Name ▲							secret_code.txt x ▾ Find		
Line	Tag	Update Timestamp	Parent Path	Extension	Sequence Number	Parent Entry Number	Parent Sequence Number	Update Sequence Number	Update Reasons
Y =	=	=	=	=	=	=	=	=	=
Entry Number: 95071 (Count=7)									
Name: secret_code.txt (Count=7)									

### Timeline Explorer

- Clicked expanding `secret_code.txt`'s results, and analyzed the Update Reasons .

Timeline Explorer v2.0.0.1									
File Tools Tabs View Help									
USN0nI.csv									
Entry Number ▲ Name ▲							secret_code.txt x ▾ Find		
Line	Tag	Update Timestamp	Parent Path	Extension	Sequence Number	Parent Entry Number	Parent Sequence Number	Update Sequence Number	Update Reasons
Y =	=	=	=	=	=	=	=	=	=
Entry Number: 95071 (Count=7)									
Name: secret_code.txt (Count=7)									
322..	□	2025-01-15 08:09:43		.txt	15	93601	2	702633768 RenameNewName	
322..	□	2025-01-15 08:09:43		.txt	15	93601	2	702633864 RenameNewName Close	
322..	□	2025-01-15 08:09:43		.txt	15	93601	2	702634328 ObjectIdChange	
322..	□	2025-01-15 08:09:43		.txt	15	93601	2	702634424 ObjectIdChange Close	
322..	□	2025-01-15 08:09:56		.txt	15	93601	2	702642328 DataExtend	
322..	□	2025-01-15 08:09:56		.txt	15	93601	2	702642424 DataExtend Close	
322..	□	2025-01-15 08:09:56		.txt	15	93601	2	702642520 FileDelete Close	

### Timeline Explorer

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

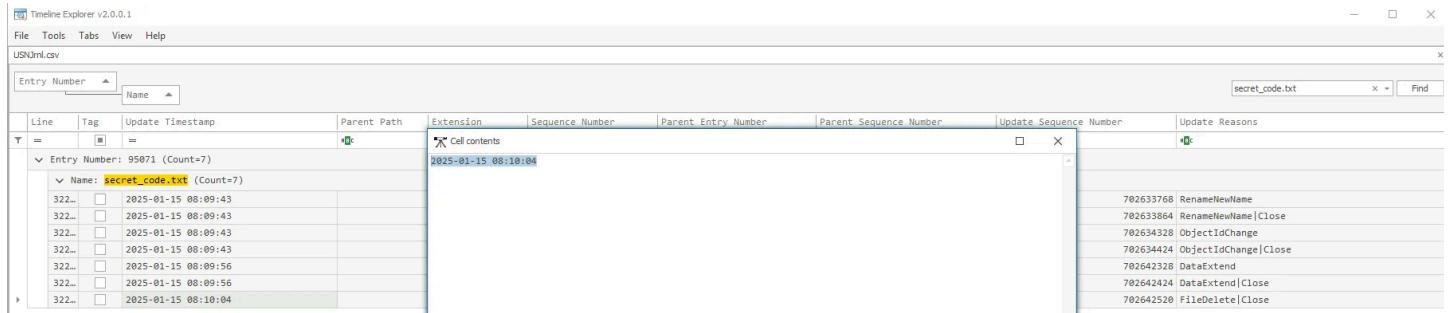
2025-01-15 08:10:04

- Discovered the answer based on 6.3.

Timeline Explorer v2.0.0.1									
File Tools Tabs View Help									
USN0nI.csv									
Entry Number ▲ Name ▲							secret_code.txt x ▾ Find		
Line	Tag	Update Timestamp	Parent Path	Extension	Sequence Number	Parent Entry Number	Parent Sequence Number	Update Sequence Number	Update Reasons
Y =	=	=	=	=	=	=	=	=	=
Entry Number: 95071 (Count=7)									
Name: secret_code.txt (Count=7)									
322..	□	2025-01-15 08:09:43		.txt	15	93601	2	702633768 RenameNewName	
322..	□	2025-01-15 08:09:43		.txt	15	93601	2	702633864 RenameNewName Close	
322..	□	2025-01-15 08:09:43		.txt	15	93601	2	702634328 ObjectIdChange	
322..	□	2025-01-15 08:09:43		.txt	15	93601	2	702634424 ObjectIdChange Close	
322..	□	2025-01-15 08:09:56		.txt	15	93601	2	702642328 DataExtend	
322..	□	2025-01-15 08:09:56		.txt	15	93601	2	702642424 DataExtend Close	
322..	□	2025-01-15 08:09:56		.txt	15	93601	2	702642520 FileDelete Close	
		2025-01-15 08:10:04							

### Timeline Explorer

- Doubled-clicked 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:

The screenshot shows the EnCase Evidence File Explorer interface. On the left, the Evidence Tree pane displays the file system structure, including the metasploit directory. The right pane, titled 'File List', shows a table of files. One file, '\$I30', is highlighted with a red cross icon and a size of 12. A yellow circle labeled '3' points to this entry. Another yellow circle labeled '2' points to a file entry with a red cross in the list.

Name	Size	Type	Date Modified
\$Extend	1	Directory	3/4/2024 1:53:08 PM
\$Recycle.Bin	1	Directory	3/4/2024 1:53:08 PM
\$Secure			
\$UpCase			
Boot			
document			
Documents and Settings			
EFI			
metasploit			
apps			
java			
nginx			
nmap			
postgresql			
ruby			
scripts			
tools			
\$I30	12	NTFS INDEX Entry	3/4/2024 1:53:08 PM
apps	1	Regular File	2/15/2024 11:48:58 AM
backup.bat	1	Regular File	3/4/2024 1:23:04 PM
console.bat	1	Regular File	2/15/2024 11:48:58 AM
createuser.bat	1	Regular File	2/15/2024 11:48:58 AM
deleteuser.bat	1	Regular File	2/15/2024 11:48:58 AM
dev_msconsole.bat	1	Regular File	3/4/2024 1:23:04 PM
dev_msfir.bat	1	Regular File	3/4/2024 1:23:04 PM
dev_msupdate.bat	1	Regular File	2/15/2024 11:48:58 AM
dev_shell.bat	1	Regular File	3/4/2024 1:23:04 PM
diaglog.bat	1	Regular File	3/4/2024 1:23:04 PM

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:

The screenshot shows the Windows File Explorer interface. The address bar indicates the path: This PC > Local Disk (C:) > metasploit. The left sidebar shows quick access links like Desktop, Documents, Downloads, Pictures, EZ tools, Local Disk (C:), and secret\_directory. The main pane lists the contents of the metasploit directory, which include four subfolders: apps, nginx, postgresql, and ruby. A yellow box highlights the list of files.

Name	Date modified	Type
apps	3/4/2024 1:17 PM	File folder
nginx	3/4/2024 1:25 PM	File folder
postgresql	3/4/2024 1:53 PM	File folder
ruby	3/4/2024 1:53 PM	File folder

This image and all the theoretical content of the present article is [TryHackMe](#)'s property.

## Forensics Value

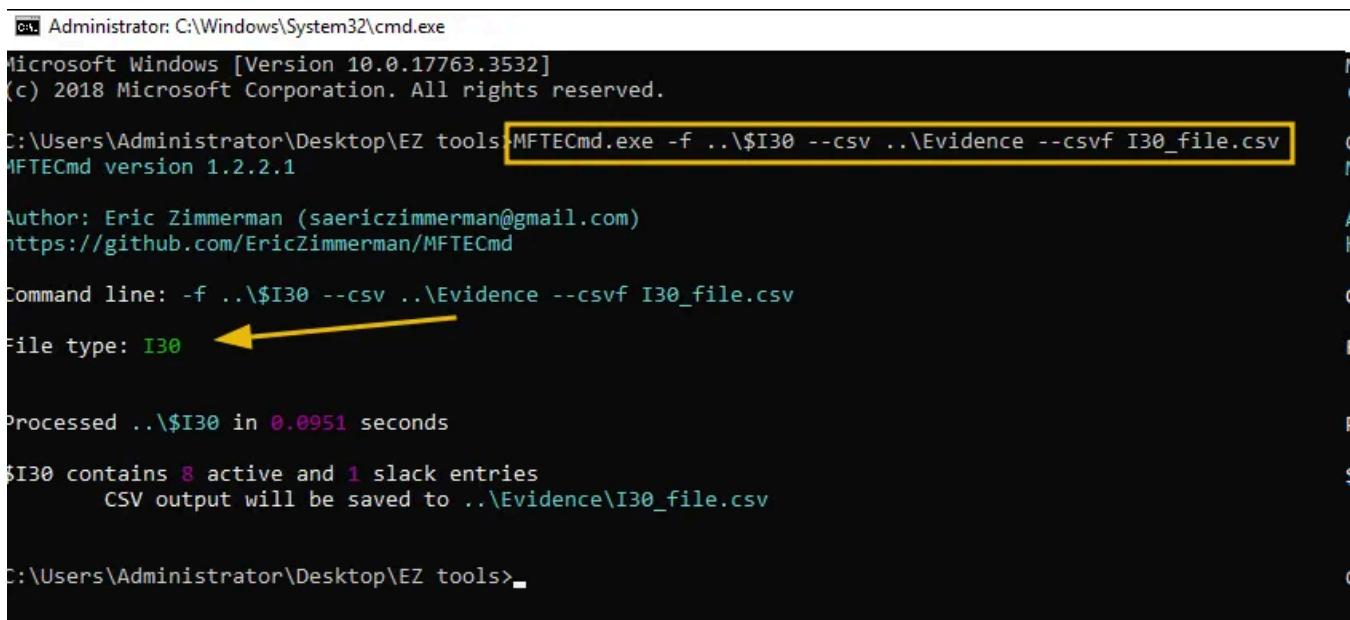
Now that we understand the \$I30 file and its purpose, it's important to understand its forensics value. Some of the key forensics values we can get from this are:

- **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 present 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



```

Administrator: C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.17763.3532]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator\Desktop\EZ tools>MFTECmd.exe -f ..\$I30 --csv ..\Evidence --csvf I30_file.csv
MFTECmd version 1.2.2.1

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

Command line: -f ..\$I30 --csv ..\Evidence --csvf I30_file.csv
File type: I30 ←

Processed ..\$I30 in 0.0951 seconds
$I30 contains 8 active and 1 slack entries
    CSV output will be saved to ..\Evidence\I30_file.csv

C:\Users\Administrator\Desktop\EZ tools>

```

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:

Line	Tag	Offset	From Slack	Self Mft Entry	Self Mft Sequence	File Name	Flags	Name Type	Parent Mft Entry	Parent Mft Sequence	Created On
36	<input type="checkbox"/>	6808	True			scripts	IsDirectory DosWindows	S12386	9		2024-03-0
37	<input type="checkbox"/>	6904	True			serviceinstall.bat	Archive Windows	S12386	9		2024-03-0
38	<input checked="" type="checkbox"/>	7024	True			servicerun.bat	Archive Windows	S12386	9		2024-03-0
39	<input checked="" type="checkbox"/>	7136	True			SERVIC~1.BAT	Archive Dos	S12386	9		2024-03-0
40	<input checked="" type="checkbox"/>	7248	True			SERVIC~2.BAT	Archive Dos	S12386	9		2024-03-0
41	<input type="checkbox"/>	7360	True			tools	IsDirectory DosWindows	S12386	9		2024-03-0
42	<input type="checkbox"/>	7456	True			uninstall.dat	Archive Windows	S12386	9		2024-03-0
43	<input type="checkbox"/>	7568	True			uninstall.dat.new	Archive Windows	S12386	9		2024-03-0
44	<input type="checkbox"/>	7688	True			uninstall.exe	Archive Windows	S12386	9		2024-03-0
45	<input type="checkbox"/>	7800	True			uninstbr.000	Archive DosWindows	S12386	9		2024-03-0
46	<input type="checkbox"/>	7912	True			UNINST~1.DAT	Archive Dos	S12386	9		2024-03-0
47	<input type="checkbox"/>	8024	True			UNINST~1.EXE	Archive Dos	S12386	9		2024-03-0
48	<input type="checkbox"/>	8256	False	554188	2	POSTGR~1	IsDirectory Dos	S12386	9		2024-03-0
49	<input type="checkbox"/>	8360	False	513538	3	ruby	IsDirectory DosWindows	S12386	9		2024-03-0
50	<input type="checkbox"/>	8456	True			tools	IsDirectory DosWindows	S12386	9		2024-03-0
51	<input type="checkbox"/>	8648	True			uninstall.exe	Archive Windows	S12386	9		2024-03-0
52	<input type="checkbox"/>	8760	True			UNINST~1.EXE	Archive Dos	S12386	9		2024-03-0
53	<input type="checkbox"/>	9544	True			UNINST~1.EXE	Archive Dos	33554448	0		2024-03-0
54	<input type="checkbox"/>	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?

- Clicked metasploit folder which is part of root .

Evidence Tree

File List

Name	Size	Type	Date Modified
apps	1	Directory	3/4/2024 1:17:37 PM
java	1	Directory	3/4/2024 1:53:01 PM
nginx	1	Directory	3/4/2024 1:25:06 PM
nmap	1	Directory	3/4/2024 1:53:02 PM
postgresql	1	Directory	3/4/2024 1:53:03 PM
ruby	1	Directory	3/4/2024 1:53:03 PM
scripts	1	Directory	3/4/2024 1:53:08 PM
tools	1	Directory	3/4/2024 1:53:08 PM
\$130	12	NTFS Index All...	3/4/2024 1:53:08 PM
apps	\$130.INDX Entry		
backup.bat	1	Regular File	2/15/2024 11:48:58 AM
console.bat	1	Regular File	3/4/2024 1:23:04 PM
createuser.bat	1	Regular File	2/15/2024 11:48:58 AM
deleteruser.bat	1	Regular File	2/15/2024 11:48:58 AM
dev_msfcconsole.bat	1	Regular File	3/4/2024 1:23:04 PM
dev_msfbat.bat	1	Regular File	3/4/2024 1:23:04 PM
dev_msfpupdate.bat	1	Regular File	2/15/2024 11:48:58 AM
dev_shell.bat	1	Regular File	3/4/2024 1:23:04 PM
diaglog.bat	1	Regular File	3/4/2024 1:23:04 PM
diagnostic_shell.bat	1	Regular File	3/4/2024 1:23:04 PM
msfbinscan.bat	1	Regular File	3/4/2024 1:23:04 PM
msfupdate.bat	1	Regular File	2/15/2024 1:23:04 PM

- Right-clicked \$130 .
- Clicked Export Files ... .
- Selected Evidence .
- Clicked OK .

Evidence Tree

File List

Name	Size	Type	Date Modified
scripts	1	Directory	3/4/2024 1:53:08 PM
tools	1	Directory	3/4/2024 1:53:08 PM
\$130	12	NTFS Index All...	3/4/2024 1:53:08 PM
apps	\$130.INDX Entry		
backup.bat	1	Regular File	2/15/2024 11:48:58 AM

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

```
C:\Users\Administrator\Desktop\EZ tools>MFTEcmd.exe -f C:\Users\Administrator\Desktop\Evidence\$130 --csv C:\Users\Administrator\Desktop\Evidence --csvf outcome.csv
MFTECmd version 1.2.2.1

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

Command line: -f C:\Users\Administrator\Desktop\Evidence\$130 --csv C:\Users\Administrator\Desktop\Evidence --csvf outcome.csv
File type: I30

Processed C:\Users\Administrator\Desktop\Evidence\$130 in 0.0356 seconds
$130 contains 4 active and 52 slack entries
CSV output will be saved to C:\Users\Administrator\Desktop\Evidence\outcome.csv

C:\Users\Administrator\Desktop\EZ tools>
```

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.

Line	Tag	Offset	Self Mft Entry	Self Mft Sequence	File Name	Flags	Name Type	Parent Mft Entry	Parent Mft Sequence	Created On	Content Modified On
>											
> From Slack: False	(Count: 4)										

> From Slack: True	(Count: 52)
--------------------	-------------

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.

Line	Tag	Offset	Self Mft Entry	Self Mft Sequence	File Name	Flags	Name Type	Parent Mft Sequence	Created On	Content Modified On	Record Modified On	Last Accessed
>												
> Parent Mft Entry: 2386296948	(Count: 1)											
> Parent Mft Entry: 33554448	(Count: 4)											
> Parent Mft Entry: 512386	(Count: 46)											
1	□	64			apps	IsDirectory	DosWindows	9	2024-03-04 13:17:37.5345288	2024-03-04 13:17:37.5345288	2024-03-04 13:27:18.6267972	2024-03-04 1
2	□	160			diagnostic_shell.bat	Archive	Windows	9	2024-03-04 13:23:04.0847800	2024-03-04 13:23:04.1160319	2024-03-04 13:27:18.6267972	2024-03-04 1
6	□	1960			DIAGNO~1.BAT	Archive	Dos	9	2024-03-04 13:23:04.0847800	2024-03-04 13:23:04.1160319	2024-03-04 13:23:04.1160319	2024-03-04 1
7	□	2072			java	IsDirectory	DosWindows	9	2024-03-04 13:19:37.6777046	2024-03-04 13:19:39.3808134	2024-03-04 13:19:39.3808134	2024-03-04 1
8	□	2168			john	IsDirectory	DosWindows	9	2024-03-04 13:17:44.8469327	2024-03-04 13:17:44.8625796	2024-03-04 13:17:44.8625796	2024-03-04 1
9	□	2264			licenses	IsDirectory	DosWindows	9	2024-03-04 13:17:44.8801022	2024-03-04 13:17:44.8812966	2024-03-04 13:17:44.8812966	2024-03-04 1
18	□	2368			msfbinscan.bat	Archive	Windows	9	2024-03-04 13:23:04.1472812	2024-03-04 13:23:04.1160319	2024-03-04 13:23:04.1160319	2024-03-04 1
11	□	2480			MSFBIN~1.BAT	Archive	Dos	9	2024-03-04 13:23:04.1472812	2024-03-04 13:23:04.1160319	2024-03-04 13:23:04.1160319	2024-03-04 1
12	□	2592			msfd.bat	Archive	DosWindows	9	2024-03-04 13:23:04.1629171	2024-03-04 13:23:04.1160319	2024-03-04 13:23:04.1160319	2024-03-04 1
13	□	2696			msfelfscan.bat	Archive	Windows	9	2024-03-04 13:23:04.1785517	2024-03-04 13:23:04.1160319	2024-03-04 13:23:04.1160319	2024-03-04 1
14	□	2808			MSFELF~1.BAT	Archive	Dos	9	2024-03-04 13:23:04.1785517	2024-03-04 13:23:04.1160319	2024-03-04 13:23:04.1160319	2024-03-04 1
15	□	2920			msfmachscan.bat	Archive	Windows	9	2024-03-04 13:23:04.1785517	2024-03-04 13:23:04.1160319	2024-03-04 13:23:04.1160319	2024-03-04 1
16	□	3032			MSFMAC~1.BAT	Archive	Dos	9	2024-03-04 13:23:04.1785517	2024-03-04 13:23:04.1160319	2024-03-04 13:23:04.1160319	2024-03-04 1
17	□	3144			nginx	IsDirectory	DosWindows	9	2024-03-04 13:17:52.2062398	2024-03-04 13:17:52.2374799	2024-03-04 13:17:52.2374799	2024-03-04 1
18	□	3240			nmap	IsDirectory	DosWindows	9	2024-03-04 13:18:22.8291944	2024-03-04 13:18:23.7052288	2024-03-04 13:18:23.7052288	2024-03-04 1
19	□	3336			postgresql	IsDirectory	Windows	9	2024-03-04 13:18:27.0177001	2024-03-04 13:18:27.1114440	2024-03-04 13:18:27.1114440	2024-03-04 1
20	□	3440			POSTGR~1	IsDirectory	Dos	9	2024-03-04 13:18:27.0177001	2024-03-04 13:18:27.1114440	2024-03-04 13:18:27.1114440	2024-03-04 1
21	□	3544			resetpw.bat	Archive	DosWindows	9	2024-03-04 13:23:04.0847800	2024-02-15 11:48:58.0000000	2024-03-04 13:23:04.0847800	2024-03-04 1
22	□	3648			restore.bat	Archive	DosWindows	9	2024-03-04 13:23:04.1004247	2024-02-15 11:48:58.0000000	2024-03-04 13:23:04.1004247	2024-03-04 1

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 andLogFile
- Understand the importance of timestamps.

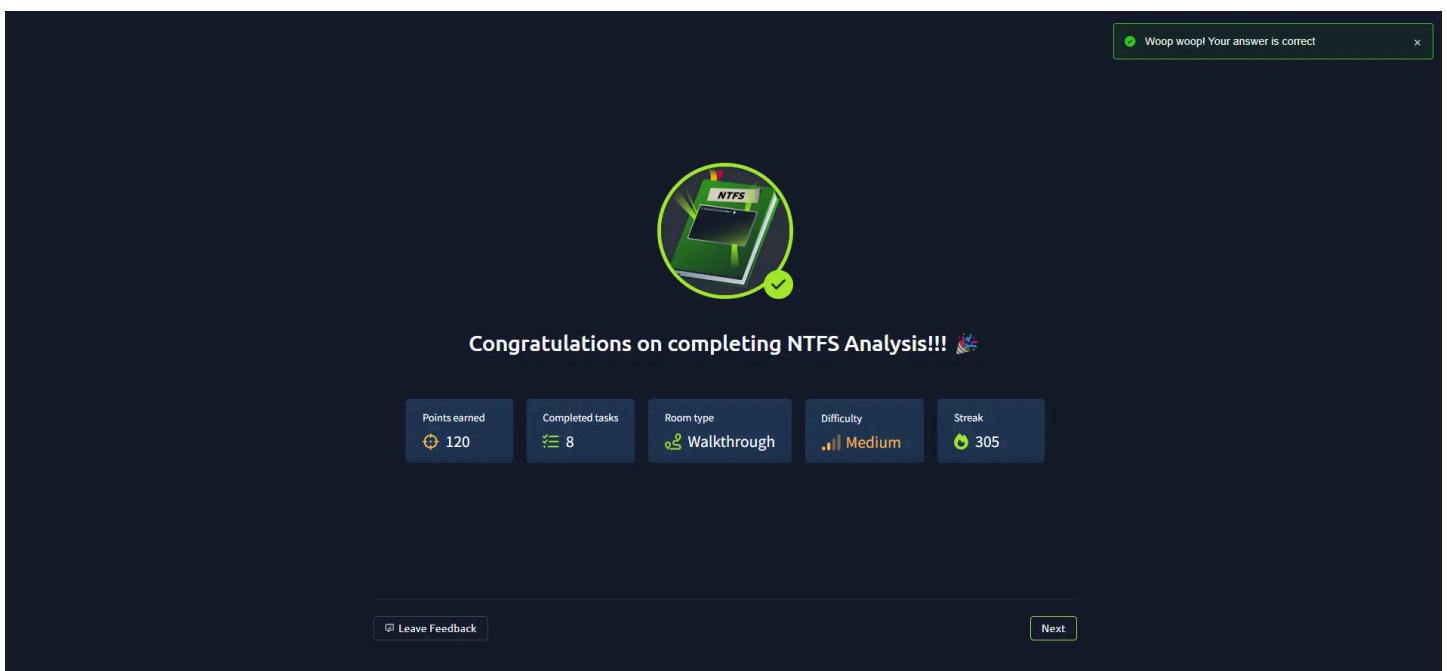
More walkthroughs and challenge rooms on File system Analysis are in the pipeline.  
Keep an eye on the socials for more updates.

*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.

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:	▪	368 <sup>th</sup> all time	▪	414 <sup>th</sup> monthly		
Brazil ranking:	▪	8 <sup>th</sup> all time	▪	7 <sup>th</sup> monthly		

Global all time ranking: 368<sup>th</sup>

Rosana [0x11][VANGUARD]

Open to Work | 13 completed learning paths: SOC Level 2, Cyber Defense . Web Application Penetration . Jr Penetration Tester . Security Engineer . SOC Level 1 . DevSecOps . CompTIA Pentest+ . Cyber Security 101 . Pre Security . Web Fundamentals . Intro to Cyber Security . Complete Beginner

Rank **368**      Badges **59**  
Streak **305**      Completed rooms **606**

Completed rooms      Certificates      Badges      Created rooms      Yearly activity      Tickets

Yearly activity

Key: No activity (Grey), 1 event (Yellow), 2 events (Green), ≥ 3 events (Dark Green)

Total events this year: **2025**

Activity events are measured by the number of machines started, questions answered or file downloads.

This image and all the theoretical content of the present article is [TryHackMe](#) 's property.

Brazil all time ranking: 8<sup>th</sup>

Welcome to the wall of fame - Here are our top 50 users.

General King of the Hill All time Monthly Brazil You are rank #8

Rank	Username	Country	Points	Rooms
1	Y4m4t0 [0x14][MYTHIC]	Country: 🇺🇸	139768	Rooms in: 1000
2	cyberguia [0x14][MYTHIC]	Country: 🇧🇷	133136	Rooms in: 981
3	br0nx [0x12][SHOGUN]	Country: 🇧🇷	107802	Rooms in: 773
4	Sys4n0m4ly [0x12][SHOGUN]	Country: 🇧🇷	103781	Rooms in: 556
5	h0ru [0x12][SHOGUN]	Country: 🇧🇷	99864	Rooms in: 818
6	3ddie [0x11][VANGUARD]	Country: 🇧🇷	94188	Rooms: 665
7	OMC.Br [0x11][VANGUARD]	Country: 🇧🇷	93106	Rooms: 718
8	Rosana [0x11][VANGUARD]	Country: 🇧🇷	85597	Rooms: 634

This image and all the theoretical content of the present article is [TryHackMe](#) 's property.

Global monthly ranking: 414<sup>th</sup>

Welcome to the wall of fame - Here are our top 50 users.

Rank	Username	Country	Points	Rooms
6	Anonymous1947 [0x9][MAGE]	INDIA	2230	83
7	DBTHMETU [0x10][SAGE]	EGYPT	1824	461
8	ROBIN [0x9][MAGE]	EGYPT	1821	93

This image and all the theoretical content of the present article is [TryHackMe](#)'s property.

Brazil monthly ranking: 7<sup>th</sup>

Welcome to the wall of fame - Here are our top 50 users.

Rank	Username	Country	Points	Rooms
6	S3rphiel [0xA][WIZARD]	BRAZIL	393	126
7	Rosana [0x11][VANGUARD]	BRAZIL	370	634
8	D3Z33 [0x9][MAGE]	BRAZIL	357	87

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

# Windows Logging for SOC

SOC • DFIR • Windows log analysis

Rosana

 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

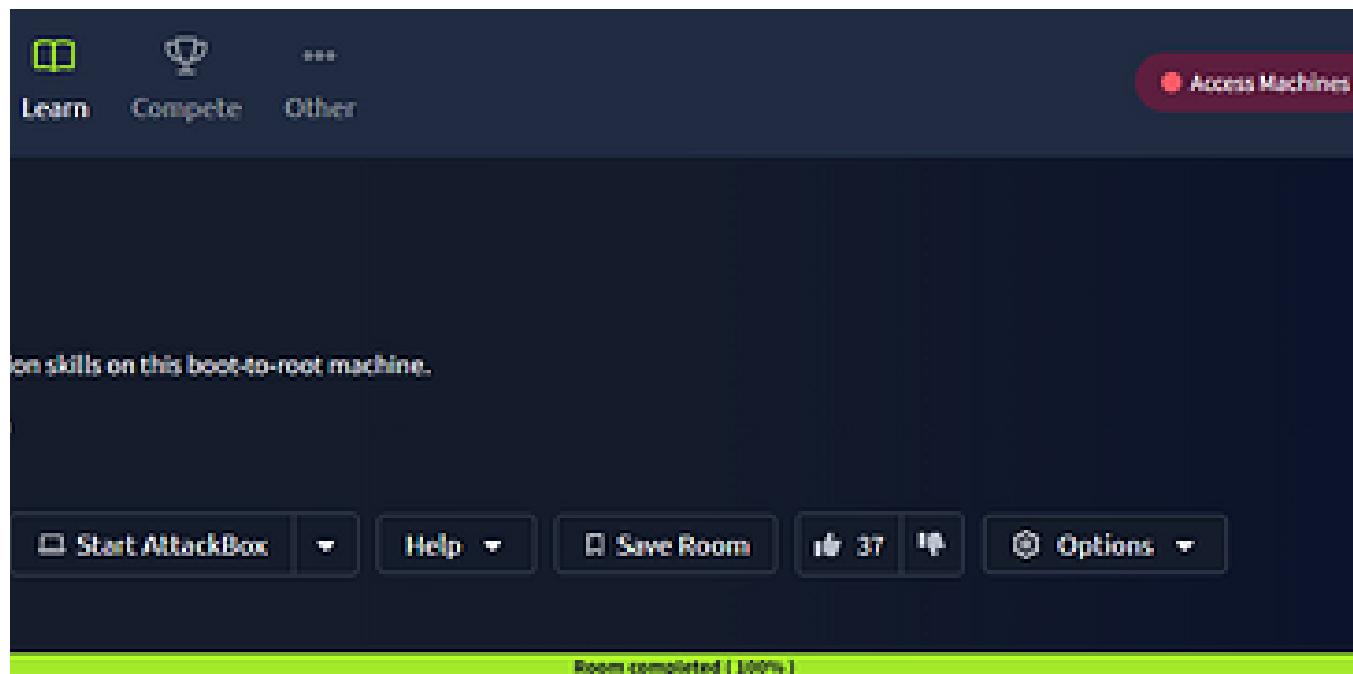


Task 1	Introduction
Task 2	What is Burp Suite
Task 3	Features of Burp Community
Task 4	Installation
Task 5	The Dashboard
Task 6	Navigation
Task 7	Options
Task 8	Introduction to the Burp Proxy
Task 9	Connecting through the Proxy (FoxyProxy)
Task 10	Site Map and Issue Definitions
Task 11	The Burp Suite Browser
Task 12	Scoping and Targeting

 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

The screenshot shows the TryHackMe platform interface. At the top, there are navigation icons for 'Learn', 'Compete', 'Other', and a red button labeled 'Access Machines'. Below the navigation bar, a large text area contains the instruction: 'Test your enumeration skills on this boot-to-root machine.' At the bottom of the screen, there is a toolbar with buttons for 'Start AttackBox', 'Help', 'Save Room' (which is highlighted in blue), '37' (likely referring to the number of challenges or rooms), and 'Options'. A progress bar at the very bottom indicates 'Room completed | 100%'.

 RosanaFSS

## Lookup . TryHackMe Walkthrough . Enumeration

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

Nov 24, 2024



Offensive Pentesting learning path > Getting Started module

# Vulnversity

- Active Reconnaissance
  - Privilege Escalation
  - Web Application Attacks



RosanaFSS

Offensive Pentesting . Vulniversity: TryHackMe Walkthrough

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

Jan 7 4



See all from RosanaFSS

## Recommended from Medium

Premium room

an attack from an implanted webshell.

New Help Save Room 3 Options

Room completed ( 100% )

Sle3pyHead 🎖️

## Infinity Shell CTF Notes | TryHackMe

Quick reference notes for TryHackMe's Infinity Shell challenge.

⭐ Jul 11 ⌛ 15



**Hide and Seek**

**Specter**  
**Forensics**

Today

A note was discovered on the compromised system, taunting us. It suggests multiple persistence mechanisms have been implanted, ensuring that Cipher can return whenever he pleases. Here's the note:

*Dear Specter,  
 I must say, it's been a thrill dancing through your systems. You lock the doors; I pick the locks. You set up alarms; I waltz right past them. But today, my dear adversary, I've left you a little game.  
 I've sprinkled a few persistence implants across your system, like digital Easter eggs, and I'm giving you a sporting chance to find them. Each one has a clue because where's the fun in a silent hack?*

- Time is on my side, always running like clockwork.
- A secret handshake gets me in every time.
- Whenever you set the stage, I make my entrance.
- I run with the big dogs, booting up alongside the system.
- I love welcome messages.

*Find them all, and you might earn a little respect. Miss one, and well... let's say I'll be back before you even realize I never left. Happy hunting, Specter. May the best ghost win.*

- Cipher

Easy Forensics

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



# Investigating Windows 3.x



MITRE ATT&CK • [Process Monitor \(Procmon\)](#) • [Event Viewer](#)  
[Autoruns](#) • [Registry Editor](#) • [CyberChef](#)

Rosana

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



See more recommendations