

Digital Forensics: Repairing a Damaged Hard Drive and Extracting the Data

 hackers-arise.com/digital-forensics-repairing-a-damaged-hard-drive-and-extracting-the-data

Co11ateral

November 10, 2025



Welcome back, aspiring digital forensic analysts!

There are times when our work requires repairing damaged disks to perform a proper forensic analysis. Attackers use a range of techniques to cover their tracks. These can be corrupting the boot sector, overwriting metadata, physically damaging a drive, or exposing hardware to high heat. That's what they did in Mr.Robot.



Physical damage often destroys data beyond practical recovery, but a much more common tactic is logical sabotage. Attackers wipe partitions, corrupt the Master Boot Record, or otherwise tamper with the file system to slow or confuse investigators. Most real-world incidents that require disk-level recovery come from remote activity rather than physical tampering, unless the case involves an insider with physical access to servers or workstations.

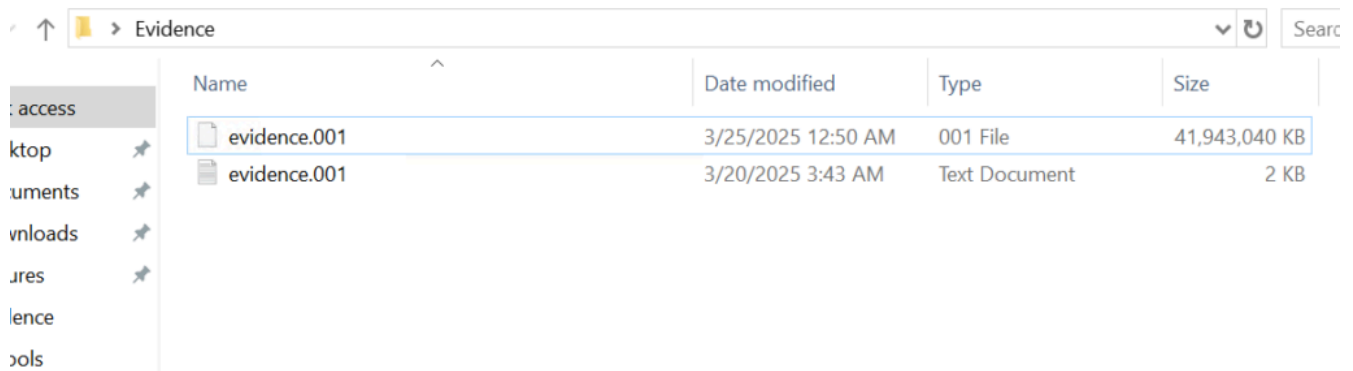
Inexperienced administrators sometimes assume that data becomes irrecoverable after tampering, or that simply deleting files destroys their content and structure. That is not true. In this article we will examine how disks can be repaired and how deleted files can still be discovered and analysed.

In our previous article, **PowerShell for Hackers: Mayhem Edition**, we showed how an attacker can overwrite the MBR and render Windows unbootable. Today we will examine an image with a deliberately damaged boot sector. The machine that produced the image was used for data exfiltration. An insider opened an important PDF that contained a canary token and that token notified the owner that the document had been opened. It also showed the host that was used to access the file. Everything else is unknown and we will work through the evidence together.

Fixing the Drive

Corrupting the disk boot sector is straightforward in principle. You alter the data the system expects to find there so the OS cannot load the disk in the normal way. File formats, executables, archives, images and other files have internal headers and structures that tell software how to interpret their contents. Changing a file extension does not change those internal headers, so renaming alone is a poor method of concealment. Tools that inspect file headers and signatures will still identify the real file type. Users sometimes try to hide VeraCrypt containers by renaming them to appear as ordinary executables. Forensic tools and signature scanners will still flag such anomalies. Windows also leaves numerous artefacts that can indicate which files were opened. Among them are MRU lists, Jump Lists, Recent Items and other traces created by common applications, including simple editors.

Before we continue, let's see what evidence we were given.



Name	Date modified	Type	Size
evidence.001	3/25/2025 12:50 AM	001 File	41,943,040 KB
evidence.001	3/20/2025 3:43 AM	Text Document	2 KB

Above is a forensic image and below is a text file with metadata about that image. As a forensic analyst you should verify the integrity of the evidence by comparing the computed hash of the image with the hash recorded in the metadata file.

evidence.001 - Notepad

File Edit Format View Help

Acquired using: ADI3.1.2.0
Case Number: 1
Evidence Number:
Unique Description:
Examiner:
Notes:

Information for E:\image\evidence:

Physical Evidentiary Item (Source) Information:

[Device Info]

Source Type: Physical

[Drive Geometry]

Cylinders: 5,221

Tracks per Cylinder: 255

Sectors per Track: 63

Bytes per Sector: 512

Sector Count: 83,886,080

[Physical Drive Information]

Drive Model: NVMe Amazon Elastic B SCSI Disk Device

Drive Serial Number: vol081f7d4eddf5bdc72_00000001.

Drive Interface Type: SCSI

Removable drive: False

Source data size: 40960 MB

Sector count: 83886080

[Computed Hashes]

MD5 checksum: 4e3b6453689669ac99f685ce7ac53101

SHA1 checksum: 24320683719dcfd5a88ec0e40291347fbc78c4cb

If the hash matches, work only on a duplicate and keep the original evidence sealed. Create a verified working copy for all further analysis.

Opening a disk image with a corrupted boot sector in Autopsy or FTK Imager will not succeed, as many of these tools expect a valid partition table and a readable boot sector. In such cases you will need to repair the image manually with a hex editor such as HxD so other tools can parse the structure.

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	33	C0	8E	D0	BC	00	7C	8E	C0	8E	D8	BE	00	7C	BF	00	3ĂŽĐ4. ŽĂŽ0%. ĉ.
000000010	06	B9	00	02	FC	F3	A4	50	68	1C	06	CB	FB	B9	04	00	. ^...üóMPh..ĚŮ^...
000000020	BD	BE	07	80	7E	00	00	7C	0B	0F	85	0E	01	83	C5	10	%%.€~..fĂ.
000000030	E2	F1	CD	18	88	56	00	55	C6	46	11	05	C6	46	10	00	âñí. ^V.UÆF..ÆF..
000000040	B4	41	BB	AA	55	CD	13	5D	72	0F	81	FB	55	AA	75	09	`A»^UÍ. r..úU^u.
000000050	F7	C1	01	00	74	03	FE	46	10	66	60	80	7E	10	00	74	÷Ă..t.þF.f`€~..t
000000060	26	66	68	00	00	00	00	66	FF	76	08	68	00	00	68	00	&fh....fÿv.h..h.
000000070	7C	68	01	00	68	10	00	B4	42	8A	56	00	8B	F4	CD	13	h..h..`BŠV.<óí.
000000080	9F	83	C4	10	9E	EB	14	B8	01	02	BB	00	7C	8A	56	00	ŸfĂ.žě.,... ŠV.
000000090	8A	76	01	8A	4E	02	8A	6E	03	CD	13	66	61	73	1C	FE	Šv.ŠN.Šn.í.fas.p
0000000A0	4E	11	75	0C	80	7E	00	80	0F	84	8A	00	B2	80	EB	84	N.u.€~.€..Š. ^€ē,,
0000000B0	55	32	E4	8A	56	00	CD	13	5D	EB	9E	81	3E	FE	7D	55	U2ăŠV.í. ěž.>þ}U
0000000C0	AA	75	6E	FF	76	00	E8	8D	00	75	17	FA	B0	D1	E6	64	^unÿv.è..u.ú°Ñæd
0000000D0	E8	83	00	B0	DF	E6	60	E8	7C	00	B0	FF	E6	64	E8	75	èf. °Bæ`è . °ÿædèu
0000000E0	00	FB	B8	00	BB	CD	1A	66	23	C0	75	3B	66	81	FB	54	.ú.,>í.f#Ău;f.úT
0000000F0	43	50	41	75	32	81	F9	02	01	72	2C	66	68	07	BB	00	CPAu2.ù..r, fh.».
000000100	00	66	68	00	02	00	00	66	68	08	00	00	00	66	53	66	.fh....fh....fSf
000000110	53	66	55	66	68	00	00	00	66	68	00	7C	00	00	66	66	SfUfh....fh. ..f
000000120	61	68	00	00	07	CD	1A	5A	32	F6	EA	00	7C	00	00	CD	ah...í.Z2ôè. ..í
000000130	18	A0	B7	07	EB	08	A0	B6	07	EB	03	A0	B5	07	32	E4	. . .è. ĩ.è. u.2ă
000000140	05	00	07	8B	F0	AC	3C	00	74	09	BB	07	00	B4	0E	CD	...<ô-<.t.»..`í
000000150	10	EB	F2	F4	EB	FD	2B	C9	E4	64	EB	00	24	02	E0	F8	.èôôéÿ+Ěădē.\$.àø
000000160	24	02	C3	49	6E	76	61	6C	69	64	20	70	61	72	74	69	\$.ĂInvalid parti
000000170	74	69	6F	6E	20	74	61	62	6C	65	00	45	72	72	6F	72	tion table.Error
000000180	20	6C	6F	61	64	69	6E	67	20	6F	70	65	72	61	74	69	loading operati
000000190	6E	67	20	73	79	73	74	65	6D	00	4D	69	73	73	69	6E	ng system.Missin
0000001A0	67	20	6F	70	65	72	61	74	69	6E	67	20	73	79	73	74	g operating syst
0000001B0	65	6D	00	00	00	63	7B	9A	CF	AD	76	E5	00	00	80	20	em...c{ší.vă..€
0000001C0	21	00	07	FE	FF	FF	00	08	00	00	00	70	C7	03	00	FE	!..þÿÿ.....þÇ..þ
0000001D0	FF	FF	0C	FE	FF	FF	00	78	C7	03	00	78	38	01	00	00	ÿÿ.þÿÿ.xÇ..x8...
0000001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000001F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000200	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

The first 512 bytes of a disk image contain the MBR (Master Boot Record) on traditional MBR-partitioned media. In this image the final two bytes of that sector were modified. A valid MBR should end with the boot signature 0x55 0xAA. Those two bytes tell the firmware and many tools that the sector contains a valid boot record. Without the signature the image may be unreadable, so restoring the correct 0x55AA signature is the first step we need to do.

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	33	C0	8E	D0	BC	00	7C	8E	C0	8E	D8	BE	00	7C	BF	00	3ÀŽĐ¼. ŽÀŽĐ¼. ž.
00000010	06	B9	00	02	FC	F3	A4	50	68	1C	06	CB	FB	B9	04	00	. ' . . úó¼Ph. . Ěú¹. .
00000020	BD	BE	07	80	7E	00	00	7C	0B	0F	85	0E	01	83	C5	10	¼¼. Ě~. fĚ.
00000030	E2	F1	CD	18	88	56	00	55	C6	46	11	05	C6	46	10	00	áñÍ. ^V.UĚF. . ĚF. .
00000040	B4	41	BB	AA	55	CD	13	5D	72	0F	81	FB	55	AA	75	09	`A»²UÍ. r. . úU²u.
00000050	F7	C1	01	00	74	03	FE	46	10	66	60	80	7E	10	00	74	÷Á. . t. pF. f`Ě~. . t
00000060	26	66	68	00	00	00	00	66	FF	76	08	68	00	00	68	00	&fh. . . . fyv. h. . h.
00000070	7C	68	01	00	68	10	00	B4	42	8A	56	00	8B	F4	CD	13	h. . h. . `BŠV. <óÍ.
00000080	9F	83	C4	10	9E	EB	14	B8	01	02	BB	00	7C	8A	56	00	ŸfĚ. žė. . . . ». ŠV.
00000090	8A	76	01	8A	4E	02	8A	6E	03	CD	13	66	61	73	1C	FE	Šv. ŠN. Šn. Í. fas. p
000000A0	4E	11	75	0C	80	7E	00	80	0F	84	8A	00	B2	80	EB	84	N. u. Ě~. Ě. . Š. `ĚĚ. .
000000B0	55	32	E4	8A	56	00	CD	13	5D	EB	9E	81	3E	FE	7D	55	U2ăŠV. Í. ěž. >p)U
000000C0	AA	75	6E	FF	76	00	E8	8D	00	75	17	FA	B0	D1	E6	64	*unÿv. ě. . u. ú°Ñăd
000000D0	E8	83	00	B0	DF	E6	60	E8	7C	00	B0	FF	E6	64	E8	75	ěf. °Bæ`ě . °ÿăděu
000000E0	00	FB	B8	00	BB	CD	1A	66	23	C0	75	3B	66	81	FB	54	. ú. . »Í. f#Ěu; f. úT
000000F0	43	50	41	75	32	81	F9	02	01	72	2C	66	68	07	BB	00	CPAu2. ù. . r, fh. ».
00000100	00	66	68	00	02	00	00	66	68	08	00	00	00	66	53	66	. fh. . . . fh. . . . fSf
00000110	53	66	55	66	68	00	00	00	00	66	68	00	7C	00	00	66	SfUfh. . . . fh. . . f
00000120	61	68	00	00	07	CD	1A	5A	32	F6	EA	00	7C	00	00	CD	ah. . . Í. Z2öě. . . Í
00000130	18	A0	B7	07	EB	08	A0	B6	07	EB	03	A0	B5	07	32	E4	. . ě. ¶. ě. µ. 2ă
00000140	05	00	07	8B	F0	AC	3C	00	74	09	BB	07	00	B4	0E	CD	. . . <ð-<. t. ». . `Í
00000150	10	EB	F2	F4	EB	FD	2B	C9	E4	64	EB	00	24	02	E0	F8	. ěòôěÿ+Ěădě. \$. àø
00000160	24	02	C3	49	6E	76	61	6C	69	64	20	70	61	72	74	69	\$. ĚInvalid parti
00000170	74	69	6F	6E	20	74	61	62	6C	65	00	45	72	72	6F	72	tion table. Error
00000180	20	6C	6F	61	64	69	6E	67	20	6F	70	65	72	61	74	69	loading operati
00000190	6E	67	20	73	79	73	74	65	6D	00	4D	69	73	73	69	6E	ng system. Missin
000001A0	67	20	6F	70	65	72	61	74	69	6E	67	20	73	79	73	74	g operating syst
000001B0	65	6D	00	00	00	63	7B	9A	CF	AD	76	E5	00	00	80	20	em. . . c{šĚ. vă. Ě
000001C0	21	00	07	FE	FF	FF	00	08	00	00	00	70	C7	03	00	FE	! . . pÿÿ. pÇ. . p
000001D0	FF	FF	0C	FE	FF	FF	00	78	C7	03	00	78	38	01	00	00	ÿÿ. pÿÿ. xÇ. . x8. . .
000001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000001F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00 55 AA
00000200	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

When editing the MBR in a hex editor, do not delete bytes with backspace, you need to overwrite them. Place the cursor before the bytes to be changed and type the new hex values. The editor will replace the existing bytes without shifting the file.

Partitions

This image contains two partitions. In a hex view you can see the partition table entries that describe those partitions. In forensic viewers such as FTK Imager and Autopsy those partitions will be displayed graphically once the MBR and partition table are valid.

```

000000190 6E 67 20 73 79 73 74 65 6D 00 4D 69 73 73 69 6E ng system.Missin
0000001A0 67 20 6F 70 65 72 61 74 69 6E 67 20 73 79 73 74 g operating syst
0000001B0 65 6D 00 00 00 63 7B 9A CF AD 76 E5 00 00 80 20 em...c{šĩ.vå..€
0000001C0 21 00 07 FE FF FF 00 08 00 00 00 70 C7 03 00 FE !..þÿÿ....pÇ..þ
0000001D0 FF FF 0C FE FF FF 00 78 C7 03 00 78 38 01 00 00 Ÿÿ.þÿÿ.xÇ..x8...
0000001E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000001F0 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA .....Uª
000000200 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

Both of them are in the black frame. The partition table entries also encode the partition size and starting sector in little-endian form, which requires byte-order interpretation and calculation to convert to human-readable sizes. For example, if you see an entry that corresponds to 63,401,984 sectors and each sector is 512 bytes, the size calculation is:

63,401,984 sectors × 512 bytes = 32,461,815,808 bytes, which is 32.46 GB (decimal) or ≈ 30.23 GiB

```

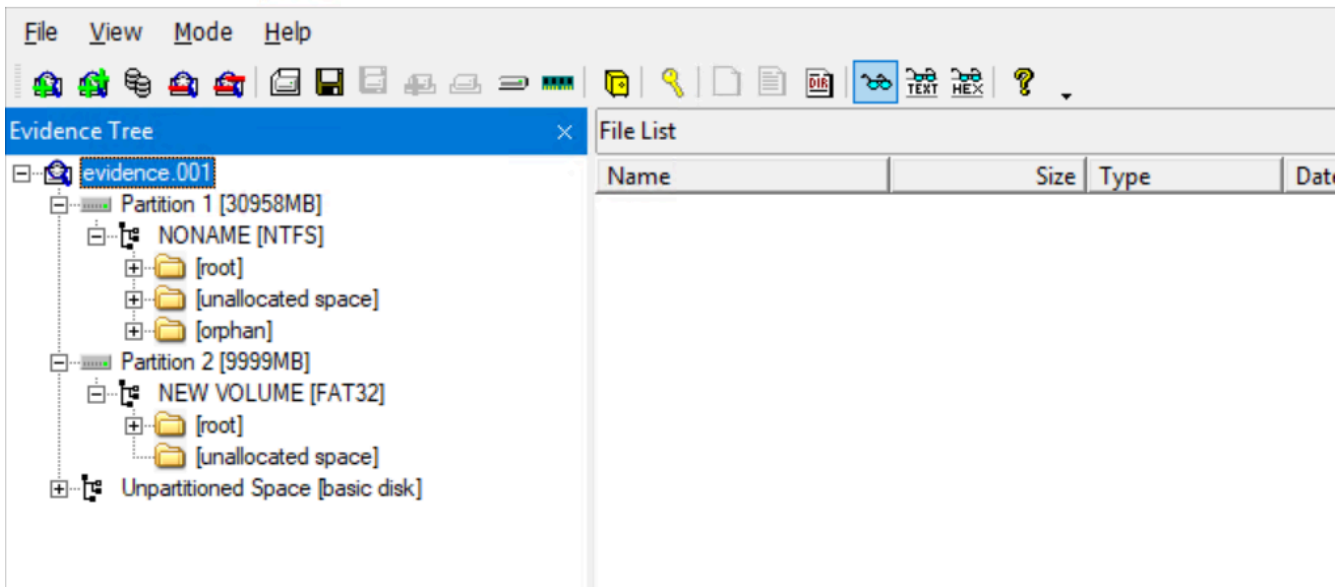
13 66 61 73 1C FE Šv.ŠN.Šn.ĩ.fas.þ
8A 00 B2 80 EB 84 N.u.€~.€.,Š.ª€ª,,
9E 81 3E FE 7D 55 U2aŠV.ĩ.]ěž.>þ)U
17 FA B0 D1 E6 64 *unÿv.è..u.ú°Ñäd
B0 FF E6 64 E8 75 èf.°Bæ`è|.°ÿædèu
75 3B 66 81 FB 54 .ú,.»Í.f#Àu;f.úT
2C 66 68 07 BB 00 CPAu2.ù..r,fh.».
00 00 00 66 53 66 .fh....fh....fSf
68 00 7C 00 00 66 SfUfh....fh.|..f
EA 00 7C 00 00 CD ah...Í.Z2öè.|..Í
03 A0 B5 07 32 E4 . .è. ¶.è. u.2ä
BB 07 00 B4 0E CD ...<ð-<.t.»..`Í
EB 00 24 02 E0 F8 .èðèÿ+Éädè.$.àø
20 70 61 72 74 69 $.ÄInvalid parti
00 45 72 72 6F 72 tion table.Error
70 65 72 61 74 69 loading operati
4D 69 73 73 69 6E ng system.Missin
67 20 73 79 73 74 g operating syst
76 E5 00 00 80 20 em...c{šĩ.vå..€
00 70 C7 03 00 FE !..þÿÿ....pÇ..þ
00 78 38 01 00 00 Ÿÿ.þÿÿ.xÇ..x8...

```

UInt24	go to:	13070336
Int32	go to:	63401984
UInt32	go to:	63401984
Int64	go to:	Invalid
UInt64	go to:	Invalid
LEB128	go to:	0
ULEB128	go to:	0
AnsiChar / char8_t		
WideChar / char16_t		澗
UTF-8 code point		(U+0000)
Single (float32)		1.17218827296
Double (float64)		Invalid
OLETIME		Invalid
FILETIME		Invalid
DOS date		Invalid
DOS time		2:00:00 PM
Byte order		

FTK Imager

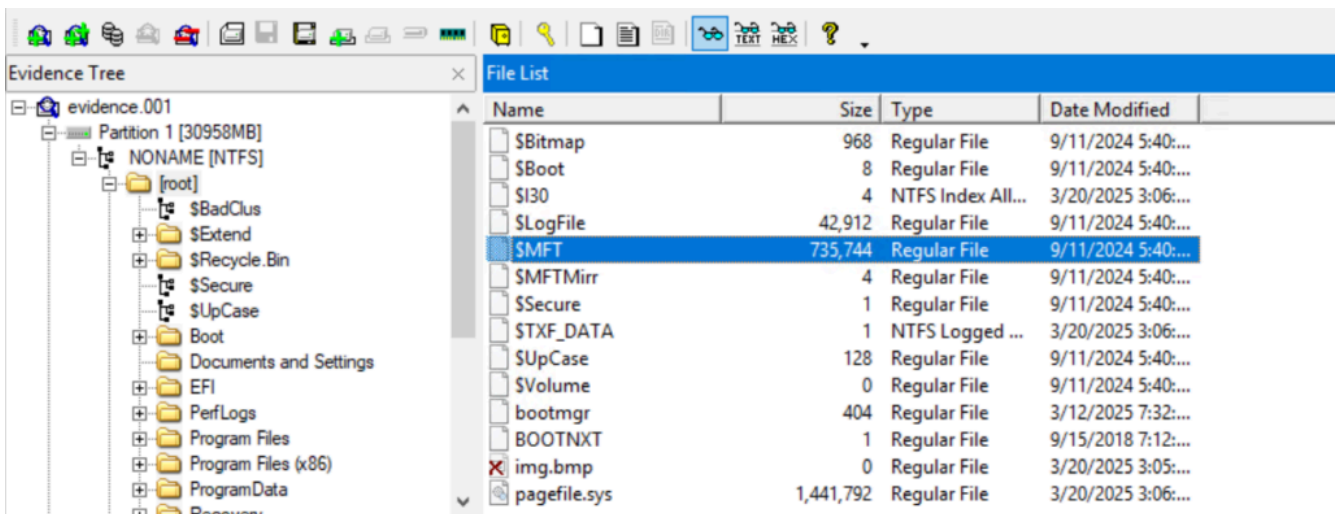
Now let's use FTK Imager to view the contents of our evidence file. In FTK Imager choose **File**, then **Add Evidence Item**, select **Image File**, and point the application to the verified copy of the image.



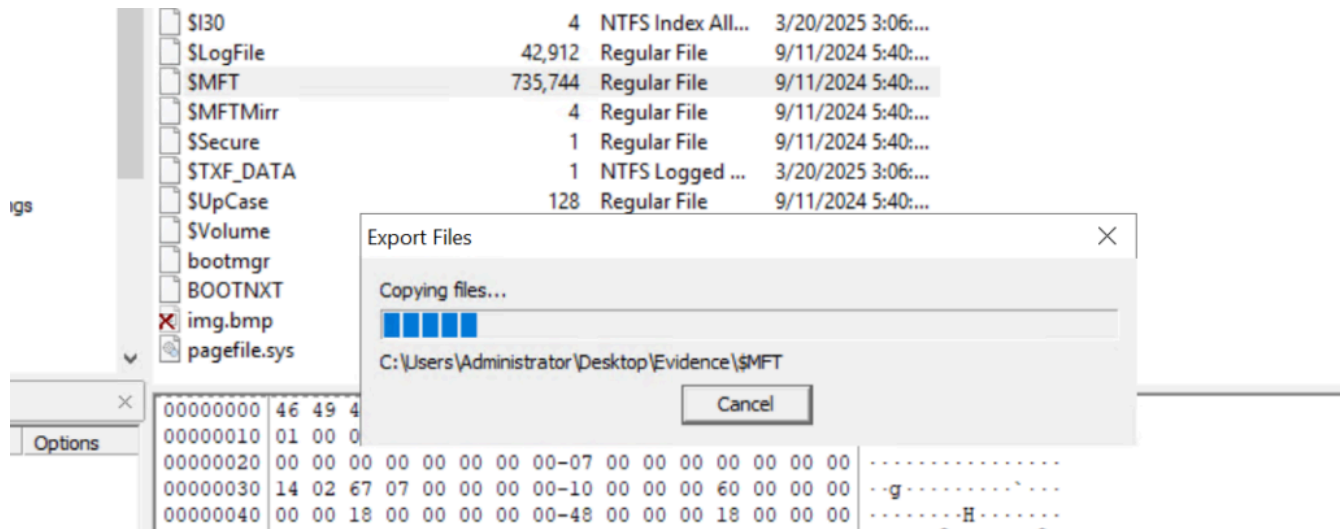
Once the MBR has been repaired and the image loaded, FTK Imager will display the partitions and expose their file systems. While Autopsy and other automated tools can handle a large portion of the analysis and save time, manual inspection gives you a deeper understanding of how Windows stores metadata and how to validate automated results. In this article we will show how to manually get the results and put the results together using Zimmer’s forensic utilities.

\$MFT

Our next goal is to analyse the \$MFT (Master File Table). The \$MFT is a special system file on NTFS volumes that acts as an index for every file and directory on the file system. It contains records with metadata about filenames, timestamps, attributes, and, in many cases, pointers to file data. The \$MFT is hidden in File Explorer, but it is always present on NTFS volumes (for example, C:\$MFT)

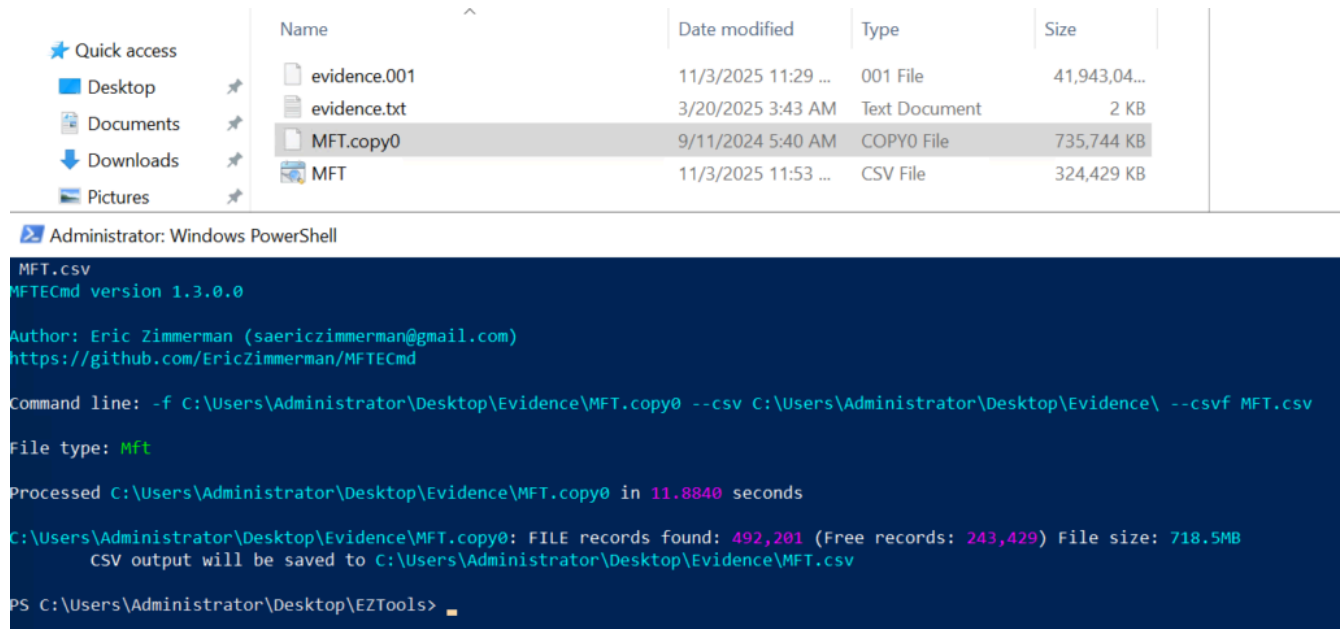


Export the \$MFT from the mounted or imaged volume. Right-click the \$MFT entry in your forensic viewer and choose **Export Files**



To parse and extract readable output from the \$MFT you can use MFTECmd.exe, a tool included in Eric Zimmerman’s EZTools collection. From a command shell run the extractor, for example:

```
PS> MFTECmd.exe -f ..\Evidence\MFT --csv ..\Evidence\ --csvf MFT.csv
```



The command above creates a CSV file you can use for keyword searches and timeline work. If needed, rename the exported files to make it easier to work with them in PowerShell.

	File Name	Extension	Is Directory	Has Ads	Is Ads	File Size	Created@x10
	password	.txt	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	=	=
gle\Chrom...	passwords.txt	.txt	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	241951	2025-03-19 22:01:57

When a CSV file is opened, you can use basic keyword search or pick an extension to see what files existed on the drive.

Understanding and working with \$MFT records is important. If a suspect deleted a file, the \$MFT may still contain its last known filename, path, timestamps and sometimes even data pointers. That information lets investigators target data recovery and build a timeline of the suspect's activity.

Suspicious Files

During inspection of the second partition we located several suspicious entries. Many were marked as deleted but can still be exported and examined.

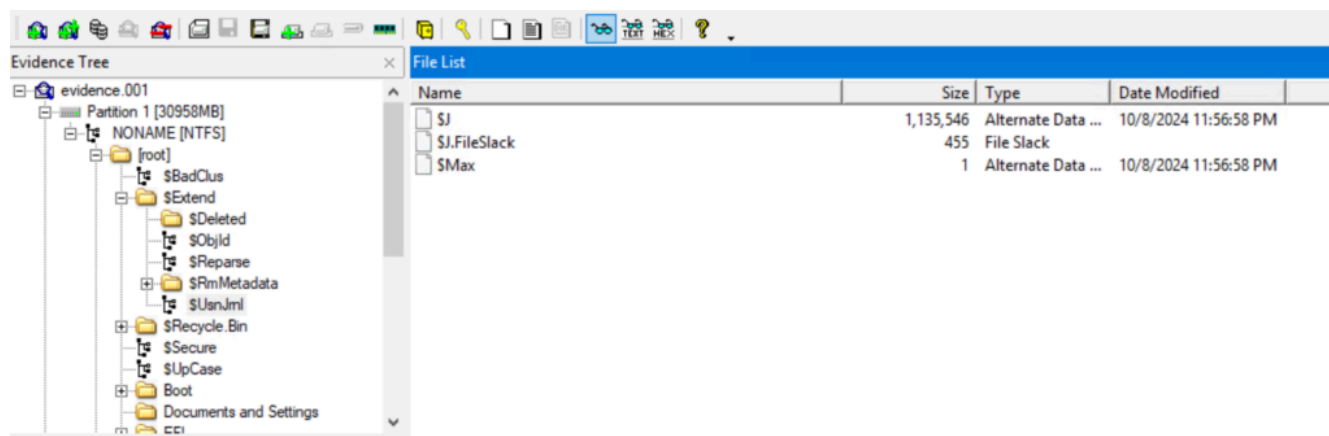
Name	Size	Type	Date Modified
!emo.zip	0	Regular File	3/20/2025 12:12:54 AM
!emo.zip	2	Regular File	3/20/2025 12:12:08 AM
DiskWipe.exe	0	Regular File	3/20/2025 12:14:08 AM
DiskWipe.exe	1,103	Regular File	3/20/2025 12:14:02 AM
employee_list.xlsx	2	Regular File	3/20/2025 1:06:50 AM
employee_list.xlsx	2	Regular File	3/20/2025 1:06:50 AM
employee_list.xlsx	2	Regular File	3/20/2025 1:06:50 AM
employee_list.xlsx	2	Regular File	3/20/2025 12:08:24 AM
importantDocs-case-20250320T000805Z-001.zip	0	Regular File	3/20/2025 12:08:12 AM
importantDocs-case-20250320T000805Z-001.zip	1	Regular File	3/20/2025 12:08:12 AM
importantDocs-case-20250320T000805Z-001.zip.FileSlack	8	File Slack	
importantDocs.zip	1	Regular File	3/20/2025 12:08:12 AM
importantDocs.zip.FileSlack	8	File Slack	
Minutes_of_the_Meeting.docx	9	Regular File	3/20/2025 1:02:40 AM
Minutes_of_the_Meeting.docx	9	Regular File	3/20/2025 1:02:40 AM
Minutes_of_the_Meeting.docx	9	Regular File	3/20/2025 1:02:40 AM
New Text Document.txt	1	Regular File	3/20/2025 1:10:12 AM
New Text Document.txt	1	Regular File	3/20/2025 1:10:12 AM
New Text Document.txt	1	Regular File	3/20/2025 1:10:12 AM
Quantum-Resistant Cryptographic Algorithms.pdf	59	Regular File	3/20/2025 12:44:40 AM
Quantum-Resistant Cryptographic Algorithms.pdf	59	Regular File	3/20/2025 12:44:40 AM

The evidence shows the perpetrator had a utility named DiskWipe.exe, which suggests an attempt to remove traces. We also found references to sensitive corporate documents, which together indicates data exfiltration. At this stage we can confirm the machine was used to access sensitive files. If we decide to analyze further, we can use registry and disk data to see whether the wiping utility was actually executed and what user executed it. This is outside of our scope today.

\$USNJRNL

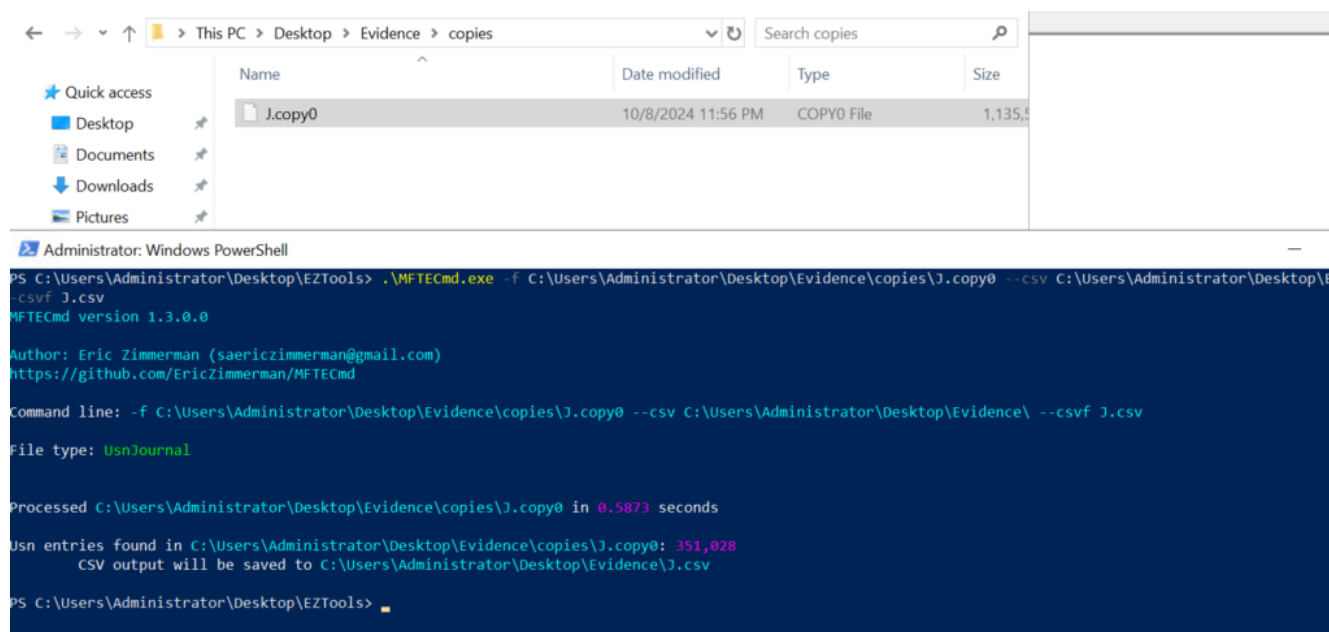
The \$USNJRNL (Update Sequence Number Journal) is another hidden NTFS system file that records changes to files and directories. It logs actions such as creation, modification and deletion before those actions are committed to disk. Because it records a history of file-system operations, \$UsnJrnl (\$J) can be invaluable in cases involving mass file deletion or tampering.

To extract the journal, first go to **root**, then **\$Extend** and double-click **\$UsnJrnl**. You need a **\$J** file.



You can then parse it with MFTECmd in the same way:

```
PS> MFTECmd.exe -f ..\Evidence$J --csv ..\Evidence\ --csvf J.csv
```



Since the second partition had the wiper, we can assume the perpetrator deleted files to cover traces. Let's open the CSV in Timeline Explorer and set the Update Reason to FileDelete to

view deleted files.

Number	Parent Sequence Number	Update Sequence Number	Update Reasons
	=	=	FileDelete
62805	1	1124143920	FileDelete Close
62805	1	1124144304	FileDelete Close
62805	1	1124144688	FileDelete Close
62805	1	1124145072	FileDelete Close
62805	1	1124145456	FileDelete Close
62805	1	1124145840	FileDelete Close
62805	1	1124146224	FileDelete Close
62805	1	1124146608	FileDelete Close

Name	Extension
data	.c
DownloadMetadata~RF7c2dc4a.TMP	.TMP
DownloadMetadata~RF7c38d99.TMP	.TMP
DownloadMetadata~RF7c38d99.TMP	.TMP
data Exfil.lnk	.lnk
AccessData FTK Imager.msi	.msi
metadata	
DownloadMetadata~RF83dc83f.TMP	.TMP
DownloadMetadata~RF83dc83f.TMP	.TMP
BootstrapperApplicationData.xml	.xml
BootstrapperApplicationData.xml	.xml
DownloadMetadata~RF86a2c91.TMP	.TMP
DownloadMetadata~RF86a2c91.TMP	.TMP
DownloadMetadata~RF86af7e0.TMP	.TMP
DownloadMetadata~RF86af7e0.TMP	.TMP
DownloadMetadata~RF86dafad.TMP	.TMP
DownloadMetadata~RF86dafad.TMP	.TMP
data Exfil	

Update Reasons Contains FileDelete And Name Contains data

Among the deleted entries we found a folder named “**data Exfil.**” In many insider exfiltration cases the perpetrator will compress those folders before transfer, so we searched \$MFT and \$J for archive extensions. Multiple entries for files named “**New Compressed (zipped) Folder.zip**” were present.

Name	Extens...	Entry Num...	Sequence...	Parent Entry...	Parent S...	Update Se...	Update Reasons
New Compressed (zipped) Folder.zip	.zip	163905	3	163896	8	1160686568	FileCreate
New Compressed (zipped) Folder.zip	.zip	163905	3	163896	8	1160686696	DataExtend FileCreate
New Compressed (zipped) Folder.zip	.zip	163905	3	163896	8	1160686824	DataExtend FileCreate Close
New Compressed (zipped) Folder.zip	.zip	163905	3	163896	8	1160686952	ObjectIdChange
New Compressed (zipped) Folder.zip	.zip	163905	3	163896	8	1160687080	ObjectIdChange Close
\$I70YEUS.zip	.zip	164477	2	7800	5	1160688048	FileCreate
\$I70YEUS.zip	.zip	164477	2	7800	5	1160688136	DataExtend FileCreate
\$I70YEUS.zip	.zip	164477	2	7800	5	1160688224	DataExtend FileCreate Close
New Compressed (zipped) Folder.zip	.zip	163905	3	163896	8	1160688312	RenameOldName
\$R70YEUS.zip	.zip	163905	3	7800	5	1160688440	RenameNewName
\$R70YEUS.zip	.zip	163905	3	7800	5	1160688528	RenameNewName Close
\$R70YEUS.zip	.zip	163905	3	7800	5	1160688616	SecurityChange
\$R70YEUS.zip	.zip	163905	3	7800	5	1160688704	SecurityChange Close
\$R70YEUS.zip	.zip	163905	3	7800	5	1162785624	FileDelete Close
\$I70YEUS.zip	.zip	164477	2	7800	5	1162785712	FileDelete Close

The journal shows the zip was created and files were appended to it. The final operation was a rename (RenameOldName). Using the Parent Entry Number exposed in \$J we can correlate entries and recover the original folder name.

Name	Extension	Entry Number	Sequence...	Parent Entry...	Parent S...	Update Se...	Update Reasons
Preferences~RF7c3fe25.TMP	.TMP	163896	7	42843	3	1160673768	FileDelete Close
New folder		163896	8	6330	5	1160674272	FileCreate
New folder		163896	8	6330	5	1160674352	FileCreate Close
New folder		163896	8	6330	5	1160679424	RenameOldName
data Exfil		163896	8	6330	5	1160679504	RenameNewName
data Exfil		163896	8	6330	5	1160679584	RenameNewName Close
data Exfil		163896	8	6330	5	1160679664	ObjectIdChange
data Exfil		163896	8	6330	5	1160679744	ObjectIdChange Close
data Exfil		163896	8	6330	5	1162775848	FileDelete Close

As you can see, using the Parent Entry Number we found that the original folder name was “data Exfil” which was later deleted by the suspect.

Timeline

From the assembled artifacts we can conclude that the machine was used to access and exfiltrate sensitive material. We found Excel sheets, PDFs, text documents and zip archives with sensitive data. The insider created a folder called “data Exfil,” packed its contents into an archive, and then attempted to cover tracks using a wiper. DiskWipe.exe and the deleted file entries support our hypothesis. To confirm execution and attribute actions to a user, we can examine registry entries, prefetch files, Windows event logs, shellbags and user profile activity that may show us process execution and the account responsible for it. The corrupted MBR suggests the perpetrator also intentionally damaged the boot sector to complicate inspection.

Summary

Digital forensics is a fascinating field. It exposes how much information an operating system preserves about user actions and how those artifacts can be used to reconstruct events. Many Windows features were designed to improve reliability and user experience, but those same features give us useful forensic traces. Although automated tools can speed up analysis, skilled analysts must validate tool output by understanding the underlying data structures and by performing manual checks when necessary. As you gain experience with the \$MFT, \$UsnJrnl and low-level disk structures, you will become more effective at recovering evidence and validating your hypotheses. See you soon!