Digital Forensics

Fundamentals

LinkedIn

Ondřej Šrámek

GMON GNFA GCTI

- Cyber Manager @ Honeywell
- AZ @ 92. VeInKyS
- Lektor Czechitas

czechitas

Agenda

In this session we will explore the fundamentals of digital forensics.

Introduction to Digital Forensics

Disk Acquisition and Evidence Handling

Filesystem Internals (FAT and NTFS)

Deleted Files and File Recovery

Common Forensic Artifacts

Forensic Tools and Investigation Workflow

Timeline and Supertimeline Analysis

Key Takeaways

Digital Forensics

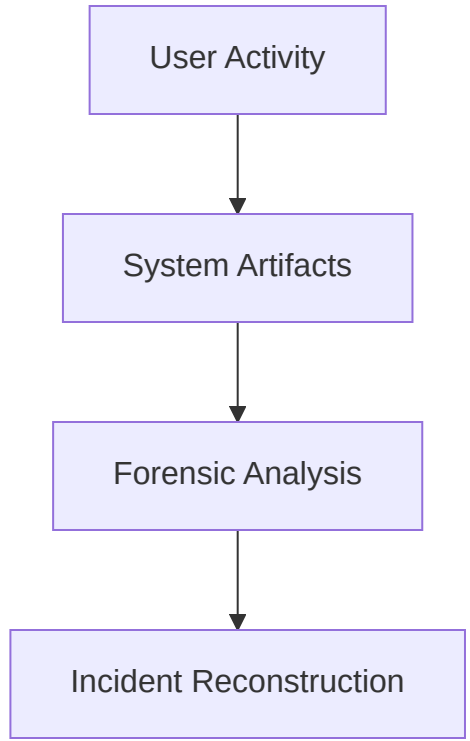
Why digital forensics matters

What is Digital Forensics?

Digital forensics is the process of:

1. Collecting evidence
2. Preserving evidence
3. Analyzing artifacts
4. Presenting findings

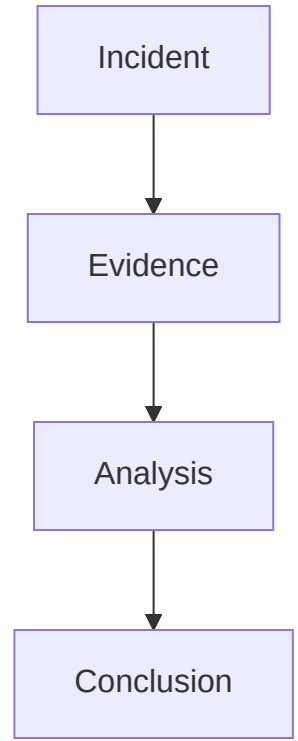
The goal is to reconstruct **what actually happened on a system**.



Questions Forensics Can Answer

Investigators try to answer questions such as:

- What happened?
- When did it happen?
- How did it happen?
- Who was involved?
- What data was affected?

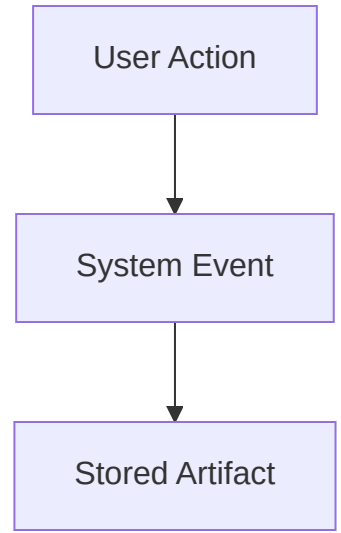


Digital Evidence is Everywhere

Modern systems constantly generate traces.
Examples include:

- filesystem metadata
- system logs
- browser history
- application data
- network logs

These traces become **forensic artifacts**.



Example Incident

Imagine the following situation:

- A company reports suspicious activity
- Files suddenly disappear
- Unusual network traffic appears

Investigators must determine:

- what happened
- how the attacker entered
- what data was accessed

Digital forensics helps answer these questions.

Real-World Use Cases

- malware investigations
- data breaches
- insider threats
- fraud investigations
- corporate investigations
- law enforcement
- incident response
- threat hunting

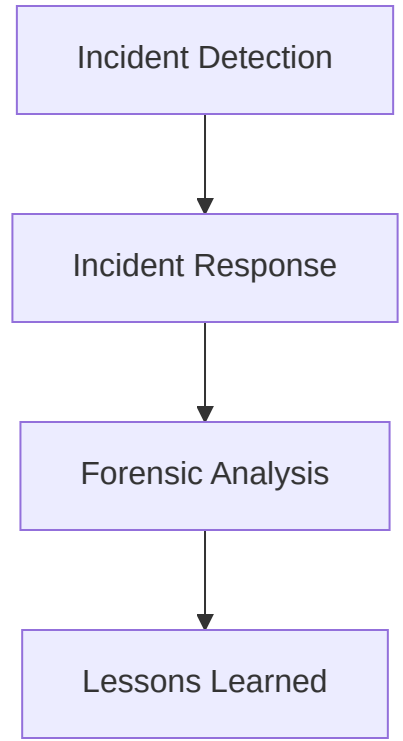
Digital Forensics vs Incident Response

Incident Response

- detect attacks
- contain threats
- restore systems

Digital Forensics

- analyze evidence
- reconstruct events
- support investigations



Skills Used in Digital Forensics

Digital forensics combines several disciplines:

- technical knowledge
- analytical thinking
- attention to detail
- investigative methodology

It is both **technical analysis and investigative work**.

What You Will Learn

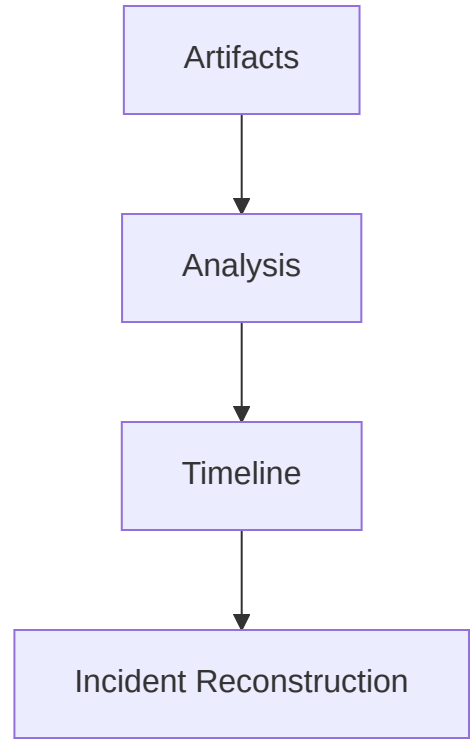
In this workshop we will explore:

1. Disk acquisition
2. Filesystem structures
3. Forensic artifacts
4. File recovery
5. Timeline reconstruction

These concepts form the foundation of forensic analysis.

From Evidence to Story

Digital forensics turns raw technical data into a narrative explaining what happened.



Disk Acquisition

Collecting forensic disk images

What is Disk Acquisition

Disk acquisition is the process of creating a **forensic copy of storage media**.

The goal is to preserve digital evidence while allowing analysis on a duplicate.

Key principle:

> *Never analyze the original evidence.*

Investigators work on a copy of the data.

Why Imaging is Necessary

Direct analysis of the original disk can:

- modify timestamps
- alter filesystem metadata
- destroy evidence

Forensic imaging prevents these risks.

Types of Acquisition

Dead acquisition

- system powered off
- disk removed
- image created from storage media

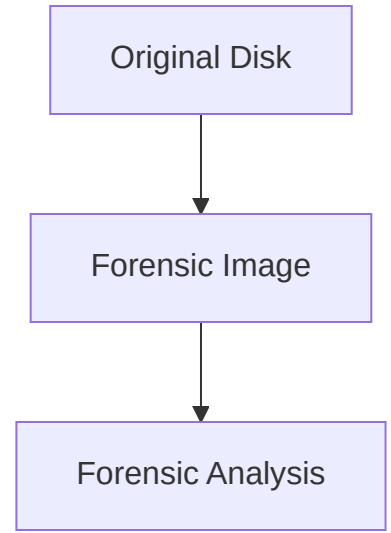
Live acquisition

- system running
- image collected from active system

Dead acquisition is preferred whenever possible.

Disk Imaging Concept

The forensic image contains an **exact sector-by-sector copy** of the disk.



Disk Image Formats

Common forensic image formats:

- RAW (`.img`)
- E01 (EnCase format)
- AFF (Advanced Forensic Format)

RAW images are simple sector-by-sector copies.

Write Blockers

Write blockers prevent modification of the original disk.

Types:

- hardware write blockers
- software write blockers

Example hardware devices:

- Tableau write blocker
- WiebeTech write blocker

These ensure evidence integrity.

Imaging with dd

The `dd` utility performs a raw disk copying.

Parameters:

- `if` → input file (source disk)
- `of` → output file (image file)
- `bs` → block size

```
dd if=/dev/sda of=disk.img bs=4M status=progress
```

dd with Error Handling

Example forensic imaging command with error handling:

```
dd if=/dev/sda of=evidence.img bs=4M conv=noerror,sync
```

Options:

- `noerror` → continue on read errors
- `sync` → pad missing blocks

Useful for damaged disks.

dc3dd

`dc3dd` is an enhanced forensic version of `dd`.

Features:

- hash calculation
- error handling
- progress reporting

```
dc3dd if=/dev/sda of=disk.img hash=sha256 log=acquisition.log
```

GUI Imaging Tools

Graphical acquisition tools include:

- Guymager
- FTK Imager
- EnCase Imager

These tools provide:

- hash verification
- evidence metadata
- acquisition logging

Hash Verification

After imaging, investigators verify integrity using hashes.

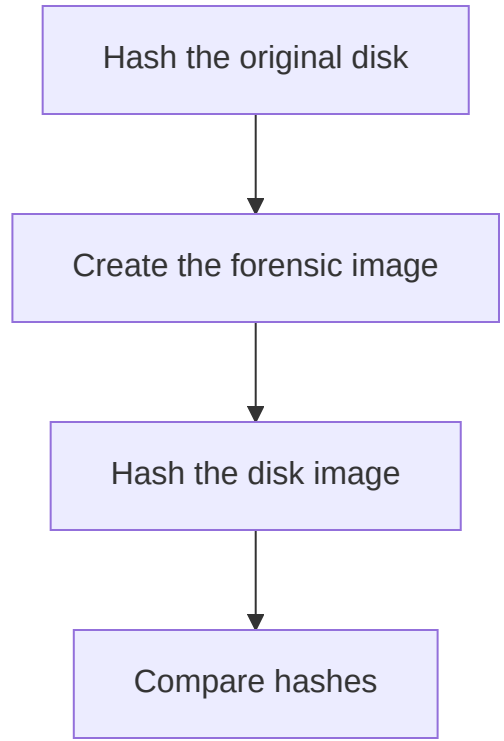
Hash values confirm the image was not modified.

```
sha256sum disk.img
```

```
b1946ac92492d2347c6235b4d2611184  disk.img
```

Hash Verification Workflow

Matching hashes confirm image integrity.



Chain of Custody

Investigators must document evidence handling.

Example information:

- who collected the evidence
- when it was collected
- where it was stored
- who accessed it

Proper documentation ensures legal validity.

Key Takeaways

Important acquisition principles:

- never modify original evidence
- create verified disk images
- use write blockers
- document the acquisition process

Filesystems

Understanding how data is stored

Why Filesystems Matter

Digital forensics often begins with understanding the filesystem.

A filesystem defines:

- how data is stored
- how files are organized
- how metadata is recorded

Forensic analysis relies heavily on filesystem structures.

What a Filesystem Does

A filesystem is responsible for:

- organizing files and directories
- allocating disk space
- storing metadata
- tracking file locations

Without the filesystem, data would simply be raw bytes on disk.

Filesystem Components

Most filesystems contain similar core structures:

- boot sector
- allocation structures
- directory entries
- data area

Understanding these structures is essential for forensic analysis.

Disk Layout Overview

Each area serves a specific purpose:

- **Boot Sector** – filesystem parameters and boot code
- **Allocation Tables** – track which clusters are in use
- **Directory Entries** – file names and metadata
- **Data Area** – actual file content

File Metadata

Filesystems store metadata about files.

Common metadata includes:

- file name
- file size
- timestamps
- file permissions
- file location

This metadata is often crucial forensic evidence.

File Timestamps

Most filesystems track timestamps such as:

- creation time
- modification time
- access time
- metadata change time

These timestamps are often used to build forensic timelines.

The MACB Model

In forensic analysis, timestamps are often summarized as MACB.

These timestamps help investigators reconstruct activity on a system.

```
M → Modified
A → Accessed
C → Metadata Changed
B → Created (Birth)
```

File Deletion

When a file is deleted:

1. The filesystem marks the entry as deleted
2. Disk space becomes available for reuse
3. The file data may still remain on disk

Because of this behavior, deleted files can often be recovered.

Why This Matters for Forensics

Filesystem analysis allows investigators to:

- recover deleted files
- analyze metadata
- reconstruct timelines
- identify suspicious activity

Understanding filesystem internals is a core skill in digital forensics.

Filesystems We Will Examine

In this workshop we will focus on:

- FAT filesystems
- NTFS filesystems

These filesystems are common and widely used in forensic investigations.

FAT Filesystem

File Allocation Table filesystem

What is FAT?

FAT stands for **File Allocation Table**.

It is one of the earliest widely used filesystems and is still found on:

- USB drives
- SD cards
- embedded systems
- legacy Windows systems

Because of its simple structure, FAT is often used to teach filesystem forensics.

FAT Filesystem Structure

A FAT disk typically contains four main areas:

- **Boot Sector** – filesystem parameters
- **FAT Table** – cluster allocation tracking
- **Root Directory** – top-level directory entries
- **Data Area** – actual file content

Each area has a specific role in managing files.

Boot Sector

The boot sector contains important filesystem metadata.

Examples include:

- filesystem type
- cluster size
- number of FAT tables
- location of filesystem structures

Investigators often analyze the boot sector to understand disk layout.

FAT Table Concept

The **File Allocation Table** tracks how clusters are linked together.

Each cluster points to the next cluster belonging to the file.

```
Cluster   FAT Entry
-------   ---------
2         3
3         4
4         EOF
```

Cluster chain:

```
2 → 3 → 4 → EOF
```

File Stored in Clusters

Files are stored across one or more clusters.

The FAT table links these clusters together into a chain.

A file may span many clusters depending on its size.

Directory Entries

Files are represented by directory entries.

Typical directory entry fields include:

- filename
- extension
- attributes
- timestamps
- starting cluster
- file size

These entries are critical forensic artifacts.

FAT File Deletion

When a file is deleted in FAT:

1. The first character of the filename is replaced
2. The directory entry becomes available
3. The data clusters remain on disk

Because of this behavior, deleted files can often be recovered.

Deleted File Marker

The first byte of the filename is replaced with `0xE5`.

Original filename:

```
TEST.TXT
```

The entry is marked as deleted but still contains recoverable data.

After deletion:

```
?EST.TXT
```

First byte → `0xE5`

Why Recovery Works

Deletion only modifies filesystem metadata.

The data clusters often remain untouched until overwritten.

Forensic tools can reconstruct files from this information.

```
Directory entry → marked deleted
Cluster chain  → still contains data
```

czechitas

FAT Forensic Opportunities

Investigators can often:

- recover deleted files
- analyze directory entries
- reconstruct cluster chains
- identify previously deleted filenames

This makes FAT relatively easy to analyze compared to more complex filesystems.

Key Takeaway

FAT deletion does **not erase data**.

It only:

- marks directory entries as deleted
- frees clusters for reuse

Until clusters are overwritten, the data may still be recoverable.

NTFS Filesystem

NTFS filesystem internals

What is NTFS?

NTFS stands for **New Technology File System**.

It is the primary filesystem used by modern Windows systems.

NTFS provides many advanced features:

- journaling
- access control lists
- large file support
- extensive metadata storage

These features also make NTFS extremely valuable for forensic investigations.

NTFS Key Concept

The most important NTFS structure is the:

Master File Table (MFT)

Every file and directory on an NTFS volume is represented by a record in the MFT.

Master File Table (MFT)

The MFT is essentially a database of all files.

Each file has its own **MFT record**.

Even system files and directories are stored as MFT records.

MFT Record Size

Typical properties:

- default size: **1024 bytes**
- contains file metadata
- contains file attributes
- may contain file data (for small files)

Small files may be stored **directly inside the MFT**.
This is called **resident data**.

NTFS Attributes

NTFS stores information using attributes.

Common attributes include:

- `$STANDARD_INFORMATION`
- `$FILE_NAME`
- `$DATA`
- `$SECURITY_DESCRIPTOR`

Each attribute stores different information about the file.

$STANDARD_INFORMATION

This attribute contains important metadata:

- timestamps
- file permissions
- file owner
- security identifiers

Timestamps stored here are often used in forensic timelines.

$FILE_NAME Attribute

This attribute contains:

- filename
- parent directory reference
- additional timestamps

Interestingly, timestamps here may differ from `$STANDARD_INFORMATION`.

This difference can reveal forensic artifacts.

NTFS Timestamps

NTFS tracks multiple timestamps, commonly referred to as **MACB**.

These timestamps are extremely important for forensic timelines.

```
M → Modified
A → Accessed
C → Metadata Changed
B → Created (Birth)
```

Timestamp Storage Locations

NTFS stores timestamps in multiple locations.

Because timestamps exist in multiple places, investigators can detect inconsistencies.

```
$STANDARD_INFORMATION
$FILE_NAME
```

Detecting Timestamp Manipulation

If an attacker performs **timestomping**, timestamps may differ between attributes.

This mismatch may indicate timestamp manipulation.

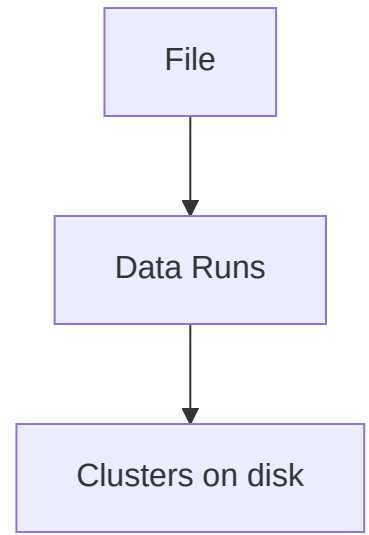
```
$STANDARD_INFORMATION → 2015
$FILE_NAME            → 2024
```

Data Runs

For larger files, NTFS stores file data outside the MFT.

The `$DATA` attribute contains **data runs** that point to disk clusters.

These runs map logical file data to physical disk locations.



Alternate Data Streams (ADS)

NTFS supports **Alternate Data Streams**.

This allows multiple data streams to exist within a single file. ADS can be abused to hide data and may not appear in normal directory listings.

```
file.txt
file.txt:hidden
```

Example ADS Usage

The hidden stream will not appear in normal directory listings. Investigators must specifically check for ADS.

```
echo secret > file.txt:hidden
```

Why NTFS is Valuable for Forensics

NTFS stores large amounts of metadata.

Investigators can analyze:

- MFT records
- timestamps
- file attributes
- alternate data streams

These artifacts help reconstruct system activity.

Key Takeaway

NTFS is rich in forensic evidence.

Because of its metadata structures, investigators can often:

- reconstruct file activity
- detect timestamp manipulation
- discover hidden data
- rebuild timelines from filesystem artifacts

File Recovery

Recovering deleted and lost files

Why File Recovery Matters

When files are deleted, the data is usually **not immediately erased**.

Instead:

- filesystem metadata changes
- disk space becomes available for reuse
- file data often remains on disk

This behavior allows investigators to recover deleted files.

How File Deletion Works

In most filesystems, deletion follows a similar process:

1. The directory entry is marked as deleted
2. Disk clusters are marked as free
3. The file data remains until overwritten

Because of this, deleted files may remain recoverable.

Recovery Methods

Investigators typically use two recovery approaches:

- filesystem-based recovery
- file carving

Both techniques are commonly used in digital forensics.

Filesystem-Based Recovery

Filesystem-based recovery relies on filesystem metadata.

Examples:

- FAT directory entries
- NTFS MFT records
- cluster chains

If metadata still exists, tools can reconstruct the file structure.

File Carving

File carving is used when filesystem metadata is missing or damaged.

Instead of metadata, carving relies on **file signatures**.

Investigators scan raw disk data to locate recognizable file headers.

File Signature Example

File signatures identify file types in raw disk data.

These bytes appear at the beginning of recognized file types and allow carving tools to locate files even without filesystem metadata.

JPEG signature:

```
FFD8FFE0
```

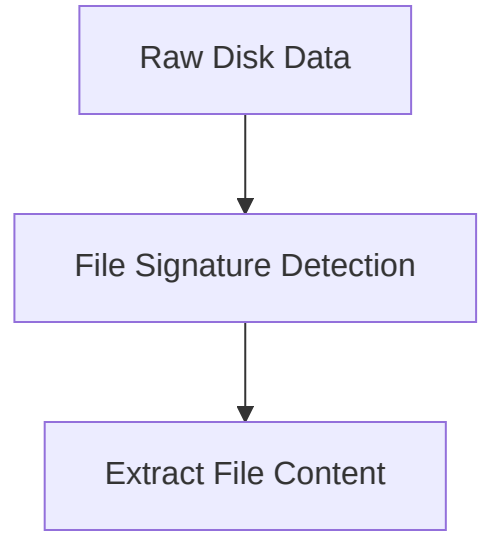
PDF signature:

```
25504446
```

File Carving Concept

The tool scans the disk for known file headers and reconstructs files.

This approach works even when filesystem metadata is missing or damaged.



File Carving Tools

Common file carving tools include:

- scalpel
- foremost
- bulk extractor
- photorec

These tools analyze raw disk data and extract recognizable files.

Example Scalpel Configuration

Scalpel uses configuration files to define file signatures.

This tells Scalpel how to detect JPEG files.

```
jpg     y       2000000     \xff\xd8\xff\xe0
```

czechitas

Running Scalpel

The tool scans the disk image and extracts files into the output directory.

```
scalpel disk.img -o output_directory
```

Limitations of File Carving

File carving is powerful but has limitations:

- fragmented files may not be recoverable
- recovered files may lack filenames
- timestamps are often missing

Because of this, carving is often used as a **last resort**.

Forensic Workflow Example

A typical recovery workflow may look like this:

1. Acquire disk image
2. Identify filesystem
3. Attempt filesystem-based recovery
4. Perform file carving if needed
5. Analyze recovered files

This approach maximizes the chance of recovering useful evidence.

Key Takeaway

Deleted files often remain recoverable.

Investigators can recover files using:

- filesystem metadata
- raw disk analysis
- file signature detection

File recovery is a fundamental capability in digital forensics.

Forensic Tools

Tools used in forensic investigations

Why Tools Matter

Digital forensics relies heavily on specialized tools.

These tools help investigators:

- acquire disk images
- analyze filesystems
- recover deleted files
- extract forensic artifacts
- build timelines

Understanding how these tools work is essential for investigations.

Tool Categories

Forensic tools generally fall into several categories:

- acquisition tools
- filesystem analysis tools
- recovery tools
- artifact extraction tools
- timeline analysis tools

Each category supports a different phase of the investigation.

Disk Imaging Tools

Common disk acquisition tools include:

- `dd`
- `dc3dd`
- FTK Imager
- Guymager

These tools create **forensic images of storage media**.

Example: dd

The `dd` command performs a raw sector-by-sector copy.

This command copies an entire disk into an image file.

```
dd if=/dev/sda of=disk.img bs=4M status=progress
```

Example: dc3dd

`dc3dd` is an enhanced forensic version of `dd`.

Additional features include:

- hash verification
- improved error handling
- logging

```
dc3dd if=/dev/sda of=disk.img hash=sha256 log=acquisition.log
```

The Sleuth Kit

The Sleuth Kit (TSK) is a widely used forensic toolkit.

It allows investigators to:

- analyze filesystems
- recover deleted files
- inspect filesystem metadata

TSK works directly with disk images.

Important Sleuth Kit Tools

Common Sleuth Kit commands include:

- `mmls`
- `fsstat`
- `fls`
- `icat`
- `istat`

Each command provides different forensic insights.

Example: mmls

`mmls` displays partition layouts.

This helps identify partitions inside a disk image.

```
mmls disk.img
```

Example: fls

`fls` lists files and directories from a filesystem.

Options:

- `-r` → recursive listing
- `-d` → show deleted files

```
fls -r disk.img
```

Example: icat

`icat` extracts file content from a disk image.

This command extracts the file associated with inode 128.

```
icat disk.img 128 > recovered_file.txt
```

File Carving Tools

File carving tools recover files without filesystem metadata.

Common tools include:

- scalpel
- foremost
- photorec
- bulk extractor

These tools scan raw disk data for file signatures.

Example: Scalpel

Scalpel scans the disk image and extracts recognized file types.

```
scalpel disk.img -o output_directory
```

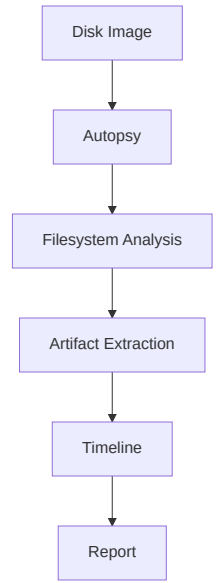
Autopsy

Autopsy is an open-source GUI forensic workbench built on top of The Sleuth Kit.

Autopsy provides a complete investigation environment:

- disk image analysis
- deleted file recovery
- keyword search
- timeline analysis
- artifact extraction (browser history, registry, email)
- built-in reporting

It is one of the most widely used tools in digital forensics and is free to use.



Autopsy vs Command-Line Tools

Autopsy (GUI)

- beginner friendly
- visual file browser
- automated modules
- integrated reporting
- all-in-one workflow

Sleuth Kit (CLI)

- full control
- scriptable
- faster for specific tasks
- better for automation
- requires more experience

Timeline Tools

Timeline tools reconstruct system activity chronologically.

Common tools include:

- log2timeline
- plaso
- Timesketch

These tools combine multiple artifact sources.

Example: log2timeline

This extracts artifacts from the disk image into a timeline database.

```
log2timeline.py timeline.plaso disk.img
```

Creating a Timeline

Example workflow:

1. Extract artifacts using log2timeline
2. Convert timeline data
3. Analyze activity chronologically

```
psort.py -o l2tcsv timeline.plaso > timeline.csv
```

This produces a timeline that investigators can analyze.

Typical DFIR Tool Workflow

A simplified investigation workflow might look like:

1. Acquire disk image (`dd`)
2. Identify partitions (`mmls`)
3. Analyze filesystem (`fsstat`)
4. List files (`fls`)
5. Recover files (`icat`)
6. Extract artifacts (`log2timeline`)
7. Build timeline

Each step reveals additional evidence.

Key Takeaway

Forensic tools help investigators transform raw disk data into useful evidence.

By combining multiple tools, investigators can:

- analyze filesystems
- recover deleted files
- extract artifacts
- reconstruct system activity

Forensic Artifacts

Where investigators find evidence

What Are Forensic Artifacts?

Forensic artifacts are traces of activity left by users, applications, or the operating system.

These artifacts help investigators answer questions such as:

- What happened?
- When did it happen?
- Which user was involved?

Artifacts provide valuable evidence about system activity.

Sources of Artifacts

Artifacts can originate from many system components:

- filesystems
- operating system logs
- application data
- browser history
- system configuration

Investigators correlate multiple artifact sources to reconstruct events.

Artifact Categories

Artifacts typically fall into several categories:

- user activity
- program execution
- file access
- system events
- network activity

Each category reveals different aspects of system behavior.

Common Windows Artifacts

Important forensic artifacts on Windows systems include:

- Prefetch files
- Event Logs
- Registry
- Jump Lists
- Recycle Bin
- Browser artifacts

These artifacts help reconstruct user and system activity.

Windows Prefetch

Prefetch files record program execution.

Location:

```
C:\Windows\Prefetch\
```

Prefetch files contain:

- program name
- execution timestamps
- number of executions
- referenced files

Windows Event Logs

Windows logs many types of system events.

Location:

```
C:\Windows\System32\winevt\Logs\
```

Event logs contain information about:

- authentication activity
- service execution
- process creation
- system events

Important Windows Event IDs

Common event IDs used in investigations include:

- **4624** – successful login
- **4625** – failed login
- **4688** – process creation
- **4720** – user account created
- **7045** – service installed

These events help investigators track system activity.

Windows Registry

The Windows Registry stores configuration and activity information.

Registry hive location:

```
C:\Windows\System32\Config\
```

Important hives include:

- SAM
- SYSTEM
- SOFTWARE
- SECURITY
- NTUSER.DAT

Registry analysis often reveals user activity and system configuration.

Linux System Logs

Important log locations:

```
/var/log/syslog
/var/log/auth.log
/var/log/messages
```

These logs record:

- system activity
- authentication attempts
- service events

Linux User Artifacts

User activity may appear in:

```
~/.bash_history
~/.ssh/
~/.config/
```

These files may reveal:

- executed commands
- SSH connections
- application configuration

macOS System Artifacts

macOS systems store artifacts in several locations:

- unified system logs
- recent items databases
- application logs
- user activity databases

These artifacts help reconstruct system usage.

macOS Unified Logs

macOS uses a centralized logging system.

Investigators can query logs using:

```
log show
```

These logs may contain:

- system events
- application activity
- security events

Artifact Correlation

A single artifact rarely tells the whole story.

Investigators correlate multiple sources:

- filesystem metadata
- system logs
- registry entries
- application artifacts

This correlation helps reconstruct a timeline of events.

Key Takeaway

Forensic artifacts exist across many system components.

Effective investigations require:

- identifying relevant artifacts
- collecting evidence from multiple sources
- correlating artifact data

Timeline Analysis

Reconstructing system activity

Why Timeline Analysis Matters

Digital investigations often generate large amounts of data.

Timeline analysis helps investigators:

- organize events chronologically
- understand system activity
- identify suspicious behavior

A timeline turns raw artifacts into a story of what happened.

What Is a Timeline?

A forensic timeline is a chronological list of events extracted from system artifacts.

Example sources include:

- filesystem timestamps
- event logs
- registry artifacts
- application logs

These sources help reconstruct system activity.

Example Timeline

This sequence helps investigators understand the order of events.

```
10:01  User downloads file
10:02  File written to disk
10:03  File executed
10:05  Network connection established
```

Filesystem Timeline Data

Filesystems store timestamps that are useful for building timelines.

Typical timestamps include:

- Created
- Modified
- Accessed
- Metadata changed

These timestamps are commonly summarized as **MACB**.

MACB Timeline Model

These timestamps provide clues about file activity and help build forensic timelines.

```
M  Modified
A  Accessed
C  Metadata Changed
B  Created (Birth)
```

czechitas

Timeline Sources

Timeline data can be extracted from multiple sources:

- filesystem metadata
- system logs
- registry entries
- browser artifacts
- application data

Combining multiple sources produces a more complete timeline.

What Is a Supertimeline?

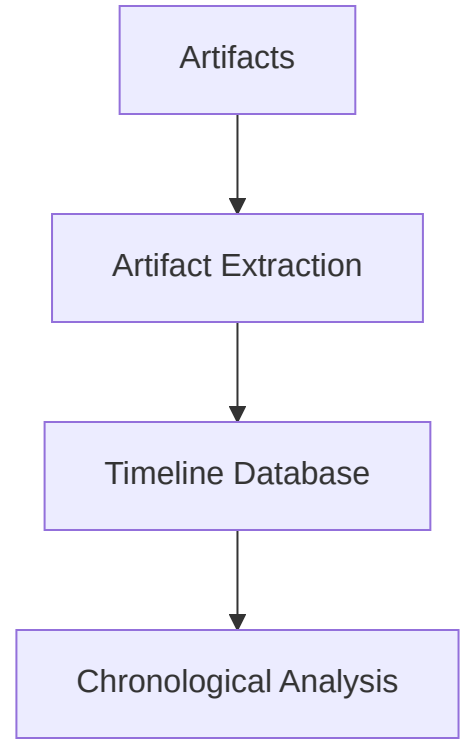
A **supertimeline** combines artifacts from many sources into a single timeline.

All events are merged into a unified chronological view.

```
Filesystem events
System logs
Registry artifacts
Application data
```

Supertimeline Concept

This approach helps investigators correlate system activity.



Creating a Supertimeline

Typical workflow:

1. Acquire disk image
2. Extract artifacts
3. Build timeline database
4. Analyze events chronologically

Tools like **log2timeline** automate this process.

Example Workflow

This produces a timeline investigators can analyze.

```
log2timeline.py timeline.plaso disk.img
```

```
psort.py -o l2tcsv timeline.plaso > timeline.csv
```

czechitas

Anti-Forensics: Timestomping

Attackers may attempt to hide activity by manipulating timestamps.

This technique is called **timestomping**.

The goal is to make malicious activity appear older or unrelated to the incident.

Example Timestomping Scenario

Original timeline:

```
10:01 malware.exe created
10:02 malware.exe executed
```

After timestomping:

```
08:13 malware.exe created
10:02 malware.exe executed
```

The file appears older than it really is.

Timestomping Tools

Attackers can modify timestamps using tools such as:

- PowerShell
- touch
- SetFile (macOS)

This command modifies file timestamps.

```
touch -t 201501010101 malware.exe
```

Detecting Timestomping

Investigators look for inconsistencies between artifacts.

Examples include:

- MFT timestamps vs `$FILE_NAME`
- filesystem timestamps vs event logs
- file creation time vs execution evidence

When timestamps do not match, manipulation may have occurred.

Correlating Artifacts

Good forensic analysis relies on **correlation**.

Investigators compare:

- filesystem metadata
- event logs
- registry artifacts
- application artifacts

This helps confirm or challenge timeline events.

Key Takeaway

Timeline analysis helps investigators reconstruct system activity.

By combining multiple artifact sources, investigators can:

- understand the sequence of events
- detect suspicious activity
- identify attempts to hide evidence

Memory Forensics

Analysing volatile memory

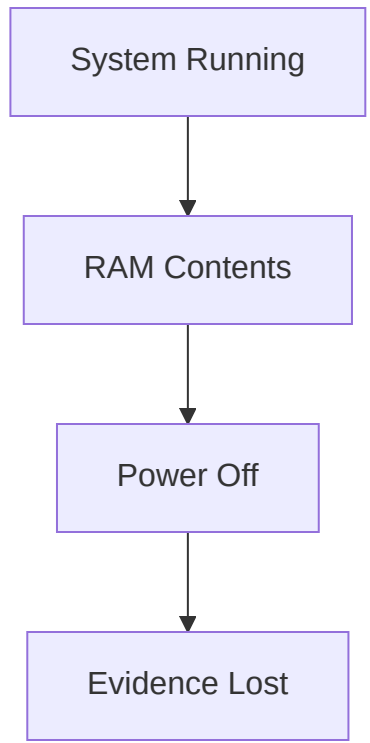
Why Memory Forensics Matters

RAM contains evidence that is never written to disk.

When a system is powered off, this evidence is permanently lost.

Memory can contain:

- running processes and their arguments
- active network connections
- decrypted file contents
- encryption keys and passwords
- malware that lives only in memory



Volatile vs Non-Volatile Evidence

Volatile (RAM)

- lost on shutdown
- running processes
- network connections
- decrypted data
- must be acquired first

Non-Volatile (Disk)

- persists after shutdown
- files and filesystem
- logs and registry
- can be acquired later

> *In live investigations, always acquire memory **before** touching the disk.*

czechitas

Memory Acquisition

Memory must be captured from a **live running system**.

Common acquisition tools:

- **WinPmem** – Windows memory acquisition
- **DumpIt** – Windows, simple single-file tool
- **LiME** – Linux Memory Extractor (kernel module)
- **OSXPmem** – macOS memory acquisition

```
winpmem.exe memory.raw
```

```
sudo insmod lime.ko
"path=/tmp/memory.raw format=raw"
```

The result is a raw memory dump file (`.raw` , `.mem` , `.dmp`).
Virtual machines are a special case — RAM is already stored as a file on disk:

- VMware → `.vmem`
- VirtualBox → `.sav`
- Hyper-V → `.bin`

Volatility 3

Volatility is the leading open-source memory forensics framework.

Version 3 improvements over v2:

- no more manual `--profile` selection
- automatic OS and version detection
- faster analysis engine
- plugins organised by OS: `windows.*`, `linux.*`, `mac.*`

Volatility 2 (old):

```
vol.py -f mem.raw --profile=Win10x64 pslist
```

Volatility 3:

```
vol -f mem.raw windows.pslist
```

Volatility 3 Basic Syntax

All plugins follow the same pattern:

```
vol -f memory.raw <plugin>
```

- `windows.*` – Windows memory analysis
- `linux.*` – Linux memory analysis
- `mac.*` – macOS memory analysis

Examples:

```
vol -f memory.raw windows.pslist
vol -f memory.raw windows.netscan
vol -f memory.raw windows.malfind
```

No profile needed — Volatility detects the OS automatically.

Process Analysis

Key process plugins:

- `windows.pslist` – list all running processes
- `windows.pstree` – show parent-child relationships
- `windows.cmdline` – command line arguments for each process

```
vol -f memory.raw windows.pslist
```

```
vol -f memory.raw windows.pstree
```

```
vol -f memory.raw windows.cmdline
```

Investigators look for:

- unexpected parent-child relationships
- processes with suspicious names
- processes running from unusual locations

Network Connections in Memory

`windows.netscan` reveals active and recently closed network connections.

Investigators look for:

- unexpected outbound connections
- connections to suspicious IP addresses
- unusual ports or protocols
- connections from unexpected processes

```
vol -f memory.raw windows.netscan
```

Example output fields:

```
Offset   Proto  LocalAddr  ForeignAddr  State   PID  Process
```

czechitas

Finding Malware with malfind

`windows.malfind` detects memory regions that may contain injected code.

It looks for:

- executable memory regions not backed by a file on disk
- regions with suspicious permissions (read, write, execute)
- common shellcode patterns

Results should be reviewed and may require further analysis.

```
vol -f memory.raw windows.malfind
```

```
vol -f memory.raw windows.dlllist
```

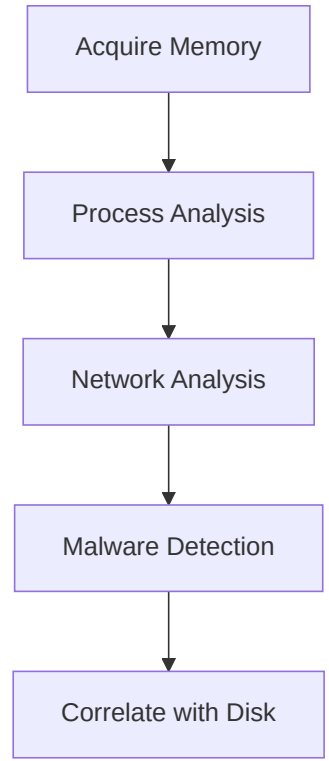
Useful Volatility 3 Plugins

Plugin	Purpose
`windows.pslist`	List running processes
`windows.pstree`	Process hierarchy
`windows.cmdline`	Process command lines
`windows.netscan`	Network connections
`windows.malfind`	Suspicious memory regions
`windows.dlllist`	Loaded DLLs per process
`windows.filescan`	File objects in memory
`windows.hashdump`	Extract password hashes

Memory Forensics Workflow

Typical memory analysis steps:

1. Acquire memory image from live system
2. Verify image integrity (hash)
3. List and review running processes
4. Inspect process command lines
5. Identify active network connections
6. Look for injected code or anomalies
7. Correlate findings with disk artifacts



Key Takeaway

Memory forensics reveals evidence that is invisible on disk.

Investigators can find:

- malware that never touches the disk
- active attacker tools and connections
- decrypted credentials and keys
- the full picture of what was running at the time of acquisition

Summary

Key concepts and investigation workflow

What We Covered

During this introduction to digital forensics we explored:

- filesystem structures
- forensic artifacts
- file recovery
- timeline analysis
- memory forensics
- forensic tools

These concepts form the foundation of forensic investigations.

Filesystem Concepts

We examined how filesystems store and manage data.

Key concepts include:

- filesystem layout
- metadata structures
- cluster allocation
- directory entries

Understanding filesystem internals is essential for forensic analysis.

FAT Filesystem

Important FAT concepts:

- cluster chains
- FAT allocation table
- directory entries
- deleted file markers

Deletion in FAT modifies metadata but often leaves data recoverable.

NTFS Filesystem

Important NTFS structures:

- Master File Table (MFT)
- file attributes
- timestamps
- alternate data streams

NTFS stores extensive metadata useful for forensic investigations.

File Recovery

When files are deleted:

1. The filesystem marks metadata as deleted
2. Disk space becomes available for reuse
3. File data may still remain on disk

Investigators can often recover deleted files from remaining disk data.

Forensic Artifacts

Artifacts provide evidence of system and user activity.

Common artifact sources include:

- filesystem metadata
- event logs
- registry data
- application artifacts

These artifacts help investigators reconstruct events.

Forensic Tools

Important forensic tools include:

- `dd`
- The Sleuth Kit
- Autopsy
- `scalpel`
- `log2timeline`
- Volatility 3

Each tool helps extract different types of forensic evidence.

Timeline Analysis

Timeline analysis reconstructs events chronologically.

Typical timeline sources include:

- filesystem timestamps
- system logs
- registry artifacts
- application data

This helps investigators understand system activity.

Memory Forensics

RAM contains volatile evidence that is lost on shutdown.

Key memory forensics concepts:

- acquire memory before shutting down the system
- Volatility 3 for memory analysis
- process analysis (`pslist`, `pstree`, `cmdline`)
- network connections (`netscan`)
- malware detection (`malfind`)

Memory forensics reveals evidence invisible on disk.

Anti-Forensics

Attackers may attempt to hide their actions.

Common techniques include:

- timestomping
- log deletion
- artifact removal

Investigators must validate evidence using multiple artifact sources.

Core DFIR Workflow

A simplified forensic investigation workflow:

1. Acquire memory image
2. Acquire disk image
3. Identify partitions
4. Analyze filesystem
5. Extract artifacts
6. Recover files
7. Build timeline
8. Reconstruct the incident

This structured process helps investigators understand what happened.

Key Takeaways

Important lessons from digital forensics:

- never trust a single artifact
- correlate multiple sources
- verify timestamps
- preserve evidence integrity

Careful analysis of system artifacts allows investigators to reconstruct events.

Závěr

Děkujeme za pozornost!

Ondřej Šrámek

Czechitas · DA Kybernetická bezpečnost · 2026

czechitas

Zpětná vazba

Budeme rádi za vaši zpětnou vazbu!

czechitas