WINDOWS AND LINUX ARTIFACT DEEP DIVE

A CONSOLIDATED FORENSICS HANDBOOK

DEPI GRADUATION PROJECT

FORENSICS INVESTIGATION TRACK

Prepared By:

ABDELRAHMAN MOHAMED

Windows and Linux Artifact Deep Dive

Contents:

About The Project

This work is part of the forensic investigation track within the **Digital Egypt Pioneers Initiative (DEPI)** and reflects a collaborative effort carried out with a professional, team-oriented approach.

The project aims to study and analyze digital evidence across both Windows and Linux operating systems.

This work represents our **graduation project** for the **DEPI Digital Forensics Track**, demonstrating our ability to apply real forensic methodologies, analyze system artifacts, and work effectively as a coordinated investigation team.

The work is organized into a structured four-week workflow, where each week focuses on a specific forensic domain.

The four-week structure of the project is as follows:

- **Week 1 – Windows Forensics**

Deep investigation of Windows volatile and non-volatile artifacts, including memory acquisition, registry analysis, jump lists, shellbags, and browser data.

- **Week 2 – Linux Forensics**

Forensic examination of a Linux environment, covering EXT4 filesystem analysis, log inspection, deleted history recovery, and RAM acquisition.

- **Week 3 – Comparative Forensics**

Cross-platform comparison of Windows and Linux artifacts, focusing on timestamp formats, user activity trails, evidence value, and tool compatibility.

- **Week 4 – Consolidated Reporting & Final Review**

Compilation of all findings into a unified forensic guide, including legal considerations and a complete professional analysis of both operating systems.

Part 1: Windows Forensics

1. Introduction

Windows forensics involves identifying, collecting, preserving, and analyzing digital evidence from Windows systems to understand security incidents or unauthorized activities. It focuses on examining system artifacts such as **event logs, registry hives, memory dumps, prefetch files, and user data** to reconstruct user actions and determine the cause of an incident.

In a forensic investigation, data can be classified into **volatile** and **non-volatile** categories based on how long it remains available after a system shutdown or reboot. Understanding this distinction helps investigators determine what to collect first and how to properly preserve evidence.

In Windows forensics, **volatile data** (such as running processes, network connections, and RAM contents) disappears once the system is powered off, so it must be collected **first** using tools like **DumpIt** or **FTK Imager (live capture)**.

Non-volatile data (such as logs, registry files, prefetch files, and documents) remains stored on disk even after shutdown and can be collected later using tools like **FTK Imager** or **Autopsy**.

Key Goals of Windows Forensics

- Detecting malicious activity such as malware execution or privilege escalation.
- Recovering volatile and non-volatile data such as memory content, deleted files, and registry keys.
- Preserving evidence integrity by using trusted forensic tools and maintaining a proper chain of custody.

2. Collecting Volatile Information

Memory forensics starts with the acquisition and analysis of volatile data, including running processes, network connections, and other live system artifacts stored in RAM. Trusted tools like DumpIt or FTK Imager should be executed from external media to prevent any alteration of the target system. The captured memory image is then analyzed using frameworks such as Volatility to extract and interpret forensic evidence accurately.

2.1 Memory Acquisition Using Dumpit

Why Dumpit?

DumpIt is a reliable, easy-to-use tool that quickly captures a system's full RAM while maintaining forensic integrity.

Tool Link: https://github.com/h4sh5/DumpIt-mirror

Command: .\DumpIt.exe /T RAW /O "E:\Memory Acquisition\memory_acquisition.raw"

```
PS E:\Tools\DumpIt> .\DumpIt.exe /T RAW /O "E:\Memory Acquisition\memory_acquisition.raw"

 DumpIt 3.0.20171228.1
 Copyright (C) 2007 - 2017, Matthieu Suiche <http://www.msuiche.net>
 Copyright (C) 2012 - 2014, MoonSols Limited <http://www.moonsols.com>
 Copyright (C) 2015 - 2017, Comae Technologies FZE <http://www.comae.io>

   WARNING: RAW memory snapshot files are considered obsolete and as a legacy format.

   Destination path:           \??\E:\Memory Acquisition\memory_acquisition.bin

   Computer name:              DESKTOP-PM94U06


   --> Proceed with the acquisition ? [y/n] y

   [+] Information:
   Dump Type:                   Raw Memory Dump


   [+] Machine Information:
   Windows version:             10.0.19045
   MachineId:                   3B124D56-8791-B241-A709-BC5EAAAD0138
   TimeStamp:                   134043275483325431
   Cr3:                         0x1ad002
   KdCopyDataBlock:             0xfffff80001d2dc18
   KdDebuggerData:              0xfffff8000241fb20
   KdpDataBlockEncoded:         0xfffff8000246fbb8

   Current date/time:          [2025-10-07 (YYYY-MM-DD) 16:19:08 (UTC)]
   + Processing... Done.

   Acquisition finished at:    [2025-10-07 (YYYY-MM-DD) 16:22:32 (UTC)]
   Time elapsed:               3:24 minutes:seconds (204 secs)

   Created file size:           2147483648 bytes (2048 Mb)
   Total physical memory size:  2047 Mb

   NtStatus (troubleshooting):   0x00000000
   Total of written pages:        524045
   Total of inacessible pages:         0
   Total of accessible pages:     524045

   SHA-256: 53A79760D652F116867669664DC9D06E71EBCA6FCB5D08DFF981642576388158

   JSON path:                  E:\Memory Acquisition\memory_acquisition.json
```

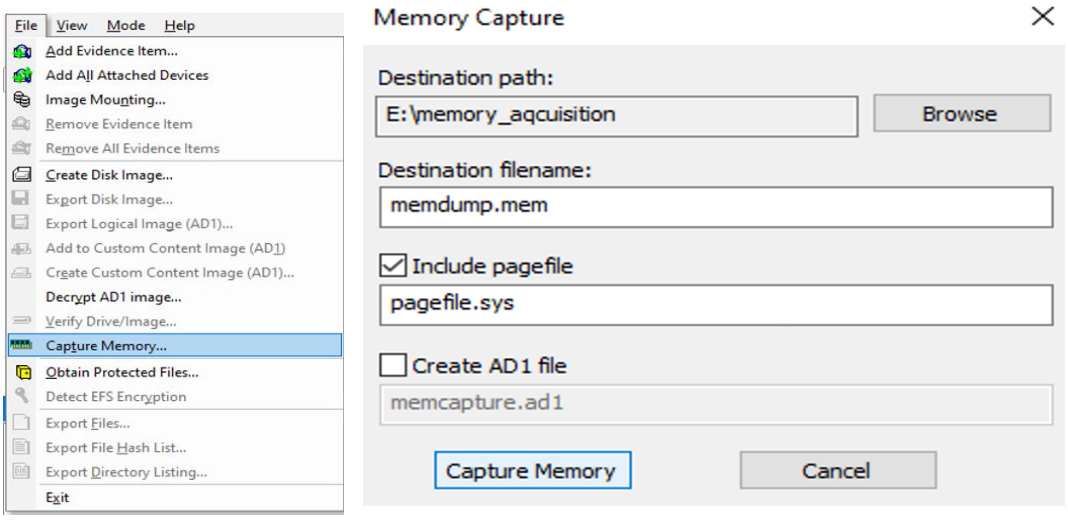
Name	Date modified	Type	Size
memory_acquisition.raw	10/7/2025 7:22 PM	RAW File	2,097,152 KB
memory_acquisition.json	10/7/2025 7:22 PM	JSON File	2 KB

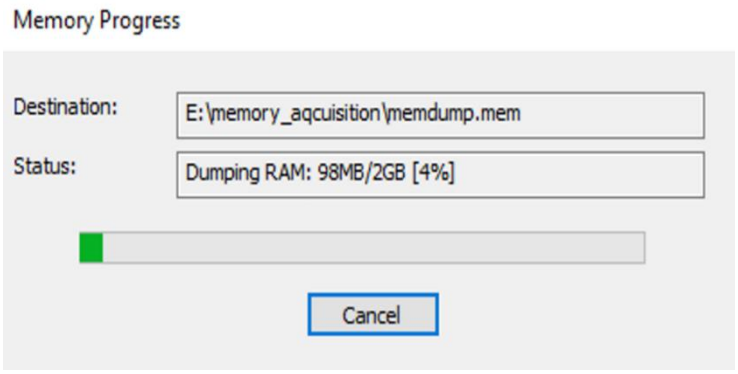
2.2 Memory Acquisition Using FTK Imager

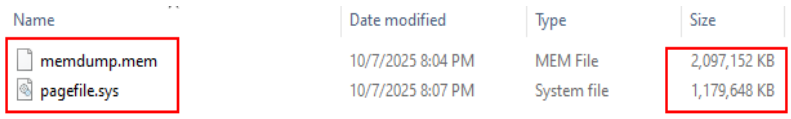
Why FTK Imager?

FTK Imager can capture the pagefile.sys, which is stored on the hard drive. DumpIt only captures RAM and cannot access disk files like the paging file.

Tool Link: https://www.exterro.com/ftk-product-downloads







2.3 Memory Analysis Using Volatility

After completing the memory acquisition process using tools like DumpIt or FTK Imager, the next step is to analyze the captured memory image to identify evidence of malicious activity or system behavior using volatility.

Why Volatility?

Volatility is a trusted, open-source memory forensics framework that enables investigators to extract and analyze information from captured memory images reliably. It operates using a plugin-based architecture, where each plugin performs a specific analysis task such as listing processes, detecting malware, or scanning network connections.

Volatility Versions:

There are two main versions of the Volatility Framework used for memory forensics:

- **Volatility 2:**
  Organized under the volatility/plugins/ directory.
  It allows adding custom plugins easily and is widely used for classic memory analysis tasks.
- **Volatility 3:**
  Plugins are structured as Python modules under volatility3/plugins/.
  This version offers improved performance, enhanced cross-platform compatibility, and enhanced code modularity.

Official Repositories:

- 🔗 Volatility 2
- 🔗 Volatility 3

2.4 Practical Volatility Demo

We will perform a Windows memory forensic investigation using a previously acquired memory image. The objective is to analyze the captured RAM for evidence of malicious activity, such as suspicious processes, network connections, or injected code. In this Practical demo, we will use Volatility v3 on SIFT OS.

Command: `python3 vol.py -f ../MemoryDump.mem windows.info` (Displays basic system information)

```
Kernel Base     0xf8076221a000
DTB     0x1ad000
Symbols file:///home/sansforensics/Desktop/memory_acquisition/
Is64Bit True
IsPAE   False
layer_name      0 WindowsIntel32e
memory_layer    1 FileLayer
KdVersionBlock  0xf80762e29398
Major/Minor     15.19041
MachineType     34404
KeNumberProcessors      4
SystemTime      2023-05-21 23:02:39+00:00
NtSystemRoot    C:\Windows
NtProductType   NtProductWinNt
NtMajorVersion  10
NtMinorVersion  0
PE MajorOperatingSystemVersion  10
PE MinorOperatingSystemVersion  0
PE Machine      34404
PE TimeDateStamp        Wed Jun 28 04:14:26 1995
```

Command: `Python3 vol.py -f ../MemoryDump.mem windows.pslist` (Lists all running processes)

```
Progress:  100.00               PDB scanning finished
PID     PPID    ImageFileName   Offset(V)       Threads Handles SessionId       Wow64   CreateTime      ExitTime        File output

4       0       System  0xad8185883180  157     -       N/A     False   2023-05-21 22:27:10.000000 UTC  N/A     Disabled
108     4       Registry        0xad81858f2080  4       -       N/A     False   2023-05-21 22:26:54.000000 UTC  N/A     Disabled
332     4       smss.exe        0xad81860dc040  2       -       N/A     False   2023-05-21 22:27:10.000000 UTC  N/A     Disabled
452     444     csrss.exe       0xad81861cd080  12      -       0       False   2023-05-21 22:27:22.000000 UTC  N/A     Disabled
528     520     csrss.exe       0xad8186f1b140  14      -       1       False   2023-05-21 22:27:25.000000 UTC  N/A     Disabled
552     444     wininit.exe     0xad8186f2b080  1       -       0       False   2023-05-21 22:27:25.000000 UTC  N/A     Disabled
588     520     winlogon.exe    0xad8186f450c0  5       -       1       False   2023-05-21 22:27:25.000000 UTC  N/A     Disabled
676     552     services.exe    0xad8186f4d080  7       -       0       False   2023-05-21 22:27:29.000000 UTC  N/A     Disabled
696     552     lsass.exe       0xad8186fc6080  10      -       0       False   2023-05-21 22:27:29.000000 UTC  N/A     Disabled
824     676     svchost.exe     0xad818761d240  22      -       0       False   2023-05-21 22:27:32.000000 UTC  N/A     Disabled
852     552     fontdrvhost.ex  0xad818761b0c0  5       -       0       False   2023-05-21 22:27:33.000000 UTC  N/A     Disabled
860     588     fontdrvhost.ex  0xad818761f140  5       -       1       False   2023-05-21 22:27:33.000000 UTC  N/A     Disabled
952     676     svchost.exe     0xad81876802c0  12      -       0       False   2023-05-21 22:27:36.000000 UTC  N/A     Disabled
1016    588     dwm.exe 0xad81876e4340  15      -       1       False   2023-05-21 22:27:38.000000 UTC  N/A     Disabled
448     676     svchost.exe     0xad8187721240  54      -       0       False   2023-05-21 22:27:41.000000 UTC  N/A     Disabled
752     676     svchost.exe     0xad8187758280  21      -       0       False   2023-05-21 22:27:43.000000 UTC  N/A     Disabled
1012    676     svchost.exe     0xad818774c080  19      -       0       False   2023-05-21 22:27:43.000000 UTC  N/A     Disabled
1196    676     svchost.exe     0xad81877972c0  34      -       0       False   2023-05-21 22:27:46.000000 UTC  N/A     Disabled
1280    4       MemCompression  0xad8187835080  62      -       N/A     False   2023-05-21 22:27:49.000000 UTC  N/A     Disabled
1376    676     svchost.exe     0xad81878020c0  15      -       0       False   2023-05-21 22:27:49.000000 UTC  N/A     Disabled
1448    676     svchost.exe     0xad818796c2c0  30      -       0       False   2023-05-21 22:27:52.000000 UTC  N/A     Disabled
1496    676     svchost.exe     0xad81879752c0  12      -       0       False   2023-05-21 22:27:52.000000 UTC  N/A     Disabled
1644    676     svchost.exe     0xad8187a112c0  6       -       0       False   2023-05-21 22:27:58.000000 UTC  N/A     Disabled
1652    676     svchost.exe     0xad8187a2d2c0  10      -       0       False   2023-05-21 22:27:58.000000 UTC  N/A     Disabled
```

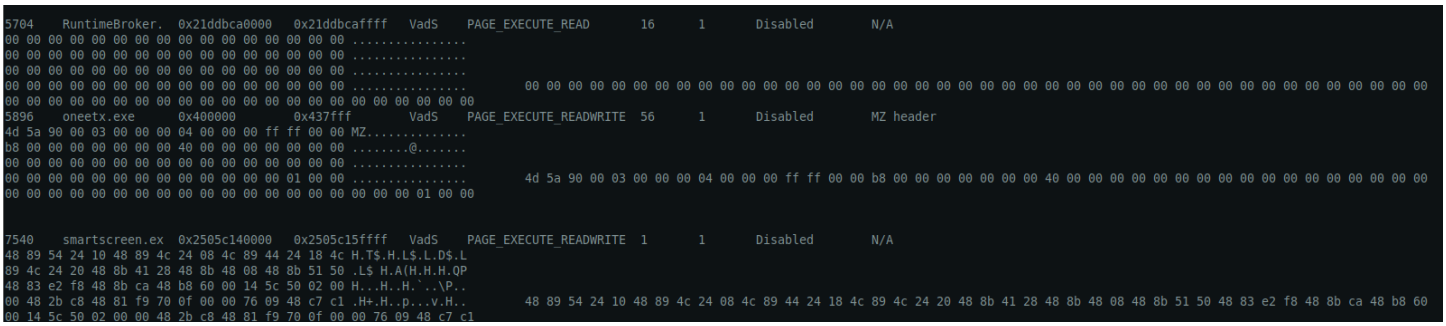
Command: python3 vol.py -f ../MemoryDump.mem windows.pstree (Shows processes in a tree format, displaying parent/child relationships)

```
PID     PPID    ImageFileName   Offset(V)       Threads Handles SessionId       Wow64   CreateTime      ExitTime        Audit   Cmd     Path

4       0       System  0xad8185883180  157     -       N/A     False   2023-05-21 22:27:10.000000 UTC  N/A     -       -       -
* 1280  4       MemCompression  0xad8187835080  62      -       N/A     False   2023-05-21 22:27:49.000000 UTC  N/A     MemCompression  -       -
* 108   4       Registry        0xad81858f2080  4       -       N/A     False   2023-05-21 22:26:54.000000 UTC  N/A     Registry        -       -
* 332   4       smss.exe        0xad81860dc040  2       -       N/A     False   2023-05-21 22:27:10.000000 UTC  N/A     \Device\HarddiskVolume3\Windows\System32\smss.exe       -       -
452     444     csrss.exe       0xad81861cd080  12      -       0       False   2023-05-21 22:27:22.000000 UTC  N/A     \Device\HarddiskVolume3\Windows\System32\csrss.exe      -       -
528     520     csrss.exe       0xad8186f1b140  14      -       1       False   2023-05-21 22:27:25.000000 UTC  N/A     \Device\HarddiskVolume3\Windows\System32\csrss.exe
552     444     wininit.exe     0xad8186f2b080  1       -       0       False   2023-05-21 22:27:25.000000 UTC  N/A     \Device\HarddiskVolume3\Windows\System32\wininit.exe    -       -
* 696   552     lsass.exe       0xad8186fc6080  10      -       0       False   2023-05-21 22:27:29.000000 UTC  N/A     \Device\HarddiskVolume3\Windows\System32\lsass.exe      C:\Windows\system32\ls
ass.exe   C:\Windows\system32\lsass.exe
* 676   552     services.exe    0xad8186f4d080  7       -       0       False   2023-05-21 22:27:29.000000 UTC  N/A     \Device\HarddiskVolume3\Windows\System32\services.exe   C:\Windows\system32\se
rvices.exe      C:\Windows\system32\services.exe
** 4228 676     SearchIndexer.  0xad818ce06240  15      -       0       False   2023-05-21 22:31:27.000000 UTC  N/A     \Device\HarddiskVolume3\Windows\System32\SearchIndexer.exe      C:\Windows\sys
tem32\SearchIndexer.exe /Embedding      C:\Windows\system32\SearchIndexer.exe
** 8708 676     svchost.exe     0xad818d431080  5       -       0       False   2023-05-21 22:57:33.000000 UTC  N/A     \Device\HarddiskVolume3\Windows\System32\svchost.exe    -       -
** 5136 676     SecurityHealth  0xad818d374280  7       -       0       False   2023-05-21 22:32:01.000000 UTC  N/A     \Device\HarddiskVolume3\Windows\System32\SecurityHealthService.exe      -
      -
** 2200 676     VGAuthService.  0xad81896b3300  2       -       0       False   2023-05-21 22:28:19.000000 UTC  N/A     \Device\HarddiskVolume3\Program Files\VMware\VMware Tools\VMware VGAuth\VGAuth
Service.exe     -       -
** 3608 676     svchost.exe     0xad818d07a080  3       -       0       False   2023-05-21 22:41:28.000000 UTC  N/A     \Device\HarddiskVolume3\Windows\System32\svchost.exe    -       -
** 2076 676     svchost.exe     0xad8187b94080  10      -       0       False   2023-05-21 22:28:19.000000 UTC  N/A     \Device\HarddiskVolume3\Windows\System32\svchost.exe    C:\Windows\System32\sv
chost.exe -k utcsvc -p    C:\Windows\System32\svchost.exe
** 1448 676     svchost.exe     0xad818796c2c0  30      -       0       False   2023-05-21 22:27:52.000000 UTC  N/A     \Device\HarddiskVolume3\Windows\System32\svchost.exe    C:\Windows\System32\sv
chost.exe -k NetworkService -p    C:\Windows\System32\svchost.exe
** 1064 676     svchost.exe     0xad8189d7c2c0  15      -       1       False   2023-05-21 22:30:09.000000 UTC  N/A     \Device\HarddiskVolume3\Windows\System32\svchost.exe    C:\Windows\System32\sv
chost.exe -k UnistackSvcGroup     C:\Windows\system32\svchost.exe
** 6696 676     svchost.exe     0xad818c532080  8       -       0       False   2023-05-21 22:34:07.000000 UTC  N/A     \Device\HarddiskVolume3\Windows\System32\svchost.exe    -       -
** 1196 676     svchost.exe     0xad81877972c0  34      -       0       False   2023-05-21 22:27:46.000000 UTC  N/A     \Device\HarddiskVolume3\Windows\System32\svchost.exe    C:\Windows\System32\sv
chost.exe -k LocalService -p      C:\Windows\system32\svchost.exe
** 1840 676     spoolsv.exe     0xad8187acb200  10      -       0       False   2023-05-21 22:28:03.000000 UTC  N/A     \Device\HarddiskVolume3\Windows\System32\spoolsv.exe    -       -
** 952  676     svchost.exe     0xad81876802c0  12      -       0       False   2023-05-21 22:27:36.000000 UTC  N/A     \Device\HarddiskVolume3\Windows\System32\svchost.exe    C:\Windows\system32\sv
chost.exe -k RPCSS -p     C:\Windows\system32\svchost.exe
** 824  676     svchost.exe     0xad818761d240  22      -       0       False   2023-05-21 22:27:32.000000 UTC  N/A     \Device\HarddiskVolume3\Windows\System32\svchost.exe    C:\Windows\system32\sv
chost.exe -k DcomLaunch -p        C:\Windows\system32\svchost.exe
*** 7312       824     ApplicationFra  0xad818e84f300  10      -       1       False   2023-05-21 22:35:44.000000 UTC  N/A     \Device\HarddiskVolume3\Windows\System32\ApplicationFrameHost.exe
      C:\Windows\system32\ApplicationFrameHost.exe -Embedding C:\Windows\system32\ApplicationFrameHost.exe
*** 4116       824     RuntimeBroker.  0xad818cd93300  3       -       1       False   2023-05-21 22:31:24.000000 UTC  N/A     \Device\HarddiskVolume3\Windows\System32\RuntimeBroker.exe      -
      -
*** 5656       824     RuntimeBroker.  0xad81876e8080  0       -       1       False   2023-05-21 21:58:19.000000 UTC  2023-05-21 22:02:01.000000 UTC  \Device\HarddiskVolume3\Windows\System32\Runti
meBroker.exe    -       -
*** 2332       824     TiWorker.exe    0xad818e780080  4       -       0       False   2023-05-21 22:58:13.000000 UTC  N/A     \Device\HarddiskVolume3\Windows\WinSxS\amd64_microsoft-windows-servici
ngstack_31bf3856ad364e35_10.0.19041.1940_none_7dd80d767cb5c7b0\TiWorker.exe
*** 7336       824     RuntimeBroker.  0xad818e8bb080  2       -       1       False   2023-05-21 22:11:39.000000 UTC  N/A     \Device\HarddiskVolume3\Windows\System32\RuntimeBroker.exe      -
```

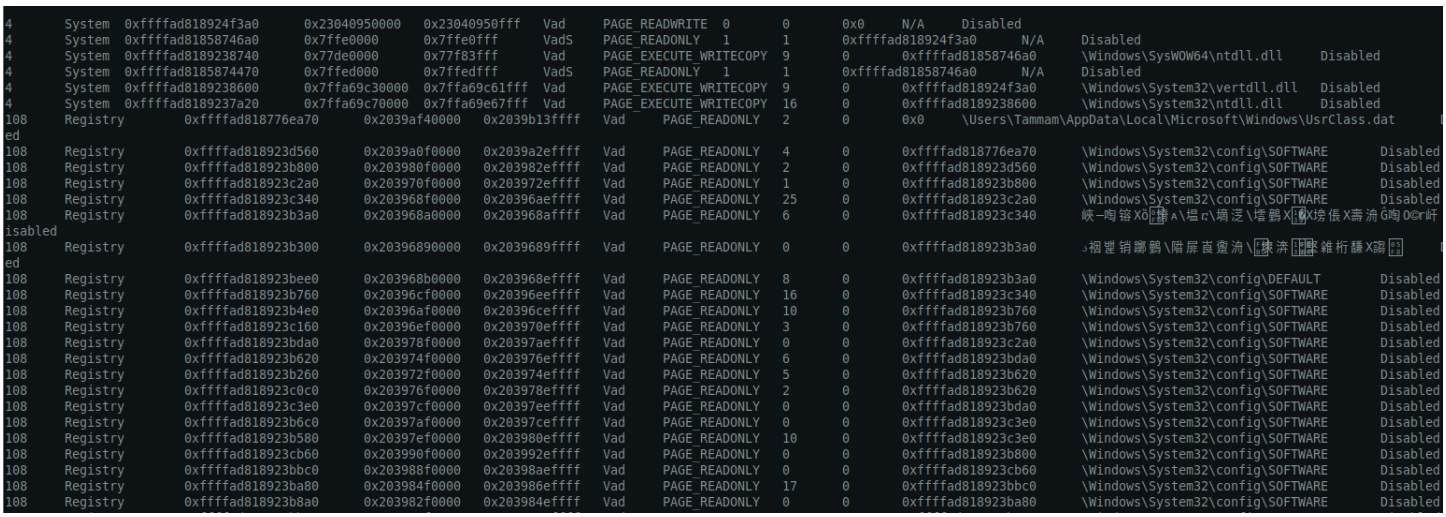
Command: python3 vol.py -f ../MemoryDump.mem windows.netscan (Scans memory for network artifacts such as open ports, connections, and sockets)

```
Offset  Proto   LocalAddr       LocalPort       ForeignAddr     ForeignPort     State   PID     Owner   Created

0xad81861e2310  TCPv4   0.0.0.0 49668   0.0.0.0 0       LISTENING       1840    spoolsv.exe     2023-05-21 22:28:09.000000 UTC
0xad81861e2310  TCPv6   ::      49668   ::      0       LISTENING       1840    spoolsv.exe     2023-05-21 22:28:09.000000 UTC
0xad81861e2470  TCPv4   0.0.0.0 5040    0.0.0.0 0       LISTENING       1196    svchost.exe     2023-05-21 22:30:31.000000 UTC
0xad81861e2730  TCPv4   0.0.0.0 135     0.0.0.0 0       LISTENING       952     svchost.exe     2023-05-21 22:27:36.000000 UTC
0xad81861e2b50  TCPv4   0.0.0.0 49665   0.0.0.0 0       LISTENING       552     wininit.exe     2023-05-21 22:27:36.000000 UTC
0xad81861e2b50  TCPv6   ::      49665   ::      0       LISTENING       552     wininit.exe     2023-05-21 22:27:36.000000 UTC
0xad81861e2e10  TCPv4   0.0.0.0 49665   0.0.0.0 0       LISTENING       552     wininit.exe     2023-05-21 22:27:36.000000 UTC
0xad81861e3230  TCPv4   0.0.0.0 49664   0.0.0.0 0       LISTENING       696     lsass.exe       2023-05-21 22:27:36.000000 UTC
0xad81861e3390  TCPv4   0.0.0.0 135     0.0.0.0 0       LISTENING       952     svchost.exe     2023-05-21 22:27:36.000000 UTC
0xad81861e3390  TCPv6   ::      135     ::      0       LISTENING       952     svchost.exe     2023-05-21 22:27:36.000000 UTC
0xad81861e34f0  TCPv4   0.0.0.0 49664   0.0.0.0 0       LISTENING       696     lsass.exe       2023-05-21 22:27:36.000000 UTC
0xad81861e34f0  TCPv6   ::      49664   ::      0       LISTENING       696     lsass.exe       2023-05-21 22:27:36.000000 UTC
0xad81861e37b0  TCPv4   0.0.0.0 49666   0.0.0.0 0       LISTENING       1012    svchost.exe     2023-05-21 22:27:49.000000 UTC
0xad81861e37b0  TCPv6   ::      49666   ::      0       LISTENING       1012    svchost.exe     2023-05-21 22:27:49.000000 UTC
0xad81861e3910  TCPv4   0.0.0.0 49667   0.0.0.0 0       LISTENING       448     svchost.exe     2023-05-21 22:27:58.000000 UTC
0xad81861e3910  TCPv6   ::      49667   ::      0       LISTENING       448     svchost.exe     2023-05-21 22:27:58.000000 UTC
0xad81861e3a70  TCPv4   0.0.0.0 49668   0.0.0.0 0       LISTENING       1840    spoolsv.exe     2023-05-21 22:28:09.000000 UTC
0xad81861e3bd0  TCPv4   0.0.0.0 49666   0.0.0.0 0       LISTENING       1012    svchost.exe     2023-05-21 22:27:49.000000 UTC
0xad81861e3e90  TCPv4   0.0.0.0 49667   0.0.0.0 0       LISTENING       448     svchost.exe     2023-05-21 22:27:58.000000 UTC
0xad818662ecb0  TCPv4   0.0.0.0 445     0.0.0.0 0       LISTENING       4       System  2023-05-21 22:29:04.000000 UTC
0xad818662ecb0  TCPv6   ::      445     ::      0       LISTENING       4       System  2023-05-21 22:29:04.000000 UTC
0xad818662f390  TCPv4   0.0.0.0 7680    0.0.0.0 0       LISTENING       5476    svchost.exe     2023-05-21 22:58:09.000000 UTC
0xad818662f390  TCPv6   ::      7680    ::      0       LISTENING       5476    svchost.exe     2023-05-21 22:58:09.000000 UTC
```

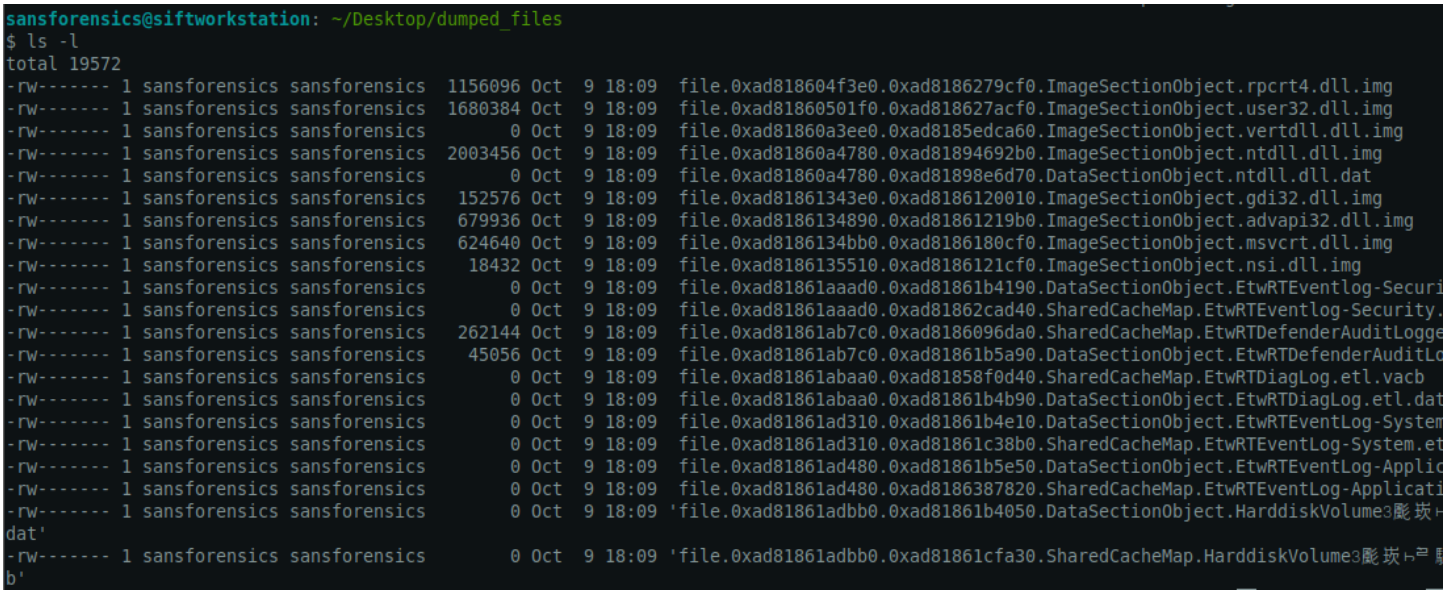
Command: `python3 vol.py -f ../MemoryDump.mem windows.malfind` (Detects injected code and hidden malware)



Command: `python3 vol.py -f ../MemoryDump.mem windows.vadinfo` (Displays detailed Virtual Address Descriptor)



Command: `python3 vol.py -f ../MemoryDump.mem -o /home/sansforensics/Desktop/dumped_files/ windows.dumpfiles` (Extracts files directly from memory into a directory)



3. Collecting Non-volatile Information

Non-volatile data acquisition focuses on collecting artifacts stored permanently on disk, which remain accessible even after system shutdown or reboot.
These artifacts play a vital role in digital forensics, providing long-term evidence of user activity, system configuration, and persistence mechanisms that may not be visible in volatile memory. Common examples include Windows Registry hives, event logs, file system metadata, browser histories, and program execution traces.

The primary goal of non-volatile data acquisition is to preserve information, ensuring that evidence is not modified or corrupted during collection. Proper acquisition enables investigators to perform **timeline reconstruction**, **user attribution**, and **incident response correlation** with accuracy and credibility in court or enterprise reports.

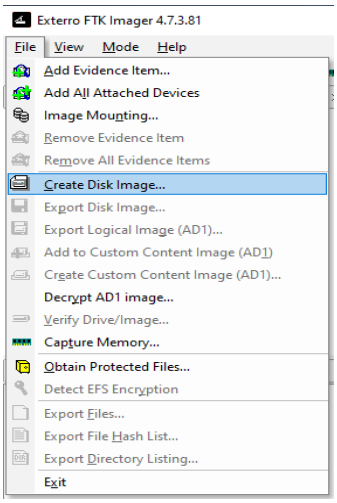
To achieve this, forensic practitioners rely on imaging tools such as **FTK Imager** or the command-line utility **dd** to perform **bit-for-bit (sector-by-sector)** copies of hard drives and removable media.

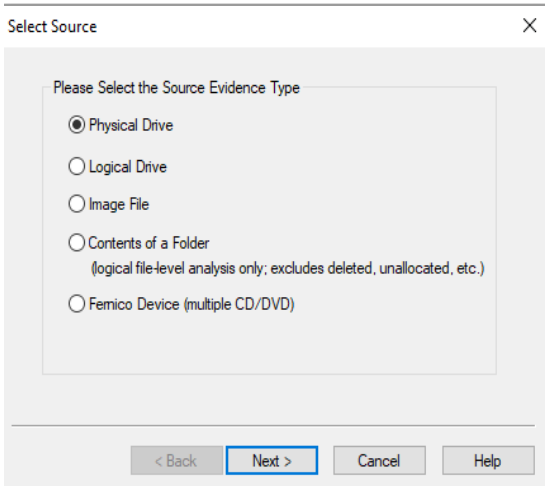
These tools maintain **data integrity** by generating **cryptographic hash values** (MD5, SHA-1, SHA-256) before and after acquisition, ensuring that the forensic image is a replica of the original media.

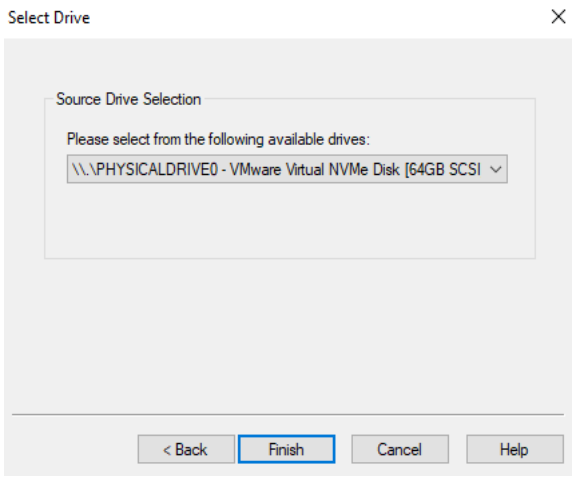
Once collected, non-volatile artifacts can be examined using tools such as RegRipper and KAPE, as well as Eric Zimmermann's suite of forensic utilities (RECmd, MFTECmd, AppCompatCacheParser) to uncover user activity, persistence mechanisms, and indicators of compromise within the Windows environment.

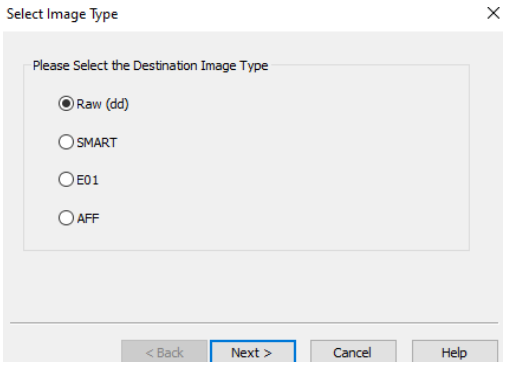
3.1 Disk Acquisition Using FTK Imager

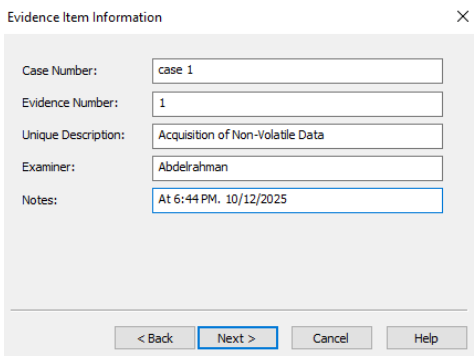
When using FTK Imager, it’s best to install it on an external device, such as a USB drive, to preserve any data that may change during acquisition.

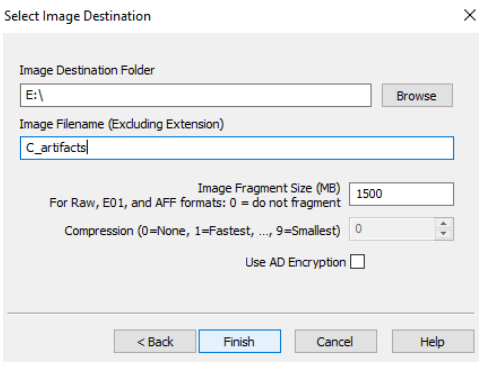




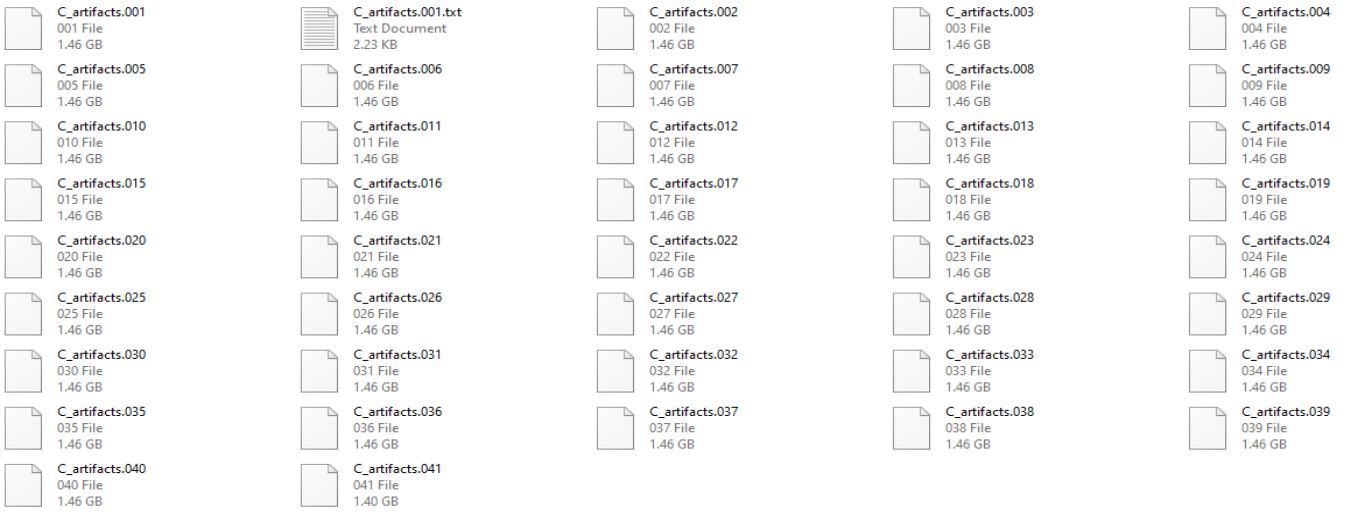








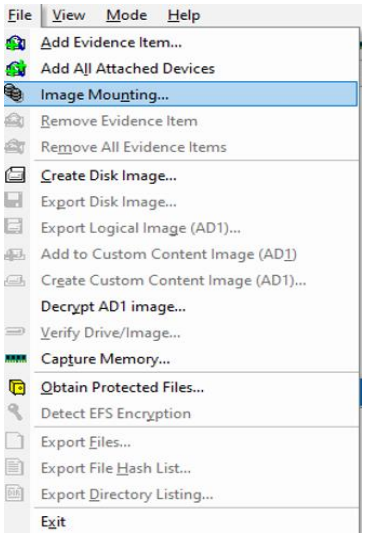
The output of the acquisition will be split into small files based on the size of the image fragments.

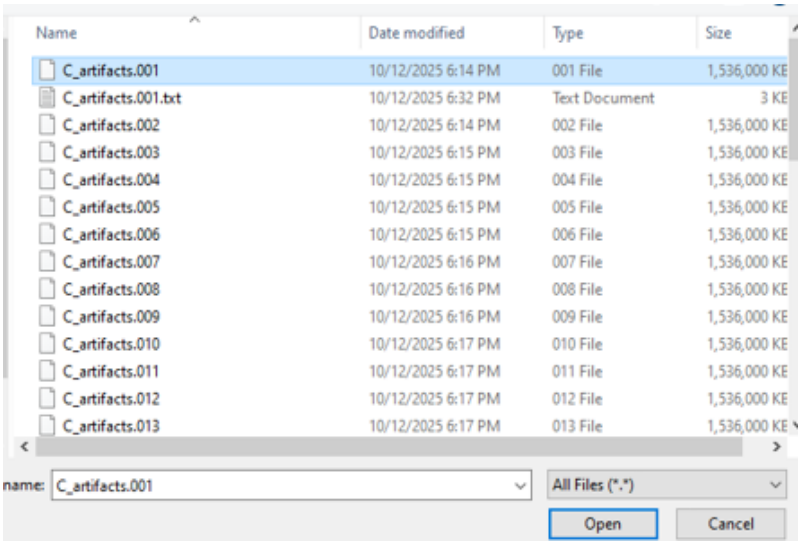


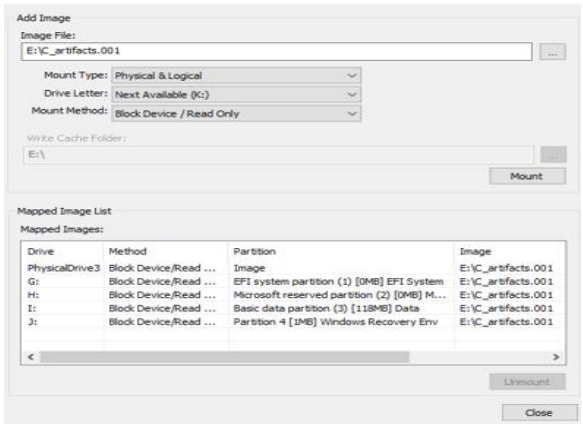
3.2 Mounting Disks Using FTK Imager

After the previous steps, it's time to mount this disk on a different Windows machine to start the investigation.

When mounting the first file, the whole disk will be mounted.







Mounting the acquired image in FTK Imager revealed multiple logical volumes corresponding to the original drive's partitions (EFI, Recovery, and OS). Among these, the **C:** partition was identified as the primary operating system volume containing user data, system configurations, and application artifacts. Therefore, the forensic analysis focused on this partition, as it holds the most relevant evidence for reconstructing user activity and system behavior.

In the end, obtaining a full disk acquisition provides a complete forensic image of the system, allowing for extensive investigations. From this image, an examiner can analyze both volatile and non-volatile artifacts, recover deleted data, examine user activity, review system logs, and reconstruct events to support incident analysis or digital evidence correlation.

4. Analyzing Different Windows Artifacts

4.1 JumpLists

Jump Lists are available on all taskbar icons for applications, allowing users quick access to recently opened files, folders, or common tasks related to that application.

Each application maintains its own Jump List, which is automatically updated as the user interacts with the system.

Type	Path	Description
Automatic Jump Lists	C:\Users\%USERNAME%\AppData\Roaming\Microsoft\Windows \Recent\AutomaticDestinations	Automatically created by Windows when users open or save files using an application. Each .automaticDestinations-ms file corresponds to a specific app.
Custom Jump Lists	C:\Users\%USERNAME%\AppData\Roaming\Microsoft\Windows \Recent\CustomDestinations	Created when users manually pin items or when applications define custom Jump Lists. Stored as .customDestinations-ms files.
Pinned Taskbar Shortcuts	C:\Users\%USERNAME%\AppData\Roaming\Microsoft\ Internet Explorer\Quick Launch\User Pinned\TaskBar	Stores .lnk files (shortcuts) representing applications pinned to the taskbar. Useful for determining user intent or application preference.
Registry Path	HKCU\Software\Microsoft\Windows\CurrentVersion\ Explorer\Taskband → Favorites, FavoritesResolve	Registry keys storing taskbar pinned items and their resolved file paths.

Forensic Value

- Application Usage Evidence – Determines which programs a user frequently accesses.
- File Access History – Identifies files recently opened by the user.
- Timestamps – Useful for building activity timelines.
- Recovery of Deleted Evidence – Even if files are deleted, their Jump List traces often remain.
- User Intent – Reveals what applications or documents the user purposefully interacted with.

Practical Section

In this section, we will analyze **Jump Lists** using **JLECmd.exe**, a powerful Windows artifact analysis tool developed by **Eric Zimmerman**. The analysis will be performed on the **C:** partition from the acquired image, as it contains user activity artifacts generated by the Windows operating system.

Tools:

- Jump Lists: https://github.com/EricZimmerman/JLECmd
- Timeline: https://download.ericzimmermanstools.com/net9/TimelineExplorer.zip

Path: C:\Users\%USERNAME%\AppData\Roaming\Microsoft\Windows\Recent\.....

Automatic Destinations are automatically created by Windows when users open or save files using an application. Each .automaticDestinations-ms file corresponds to a specific app.

Command:

```
JLECmd.exe -d  "C:\Users\demo\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations" --csv F:\JumpLists_dumping
```

We can also dump CustomDestinations or User Pinned TaskBar for different purposes.

After that, we can open the file with Timeline Explorer.

Count	Mac Address	Path	Target Created	Target Modified	Target Accessed
1		F:\JumpLists_dumping\20251013175310_CustomDestinations.csv	2025-10-13 17:53:14	2025-10-13 17:53:16	2025-10-13 17:53:16
2		F:\JumpLists_dumping	2025-10-13 17:48:22	2025-10-13 17:48:22	2025-10-13 17:48:22
1	00:0c:29:ad:01:38	E:\C_artifacts.001.txt	2025-10-13 17:03:33	2025-10-13 17:03:33	2025-10-13 17:03:33
1	00:0c:29:ad:01:38	E:\C_artifacts.001	2025-10-13 16:50:32	2025-10-13 16:50:59	2025-10-13 17:04:25
1	00:0c:29:ad:01:38	C:\Users\demo\Pictures\Screenshots\Screenshot (28).png	2025-10-13 15:20:50	2025-10-13 15:20:50	2025-10-13 15:20:50
1	00:0c:29:ad:01:38	C:\Users\demo\Pictures\Screenshots\Screenshot (27).png	2025-10-13 15:17:27	2025-10-13 15:17:27	2025-10-13 15:17:27
1	00:0c:29:ad:01:38	C:\Users\demo\Pictures\Screenshots\Screenshot (26).png	2025-10-13 15:08:42	2025-10-13 15:08:42	2025-10-13 15:08:42
1	00:0c:29:ad:01:38	C:\Users\demo\Pictures\Screenshots\Screenshot (25).png	2025-10-13 15:07:42	2025-10-13 15:07:42	2025-10-13 15:07:42
1	00:0c:29:ad:01:38	C:\Users\demo\Pictures\Screenshots\Screenshot (24).png	2025-10-13 15:07:39	2025-10-13 15:07:39	2025-10-13 15:07:39
1	00:0c:29:ad:01:38	C:\Users\demo\Pictures\Screenshots\Screenshot (23).png	2025-10-13 15:07:26	2025-10-13 15:07:26	2025-10-13 15:07:26
1	00:0c:29:ad:01:38	C:\Users\demo\Pictures\Screenshots\Screenshot (22).png	2025-10-13 15:07:16	2025-10-13 15:07:16	2025-10-13 15:07:16
2		F:\New folder	2025-10-13 14:48:44	2025-10-13 14:48:44	2025-10-13 14:48:44
1	00:0c:29:ad:01:38	F:\New folder\Screenshot (21).png	2025-10-13 14:48:21	2025-10-13 14:48:22	2025-10-13 14:48:46
1	00:0c:29:ad:01:38	F:\New folder\Screenshot (20).png	2025-10-13 14:48:17	2025-10-13 14:48:18	2025-10-13 14:48:46
1	00:0c:29:ad:01:38	F:\New folder\Screenshot (19).png	2025-10-12 15:44:30	2025-10-12 15:44:32	2025-10-13 14:50:42
2	00:0c:29:ad:01:38	C:\Users\demo\Pictures\Screenshots\Screenshot (14).png	2025-10-12 15:32:23	2025-10-12 15:32:23	2025-10-12 15:34:53
2	00:0c:29:ad:01:38	E:\C_artifacts.001.txt	2025-10-12 15:21:58	2025-10-12 15:21:58	2025-10-12 15:21:58

The parsed data includes critical forensic timestamps—creation, modification, and access times —along with file paths and application identifiers, providing valuable evidence of **user activity, file access patterns, and application execution**.

4.2 Application Compatibility Cache (ShimCache)

The Application Compatibility Cache, commonly known as ShimCache, is a Windows feature designed to help the operating system identify applications that may require specific compatibility settings to operate correctly on newer Windows versions.

From a digital forensics perspective, ShimCache serves as a valuable artifact that records information about executables that have been launched or interacted with on a system. This cache enables investigators to reconstruct program execution timelines, identify potentially malicious binaries, and provide context for system activity during an investigation.

How ShimCache (AppCompatCache) Works

- The ShimCache is first stored in system memory (RAM) during runtime.
- It's part of the AppCompatCache service, which helps Windows manage compatibility information for executables that have run or been accessed.
- This cache is not immediately written to disk; it's held in RAM and only written back to the SYSTEM hive (at `HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\AppCompatCache`) when the system shuts down cleanly.

Forensic Value

The ShimCache provides insight into:

- File name and full path of the executable
- Execution indication (whether the file was likely executed)
- Last modified timestamp (retrieved from the file's metadata at the time of caching)
- Size of the binary file

Note: The ShimCache does not guarantee execution — an entry only indicates that the system became aware of the file.

Practical Section

- Registry Path: `HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\AppCompatCache\AppCompatCache`
- Hive File (Offline Analysis): `C:\Windows\System32\Config\SYSTEM`

Tools:

- AppCompatCacheParser https://download.ericzimmermanstools.com/net9/AmcacheParser.zip
- AmcacheParser https://download.ericzimmermanstools.com/net9/AmcacheParser.zip

Both AmcacheParser and AppCompatCacheParser are Windows forensic tools used to analyze program execution artifacts.

- AmcacheParser focuses on installed or first-run applications, providing detailed metadata such as file paths and hashes.
- AppCompatCacheParser focuses on executables recognized or possibly run by Windows, helping investigators build execution timelines and spot suspicious binaries.

In this practical demo, we will use AppCompatCacheParser to focus on the executables recognized.

Command:

```
AmcacheParser.exe -f C:\Windows\appcompat\Programs\Amcache.hve --csv "F:\AppCompatCacheParser output 2" --csvf AmCachPars.csv
```

Output will be like.

Name	Date modified	Type	Size
AmCachPars_DeviceContainers.csv	10/14/2025 10:17 PM	CSV File	13 KB
AmCachPars_DevicePnps.csv	10/14/2025 10:17 PM	CSV File	97 KB
AmCachPars_DriveBinaries.csv	10/14/2025 10:17 PM	CSV File	112 KB
AmCachPars_DriverPackages.csv	10/14/2025 10:17 PM	CSV File	2 KB
AmCachPars_ShortCuts.csv	10/14/2025 10:17 PM	CSV File	9 KB
AmCachPars_UnassociatedFileEntries.csv	10/14/2025 10:17 PM	CSV File	57 KB

AmCachPars_DeviceContainers.csv: Contains information about device containers, which are logical groupings of hardware devices (such as USB hubs or Bluetooth sets).

AmCachPars_DevicePnps.csv: List Plug and Play (PnP) devices detected on the system.Includes device IDs, friendly names, and timestamps.

AmCachPars_DriveBinaries.csv: Contains data about executables, scripts, and binary files recorded in AmCache. Includes file path, hash, publisher, version, and timestamps.

AmCachPars_DriverPackages.csv: Lists all detected hardware driver packages associated with system devices.

AmCachPars_ShortCuts.csv: Contains entries for Windows shortcut (.lnk) files related to executables in AmCache.

AmCachPars_UnassociatedFileEntries.csv: Lists unassociated executables (.exe, .dll, etc.) that were seen but not linked to a known program or manifest entry.

4.3 Shellbags

ShellBags are a set of Windows registry keys located in NTUSER.dat and USERClass.dat that maintain the view, icon, position, and size of folders when using Windows Explorer.

ShellBags show which folders were accessed, when, and sometimes where (even on external drives or network shares).

Forensic Value:

- **User Activity Tracking:** Records which folders were accessed, including paths, views, and sort preferences.
- **Timestamp Evidence:** Stores creation, modification, and access times that can be used for timeline analysis.
- **Persistence:** Remains on the system even after folders, files, or external drives are deleted or disconnected.
- **External Device Traces:** Can reveal folders opened from USB drives, external disks, or network shares.
- **User Attribution:** Located under the user's registry hive (NTUSER.DAT), it helps link activity to a specific account.

Practical Section:

ShellBags Registry Locations, we focus on two things: NTUSER.DAT and USRCLASS.DAT.

NTUSER.DAT: What the user does and prefers.

Path on disk: C:\Users\<username>\NTUSER.DAT

Registry paths inside NTUSER.DAT:

HKCU\Software\Microsoft\Windows\Shell\BagMRU

HKCU\Software\Microsoft\Windows\Shell\Bags

USRCLASS.DAT: How Windows and Explorer behave for that user.

Path on disk: C:\Users\<username>\AppData\Local\Microsoft\Windows\USRCLASS.DAT

Registry paths inside USRCLASS.DAT:

HKCU\Software\Classes\Local Settings\Software\Microsoft\Windows\Shell\BagMRU

HKCU\Software\Classes\Local

Inside the USERCALSS.DAT, there are two main sybkeys we want to investigate.

- BagMRU
  - Stores a hierarchical list (MRU = Most Recently Used) of folders accessed by the user. Used to identify which folders the user viewed and in what order.
- Bags
  - Contains the view settings (size, position, layout, sort order, icon type) for folders accessed via Explorer. Helps determine how the folder looked when opened

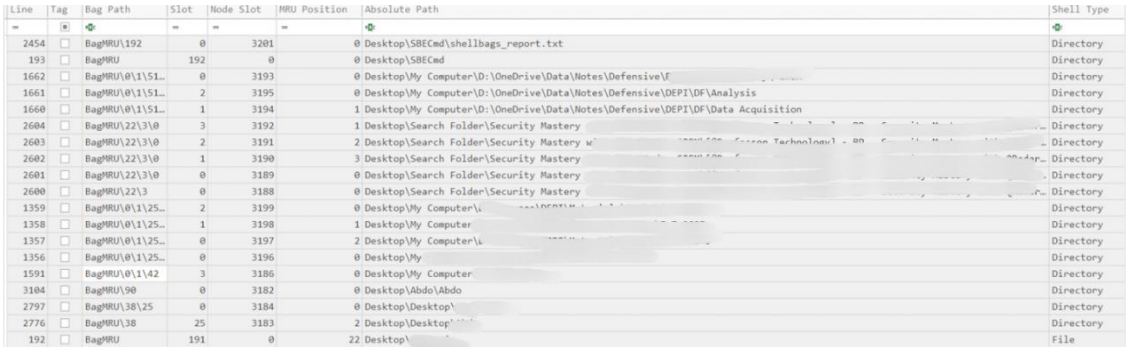
Tools:

- ShellBags Explorer https://ericzimmerman.github.io/#!index.md
- Sbecmd https://ericzimmerman.github.io/#!index.md

Using the sbecmd command like a tool, here is the command we should run: SBECmd.exe -l --csv ./shellbags_report

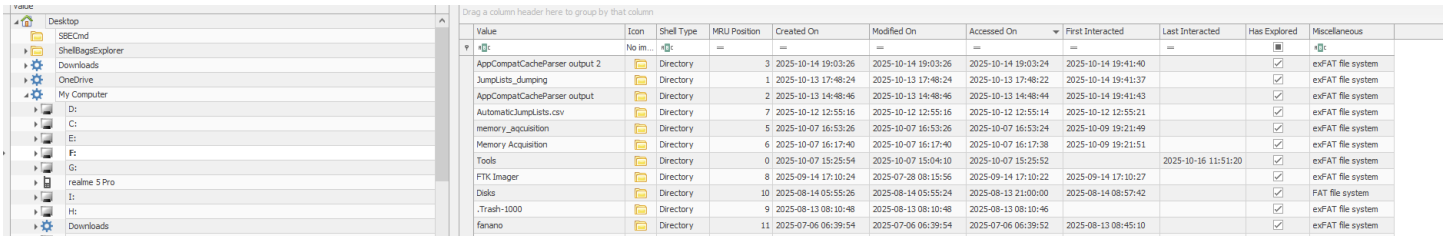
We can open this file with the Timeline tool to make things easy.shellbags_report contains a lot of useful information, like MAC time and path to each file, and more.

```
Totals by bag type

Directory: 2,615
Zip file contents: 167
File: 118
Drive letter: 8
**** Unknown: 11
ZipFileContents: 2
Root folder: GUID: 27
MTP folder: 19
MTP storage: 1
MTP device: 1
Users Files Folder: 8
Variable: 1
Users property view: Drive letter: 4
GUID: Control panel: 13
Control Panel Category: 6
Variable: Users property view: 39
Users property view: 75
Variable: HTTP URI: 6

Finished processing live registry
```



Line	Tag	Bag Path	Slot	Node Slot	MRU Position	Absolute Path	Shell Type
2454		BagMRU\192	0	3201	0	Desktop\SBECmd\shellbags_report.txt	Directory
193		BagMRU	192	0	0	Desktop\SBECmd	Directory
1662		BagMRU\0\1\51…	0	3193	0	Desktop\My Computer\D:\OneDrive\Data\Notes\Defensive\[redacted]	Directory
1661		BagMRU\0\1\51…	2	3195	0	Desktop\My Computer\D:\OneDrive\Data\Notes\Defensive\DEPI\DF\Analysis	Directory
1660		BagMRU\0\1\51…	1	3194	1	Desktop\My Computer\D:\OneDrive\Data\Notes\Defensive\DEPI\DF\Data Acquisition	Directory
2604		BagMRU\22\3\0	3	3192	1	Desktop\Search Folder\Security Mastery [redacted]	Directory
2603		BagMRU\22\3\0	2	3191	2	Desktop\Search Folder\Security Mastery w[redacted]	Directory
2602		BagMRU\22\3\0	1	3190	3	Desktop\Search Folder\Security Mastery [redacted]	Directory
2601		BagMRU\22\3\0	0	3189	0	Desktop\Search Folder\Security Mastery [redacted]	Directory
2600		BagMRU\22\3	0	3188	0	Desktop\Search Folder\Security Mastery [redacted]	Directory
1359		BagMRU\0\1\25…	2	3199	0	Desktop\My Computer\[redacted]	Directory
1358		BagMRU\0\1\25…	1	3198	1	Desktop\My Computer [redacted]	Directory
1357		BagMRU\0\1\25…	0	3197	2	Desktop\My Computer\[redacted]	Directory
1356		BagMRU\0\1\25…	0	3196	0	Desktop\My [redacted]	Directory
1591		BagMRU\0\1\42	3	3186	0	Desktop\My Computer [redacted]	Directory
3104		BagMRU\90	0	3182	0	Desktop\Abdo\Abdo	Directory
2797		BagMRU\38\25	0	3184	0	Desktop\Desktop\[redacted]	Directory
2776		BagMRU\38	25	3183	2	Desktop\Desktop[redacted]	Directory
192		BagMRU	191	0	22	Desktop\[redacted]	File

We can also use the GUI ShellBags Explorer after the installation. We open the tool and start loading the registry hives.



Drag a column header here to group by that column

Value	Icon	Shell Type	MRU Position	Created On	Modified On	Accessed On	First Interacted	Last Interacted	Has Explored	Miscellaneous
AppCompatCacheParser output 2		Directory	3	2025-10-14 19:03:26	2025-10-14 19:03:26	2025-10-14 19:03:24	2025-10-14 19:41:40		☑	exFAT file system
JumpLists_dumping		Directory	1	2025-10-13 17:48:24	2025-10-13 17:48:24	2025-10-13 17:48:22	2025-10-14 19:41:37		☑	exFAT file system
AppCompatCacheParser output		Directory	2	2025-10-13 14:48:46	2025-10-13 14:48:46	2025-10-13 14:48:44	2025-10-14 19:41:43		☑	exFAT file system
AutomaticJumpLists.csv		Directory	7	2025-10-12 12:55:16	2025-10-12 12:55:16	2025-10-12 12:55:14	2025-10-12 12:55:21		☑	exFAT file system
memory_aqcuisition		Directory	5	2025-10-07 16:53:26	2025-10-07 16:53:26	2025-10-07 16:53:24	2025-10-09 19:21:49		☑	exFAT file system
Memory Acquisition		Directory	6	2025-10-07 16:17:40	2025-10-07 16:17:40	2025-10-07 16:17:38	2025-10-09 19:21:51		☑	exFAT file system
Tools		Directory	0	2025-10-07 15:25:54	2025-10-07 15:04:10	2025-10-07 15:25:52		2025-10-16 11:51:20	☑	exFAT file system
FTK Imager		Directory	8	2025-09-14 17:10:24	2025-07-28 08:15:56	2025-09-14 17:10:22	2025-09-14 17:10:27		☑	exFAT file system
Disks		Directory	10	2025-08-14 05:55:26	2025-08-14 05:55:24	2025-08-13 21:00:00	2025-08-14 08:57:42		☑	FAT file system
.Trash-1000		Directory	9	2025-08-13 08:10:48	2025-08-13 08:10:48	2025-08-13 08:10:46			☑	exFAT file system
fanano		Directory	11	2025-07-06 06:39:54	2025-07-06 06:39:54	2025-07-06 06:39:52	2025-08-13 08:45:10		☑	exFAT file system

4.4 Windows Recycle Bin

The Windows Recycle Bin is a temporary storage location for files and folders deleted by users through the graphical user interface (Windows Explorer). When a file is sent to the Recycle Bin, it is not immediately removed from the storage medium. Instead, the file remains physically present on the disk until the Recycle Bin storage space is reclaimed.

Each user account on a Windows system maintains an independent Recycle Bin, ensuring separation of deleted content among multiple profiles.

Recycle Bin structure varies slightly depending on the Windows version:

- Windows Vista and later: C:\$Recycle.Bin\<SID>
- Windows XP and earlier: C:\RECYCLER\<SID>

Forensic Value

The $Recycle.Bin stores two files for each deleted item:

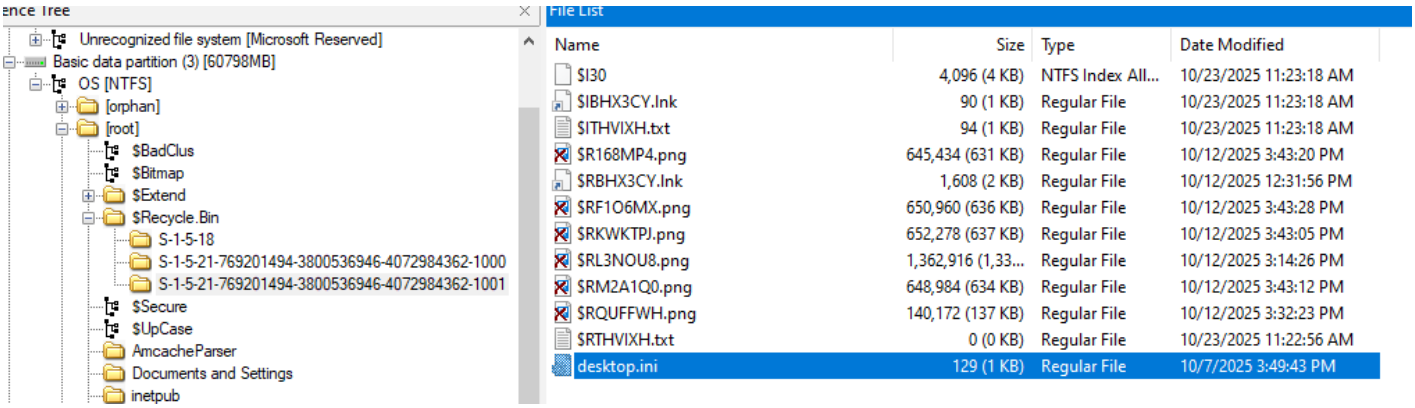
- $I file metadata: Metadata file (contains original path, deletion date/time, and file size)
- $R file: Actual deleted content
- SID path: Links the deleted file to a specific user
- Timestamps: Reveal when the user deleted the file.
- File names and extensions: May indicate attempted concealment (renamed files)

Practical Section

Tools

- FTK Imager https://www.exterro.com/ftk-product-downloads
- RBCmd.exe (Eric Zimmerman) https://ericzimmerman.github.io/?utm_source=chatgpt.com#!index.md
- Rifiuti https://github.com/EricZimmerman/RBCmd.git

From FTK Imager, we mount the disk and start analyzing the $Recycle.Bin. We use FTK Imager to parse old deleted files in some tools. When files are deleted from the recycle bin, you can see them.



We can also use the Rifiuti tool to get more info.

Command: rifiuti-vista.exe C:\$Recycle.Bin\S-1-5-21-769201494-3800536946-4072984362-1001 /a

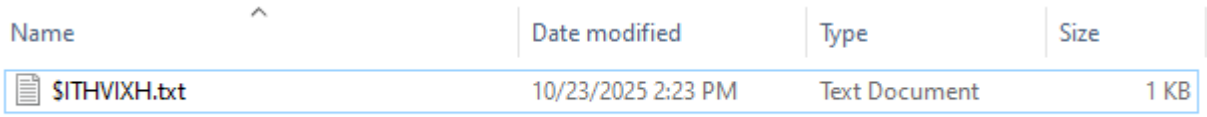
```
F:\Tools\rifiuti2-0.8.2-win64>rifiuti-vista.exe C:\$Recycle.Bin\S-1-5-21-769201494-3800536946-4072984362-1001
Recycle bin path: 'C:\$Recycle.Bin\S-1-5-21-769201494-3800536946-4072984362-1001'
Version: 2
OS Guess: Windows 10 or above
Time zone: UTC [+0000]

Index   Deleted Time    Gone?   Size    Path
$IBHX3CY.lnk    2025-10-23 11:23:18     FALSE   1608    C:\Users\demo\Desktop\info.lnk
$ITHVIXH.txt    2025-10-23 11:23:18     FALSE   0       C:\Users\demo\Desktop\Secret.txt
```

FTK Imager successfully displayed deleted files from the Recycle Bin because it accesses the entire NTFS file system, including unallocated and deleted clusters.

In contrast, Rifiuti2 only parses existing $I metadata files. Therefore, if $I files are deleted or absent, Rifiuti2 will not display any results.

To recover the file from the Recycle Bin, we can make this

Cammand: copy "C:\$Recycle.Bin\S-1-5-21-769201494-3800536946-4072984362-1001\$ITHVIXH.txt" "C:\Users\demo\Desktop\Recoverd"

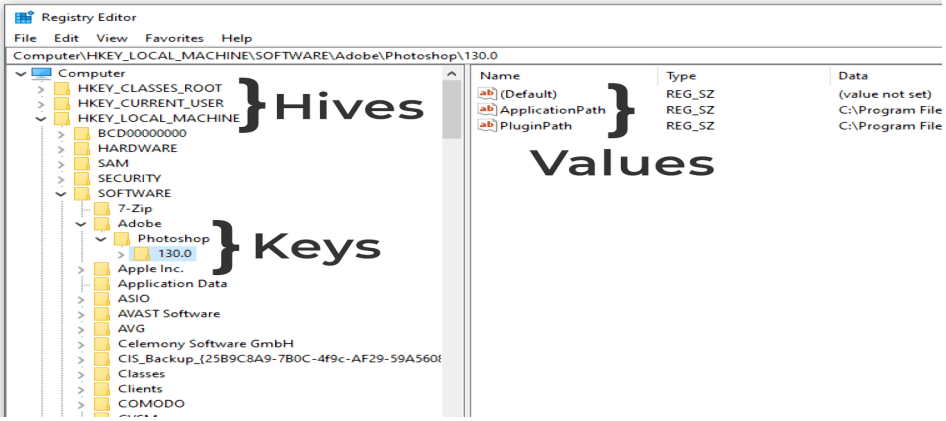


4.5 Windows Registry keys

The registry is a hierarchical database. The Windows Registry stores configuration information for all applications on the system, user-specific settings, configuration of various hardware devices used by the system, and settings for all software on the system, etc.

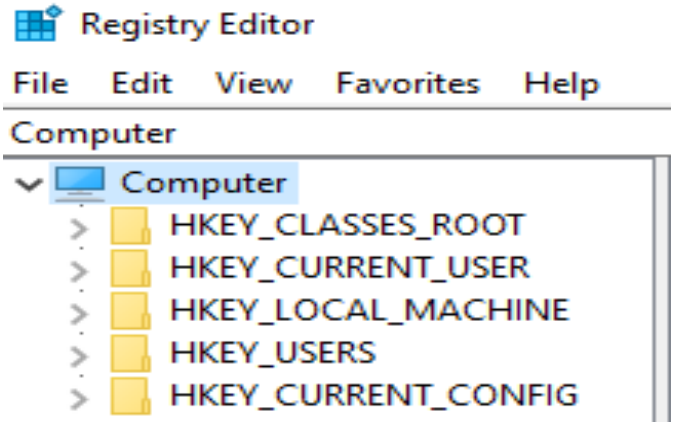
From a forensic perspective, the registry is a goldmine of evidence, providing insight into user activities, connected devices, system configuration, and software execution history.

Registry Structure



Inside the registry, there are root folders. These root folders are referred to as hives. There are five (5) registry hives.

- HKEY_USERS: contains all the loaded user profiles
- HKEY_CURRENT_USER: profile of the currently logged-on user
- HKEY_CLASSES_ROOT: configuration information on the application used to open files
- HKEY_CURRENT_CONFIG: hardware profile of the system at startup
- HKEY_LOCAL_MACHINE: configuration information, including hardware and software settings

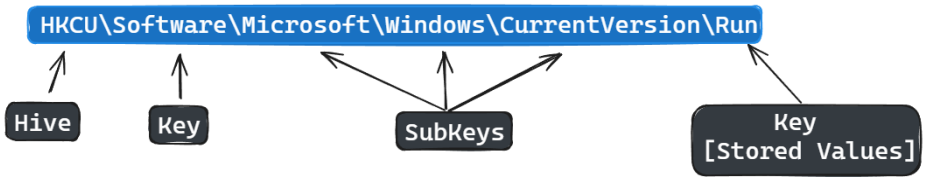


Keys:

- Similar to folders(keys) and subfolders(subkeys),
- produces a folder directory hierarchy

Values:

- Data stored within a key contains data in the form of strings, binary data, integers, and lists.
- where the most valuable forensics data is found.



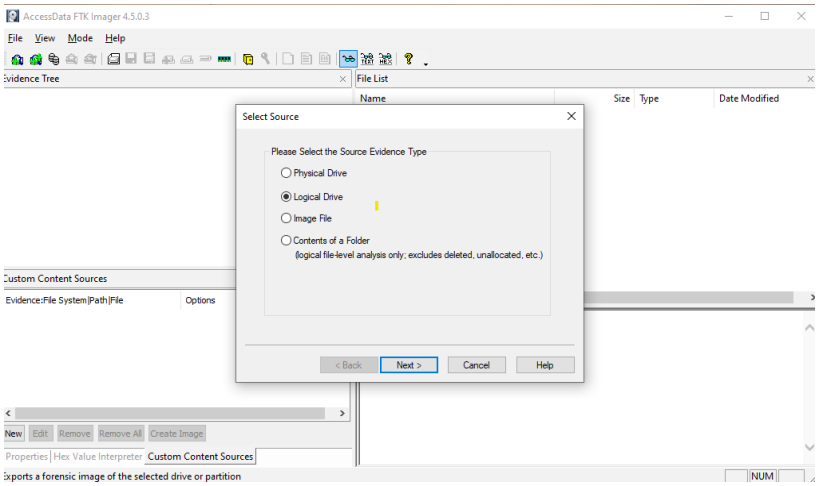
Practical Section

Step 1: Registry file acquisition

Investigating the Windows registry is quite a difficult task because to investigate it properly, the registry needs to be extracted from the computer. Extraction of the registry file is not just a normal copy-and-paste function.

Since registry files store all the configuration information of the computer, they automatically update every second. To extract Windows registry files from the computer, investigators have to use third-party software such as FTK Imager.

Open FTK Imager → click Add Evidence Item → choose Logical Drive → select the drive → Finish.

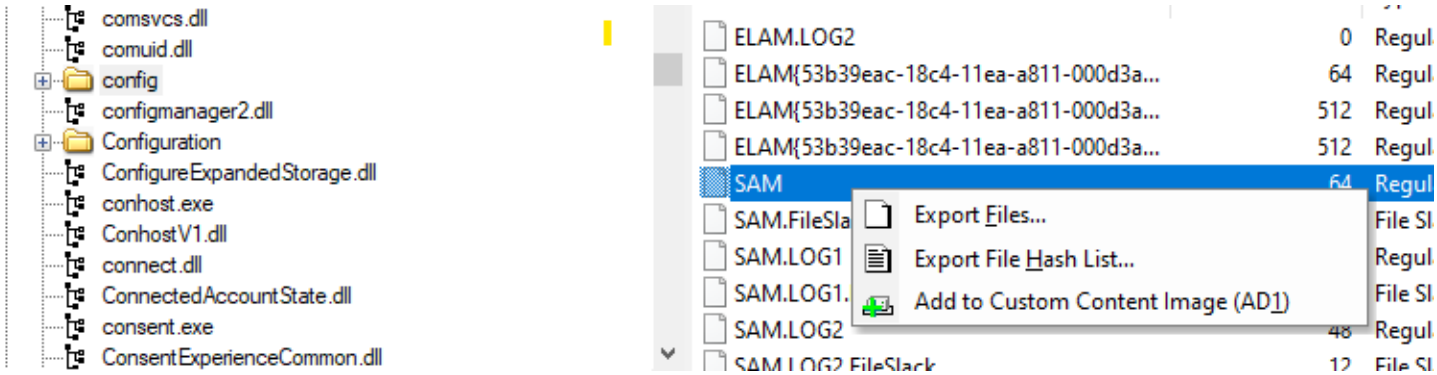


In the Evidence Tree, go to:

- Windows\System32\config → for SYSTEM, SAM, SOFTWARE, SECURITY, DEFAULT
- Users\<username>\NTUSER.DAT
- Users\<username>\AppData\Local\Microsoft\Windows\UsrClass.dat

Right-click any file (like SAM) → Export Files

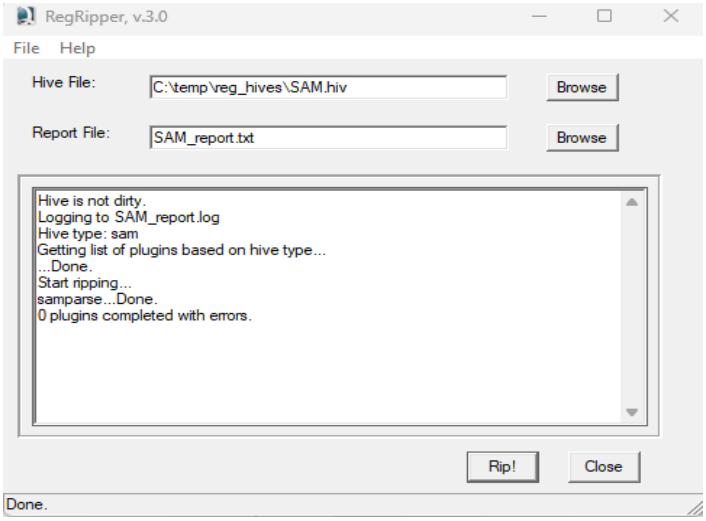
Choose a destination folder and click OK



Step 2: Analyze Registry Files

After exporting the hive, use a forensic analysis tool like RegRipper.

RegRipper reads and extracts useful information automatically from registry hives. It can be used both as a command-line and a GUI



You'll get a **report file** (like SAM_report.txt).
Now:

1. Open the report file using any text editor (like Notepad or Notepad++).
2. Review the extracted data related to the hive
3. Highlight suspicious findings, like strange autoruns or unknown executables.
4. Add notes — what each key tells you and how it supports your investigation.

Important Windows Registry Keys for Forensic Investigation

OS Version:

SOFTWARE\Microsoft\Windows NT\CurrentVersion

Current Control set:

HKLM\SYSTEM\CurrentControlSet

Computer Name:

SYSTEM\CurrentControlSet\Control\ComputerName

Time Zone Information:

SYSTEM\CurrentControlSet\Control\TimeZoneInformation

Network Interfaces and Past Networks:

SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces

Autostart Programs (Autoruns):

NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Run

NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\RunOnce

SAM hive and user information:

SAM\Domains\Account\Users

Windows DHCP Configuration:

SYSTEM\ControlSet00#\Services\Tcpip\Parameters\Interfaces\{GUID}\DhcpIPAddress

Recent Files:

NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs

Office Recent Files:

NTUSER.DAT\Software\Microsoft\Office\VERSION\UserMRU\LiveID_####\FileMRU

Device identification:

SYSTEM\CurrentControlSet\Enum\USBSTOR

SYSTEM\CurrentControlSet\Enum\USB

Installed Applications

SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall

We can use the official Microsoft software to get more information

https://learn.microsoft.com/en-us/troubleshoot/windows-server/performance/windows-registry-advanced-users

4.6 Windows Browser Artifacts

Web browsers are one of the richest sources of forensic evidence on a system. They store traces of user activities such as visited websites, downloads, saved credentials, cookies, and session data. Analyzing browser artifacts helps investigators reconstruct user behavior, identify accessed content, and trace data exfiltration or malicious activity.

Both **Google Chrome** and **Microsoft Edge** (Chromium-based) maintain their data in the user's local profile directory under AppData\Local. These data files, mostly in SQLite and JSON formats, include critical information such as browsing history, cookies, downloads, and login data.

Forensic Value:

- Reconstruct browsing sessions and timelines
- Identify websites visited and files downloaded
- Reveal saved credentials and form data
- Correlate web activity with other system evidence

Microsoft Edge (Chromium-based)

User Data Path: `C:\Users\<USERNAME>\AppData\Local\Microsoft\Edge\User Data\`

Key Files:

- `Default\History` → browsing history (SQLite database)
- `Default\Cookies` → cookies (SQLite database)
- `Default\Login Data` → saved passwords (encrypted)
- `Default\Bookmarks` → user bookmarks (JSON format)
- `Default\Downloads` → list of downloaded files
- `Default\Preferences` → browser settings

Google Chrome

User Data Path: `C:\Users\<USERNAME>\AppData\Local\Google\Chrome\User Data\`

Key Files:

- `Default\History` → browsing history (SQLite database)
- `Default\Cookies` → cookies (SQLite database)
- `Default\Login Data` → saved passwords (encrypted)
- `Default\Bookmarks` → user bookmarks (JSON format)
- `Default\Downloads` → download activity
- `Default\Preferences` → browser configuration

Because they both use the *Chromium engine*, their data structure are nearly identical. These files can be analyzed using forensic tools like *BrowserHistoryView, DB Browser for SQLite, or Belkasoft Evidence Center*.

Practical Section

In this section, we will investigate the Chrome browser and begin by examining data such as caches, history, and login information.

To export these artifacts, we can mount the disk and then open FTK Imager to export any file. We can also use tools like ChromeCacheView and DB Browser for SQLite to parse these files.

Paths we want to check, like browser caches

We will use ChromeCacheView to see the content of the cache file.

C:\Users\<USERNAME>\AppData\Local\Google\Chrome\User Data\Default\Cache

Here, we can investigate the Cached HTML pages, images, CSS, JavaScript, and even some video fragments. Each file may contain URLs, timestamps, and sometimes snippets of page content.

To see the history of user browsing, downloading, and searching URLs

C:\Users\<USERNAME>\AppData\Local\Google\Chrome\User Data\Default\Default\History

In this case, the DB Browser for SQLite can help us to parse the History file

Table: downloads — Filter in any column

	id	guid	current_path	target_path	start_time	received_bytes	total_bytes
1	8	707c9014-cf76-4da7-895f-cda39e2c3f24	C:…	C:…	13363446919449379	7653774	7653774
2	9	75cd7a61-ccfd-4481-bb2b-4f2b5f28f30d	F:\debian-live-12.5.0-amd64-kde.iso	F:\debian-live-12.5.0-amd64-kde.iso	13363513671530894	3499589632	3499589632
3	10	973cb7c3-02a7-4401-b687-f4b523682f4e	C:\Users\VM\Desktop\Abdo.Mo.ovpn	C:\Users\VM\Desktop\Abdo.Mo.ovpn	13363518231225674	8312	8312
4	15	95bba909-3d55-4d0d-aaa5-fec8b641ea99	C:\Users\VM\Desktop\35807.pdf	C:\Users\VM\Desktop\35807.pdf	13363696665589049	2339317	2339317
5	17	7ad5c310-06e9-45d6-9370-64d05997742f	C:\Users\VM\Desktop\Xmind-for-…	C:\Users\VM\Desktop\Xmind-for-…	13364124920464877	165650888	165650888
6	19	8d1ee9b5-be1c-4036-a6e5-dc68b8914a63	C:\Users\VM\Desktop\Google-Hacking-…	C:\Users\VM\Desktop\Google-Hacking-…	13364366616707415	3030345	3030345
7	23	1295c99e-3aec-4d69-9eb8-69f67908df93	C:\Users\VM\Desktop\7z2407-x64.exe	C:\Users\VM\Desktop\7z2407-x64.exe	13364487566592864	1620576	1620576
8	25	85887fb2-5dcd-45bd-aa0f-27369125e1a6	C:\Users\VM\Downloads\ubuntu-24.04-…	C:\Users\VM\Downloads\ubuntu-24.04-…	13364557693915000	6114656256	6114656256
9	26	f4c9f21f-ec45-4a91-bf88-be6b181699ee	C:\Users\VM\Downloads\pexels-…	C:\Users\VM\Downloads\pexels-…	13364568726061343	3081163	3081163
10	27	54bba4ba-c741-4b6d-bb3e-a187a958bffc	C:\Users\VM\Downloads\pexels-…	C:\Users\VM\Downloads\pexels-…	13364568734691865	1000203	1000203

To understand the time, we can use https://www.epochconverter.com/webkit

Here, we can analyze the downloaded files, local path, start/finish time, visited URLs, visit count, last visit time, and this may help us to identify visited malicious sites or data exfiltration activity.

We can also analyze several important browser artifacts, such as cookies, login data, and bookmarks, to gather valuable information about user activity, authentication details, and frequently accessed websites

4.7 USB Forensic

USB (Universal Serial Bus) devices are widely used to connect external hardware components to computers. These devices include not only USB flash drives, but also external hard disks, smartphones, digital cameras, keyboards, printers, and other peripherals.

From a forensic perspective, analyzing USB activity is critical because USB devices can be used to transfer data, exfiltrate information, or introduce malicious files. Even after a device is removed, Windows maintains several registry artifacts that record details about connected USB devices, such as device names, serial numbers, connection times, and assigned drive letters.

Forensic Value:

- Reveal what devices were connected, when, and by which user.
- Help reconstruct device usage history and establish links between users and devices.
- Provide evidence of possible data exfiltration or introduction of malicious files.

Practical Section

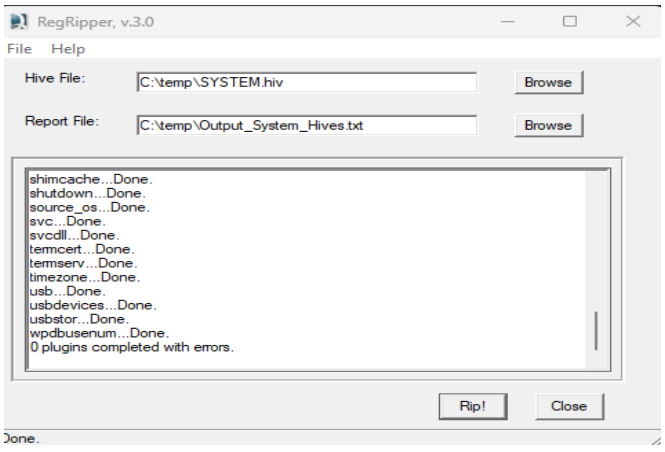
Paths

- Registry Location (vendor, product, revision number) `HKLMSYSTEM\CurrentControlSet\Enum\USBSTOR`
- Mounted Devices `HKLM\SYSTEM\MountedDevices`
- Proving of usage: `Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2`

Tools

- RipRipper https://github.com/keydet89/RegRipper3.0
- USBForensicTracker https://e5hforensics.com/index.php/downloads/software/usb-forensic-tracker/
- USBDeview https://usbdeview.com/

We will use RipRipper to dump the SYSTEM hive.



In the output file, if we search for Enum\USBSTOR, we will see all USBs that have connected to the device.

```
Disk&Ven_Kingston&Prod_DataTraveler_2.0&Rev_PMAP [2024-11-22 09:11:20]
  S/N: 001A4D5F10FABF419951F4278&0 [2025-08-14 08:53:24Z]
  Device Parameters LastWrite: [2024-11-22 09:11:20Z]
  Properties LastWrite       : [2024-11-22 09:11:21Z]
    FriendlyName          : Kingston DataTraveler 2.0 USB Device
    First InstallDate     : 2024-11-22 09:11:20Z
    InstallDate           : 2024-11-22 09:11:20Z
    Last Arrival          : 2025-08-21 14:59:29Z
    Last Removal          : 2025-08-21 15:30:11Z
```

DISK&Ven_{Name}&Prod_{Name}&Rev_{Value}

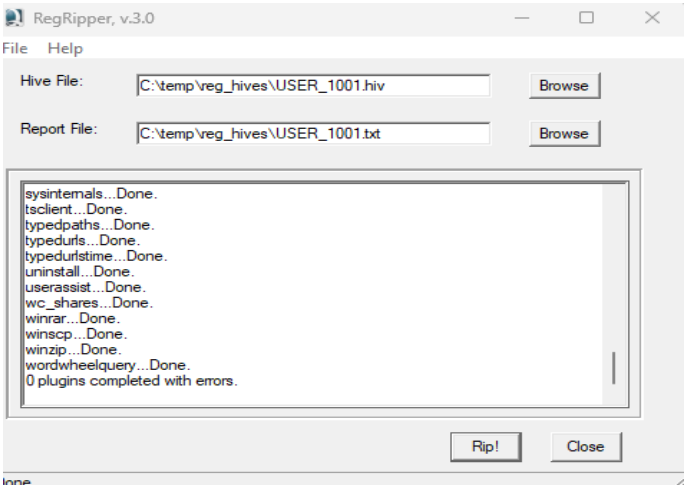
So, for the entry, we have:

- Vendor: Kingstone
- Product: DataTraveler
- Revision number: PMAP
- Serial No: 001A4D5F10FABF419951F427&0
- Friendly Name: Kingston DataTraveler 2.0 USB Device
- First InstallDate: 2024-11-22 09:11:20
- InstallDate: 2024-11-22 09:11:20
- Last Arrival: 2025-08-21 14:59:29
- Last Removal: 2025-08-21 15:30:1

If we search in the same file for MountedDevices, we will see Volume GUIDs and mounted letters.

```
Device: _??_USBSTOR#Disk&Ven_Kingston&Prod_DT_101_G2&Rev_PMAP#001372982A06BB61957200059&0#{53f56307-b6bf-11d0-94f2-00a0c91efb8b}
  \??\Volume{1454efd2-2d6b-11ef-8b00-4ccc6a6f97f5}
  \DosDevices\F:
```

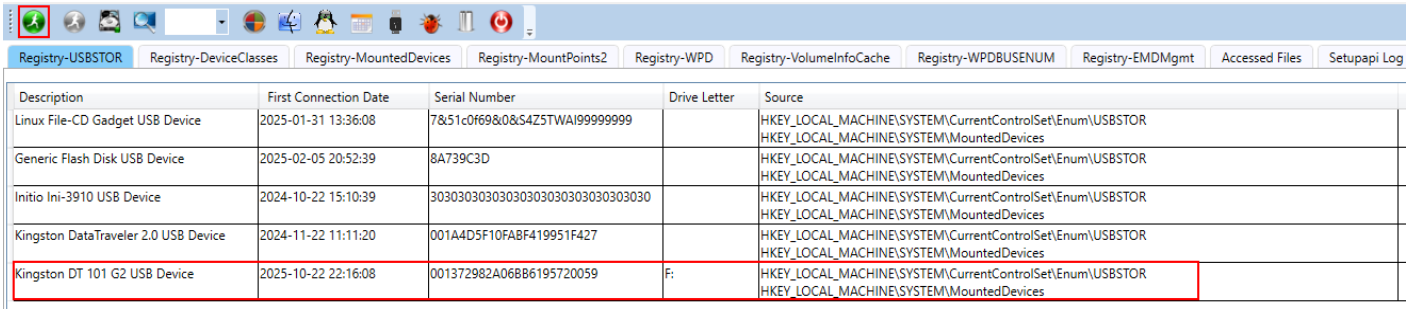
That we have found what USB devices were connected to this computer, we now need to check which user was using them. We will go th[i]rough the single NTUSER.DAT file. Again, start RegRipper, but this time make sure to load the “NTUSER.DAT” registry file.



After searching for MountPoints2, we are now sure this user was responsible for this USB device.

```
Remote Drives:

Volumes:
2025-10-22 21:09:20Z
  {1454efd2-2d6b-11ef-8b00-4ccc6a6f97f5}
```

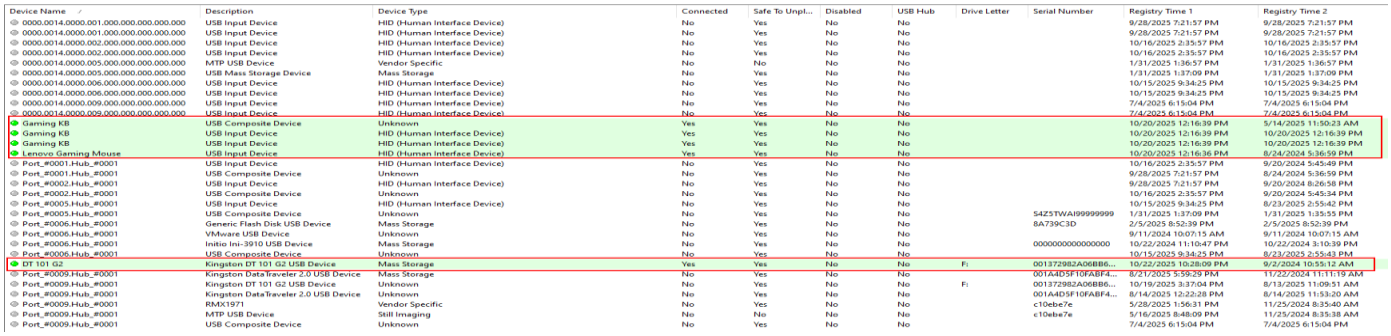
We can also use some tools for USB trackers, like USBForensicTracker and USBDeview. After the installation of USBForensicTracker, we execute USBFT64.exe.



Registry-USBSTOR | Registry-DeviceClasses | Registry-MountedDevices | Registry-MountPoints2 | Registry-WPD | Registry-VolumeInfoCache | Registry-WPDBUSENUM | Registry-EMDMgmt | Accessed Files | Setupapi Log

Description	First Connection Date	Serial Number	Drive Letter	Source
Linux File-CD Gadget USB Device	2025-01-31 13:36:08	7&51c0f69&0&S4Z5TWAI99999999		HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\USBSTOR HKEY_LOCAL_MACHINE\SYSTEM\MountedDevices
Generic Flash Disk USB Device	2025-02-05 20:52:39	8A739C3D		HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\USBSTOR HKEY_LOCAL_MACHINE\SYSTEM\MountedDevices
Initio Ini-3910 USB Device	2024-10-22 15:10:39	3030303030303030303030303030		HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\USBSTOR HKEY_LOCAL_MACHINE\SYSTEM\MountedDevices
Kingston DataTraveler 2.0 USB Device	2024-11-22 11:11:20	001A4D5F10FABF419951F427		HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\USBSTOR HKEY_LOCAL_MACHINE\SYSTEM\MountedDevices
Kingston DT 101 G2 USB Device	2025-10-22 22:16:08	001372982A06BB6195720059	F:	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\USBSTOR HKEY_LOCAL_MACHINE\SYSTEM\MountedDevices

We can also use USBDeview. After the installation, we execute USBDeview.exe.



Device Name	Description	Device Type	Connected	Safe To Unpl...	Disabled	USB Hub	Drive Letter	Serial Number	Registry Time 1	Registry Time 2
0000.0014.0000.001.000.000.000.000.000	USB Input Device	HID (Human Interface Device)	No	Yes	No	No			9/28/2025 7:21:57 PM	9/28/2025 7:21:57 PM
0000.0014.0000.001.000.000.000.000.000	USB Input Device	HID (Human Interface Device)	No	Yes	No	No			9/28/2025 7:21:57 PM	9/28/2025 7:21:57 PM
0000.0014.0000.002.000.000.000.000.000	USB Input Device	HID (Human Interface Device)	No	Yes	No	No			10/16/2025 2:35:57 PM	10/16/2025 2:35:57 PM
0000.0014.0000.002.000.000.000.000.000	USB Input Device	HID (Human Interface Device)	No	Yes	No	No			10/16/2025 2:35:57 PM	10/16/2025 2:35:57 PM
0000.0014.0000.005.000.000.000.000.000	MTP USB Device	Vendor Specific	No	No	No	No			1/31/2025 1:36:57 PM	1/31/2025 1:36:57 PM
0000.0014.0000.005.000.000.000.000.000	USB Mass Storage Device	Mass Storage	No	Yes	No	No			1/31/2025 1:37:09 PM	1/31/2025 1:37:09 PM
0000.0014.0000.006.000.000.000.000.000	USB Input Device	HID (Human Interface Device)	No	Yes	No	No			10/15/2025 9:34:25 PM	10/15/2025 9:34:25 PM
0000.0014.0000.006.000.000.000.000.000	USB Input Device	HID (Human Interface Device)	No	Yes	No	No			10/15/2025 9:34:25 PM	10/15/2025 9:34:25 PM
0000.0014.0000.009.000.000.000.000.000	USB Input Device	HID (Human Interface Device)	No	Yes	No	No			7/4/2025 6:15:04 PM	7/4/2025 6:15:04 PM
0000.0014.0000.009.000.000.000.000.000	USB Input Device	HID (Human Interface Device)	No	Yes	No	No			7/4/2025 6:15:04 PM	7/4/2025 6:15:04 PM
Gaming KB	USB Composite Device	Unknown	Yes	Yes	No	No			10/20/2025 12:16:39 PM	5/14/2025 11:50:23 AM
Gaming KB	USB Input Device	HID (Human Interface Device)	Yes	Yes	No	No			10/20/2025 12:16:39 PM	10/20/2025 12:16:39 PM
Gaming KB	USB Input Device	HID (Human Interface Device)	Yes	Yes	No	No			10/20/2025 12:16:39 PM	10/20/2025 12:16:39 PM
Lenovo Gaming Mouse	USB Input Device	HID (Human Interface Device)	Yes	Yes	No	No			10/20/2025 12:16:36 PM	8/24/2024 5:36:59 PM
Port_#0001.Hub_#0001	USB Input Device	HID (Human Interface Device)	No	Yes	No	No			10/16/2025 2:35:57 PM	9/20/2024 5:45:48 PM
Port_#0001.Hub_#0001	USB Composite Device	Unknown	No	Yes	No	No			9/28/2025 7:21:57 PM	8/24/2024 5:36:59 PM
Port_#0002.Hub_#0001	USB Input Device	HID (Human Interface Device)	No	Yes	No	No			9/28/2025 7:21:57 PM	9/20/2024 8:26:58 PM
Port_#0002.Hub_#0001	USB Composite Device	Unknown	No	Yes	No	No			10/16/2025 2:35:57 PM	9/20/2024 5:45:34 PM
Port_#0005.Hub_#0001	USB Input Device	HID (Human Interface Device)	No	Yes	No	No			10/15/2025 9:34:25 PM	8/23/2025 2:55:42 PM
Port_#0005.Hub_#0001	USB Composite Device	Unknown	No	Yes	No	No		S4Z5TWAI99999999	1/31/2025 1:37:09 PM	1/31/2025 1:35:55 PM
Port_#0006.Hub_#0001	Generic Flash Disk USB Device	Mass Storage	No	Yes	No	No		8A739C3D	2/5/2025 8:52:39 PM	2/5/2025 8:52:39 PM
Port_#0006.Hub_#0001	VMware USB Device	Unknown	No	Yes	No	No			9/11/2024 10:07:15 AM	9/11/2024 10:07:15 AM
Port_#0006.Hub_#0001	Initio Ini-3910 USB Device	Mass Storage	No	Yes	No	No		0000000000000000	10/22/2024 11:10:47 PM	10/22/2024 3:10:39 PM
Port_#0006.Hub_#0001	USB Composite Device	Unknown	No	Yes	No	No			10/15/2025 9:34:25 PM	8/23/2025 2:55:43 PM
DT 101 G2	Kingston DT 101 G2 USB Device	Mass Storage	Yes	Yes	No	No	F:	001372982A06BB6...	10/22/2025 10:28:09 PM	9/2/2024 10:55:12 AM
Port_#0009.Hub_#0001	Kingston DataTraveler 2.0 USB Device	Mass Storage	No	Yes	No	No		001A4D5F10FABF4...	8/21/2025 5:59:29 PM	11/22/2024 11:11:19 AM
Port_#0009.Hub_#0001	Kingston DT 101 G2 USB Device	Unknown	No	Yes	No	No	F:	001372982A06BB6...	10/19/2025 3:37:04 PM	8/13/2025 11:09:51 AM
Port_#0009.Hub_#0001	Kingston DataTraveler 2.0 USB Device	Unknown	No	Yes	No	No		001A4D5F10FABF4...	8/14/2025 12:22:28 PM	8/14/2025 11:53:20 AM
Port_#0009.Hub_#0001	RMX1971	Vendor Specific	No	Yes	No	No		c10ebe7e	5/28/2025 1:56:31 PM	11/25/2024 8:35:40 AM
Port_#0009.Hub_#0001	MTP USB Device	Still Imaging	No	No	No	No		c10ebe7e	5/16/2025 8:48:09 PM	11/25/2024 8:35:38 AM
Port_#0009.Hub_#0001	USB Composite Device	Unknown	No	Yes	No	No			7/4/2025 6:15:04 PM	7/4/2025 6:15:04 PM

Since USB not only means a USB device, but also any external device attached to the computer using a USB interface (Printer, camera and etc.)

Part 2: Linux Forensics

1. Introduction

Linux forensics focuses on identifying, collecting, preserving, and analyzing digital evidence from Linux-based systems to investigate security incidents, intrusions, or unauthorized activities. It involves examining key system artifacts such as log files, configuration files, shell history, system binaries, and memory images to reconstruct user actions and determine the root cause of an incident.

Just like in Windows, data in Linux systems can be categorized into **volatile** and **non-volatile** data. Understanding this classification helps forensic investigators decide the correct order of evidence acquisition and maintain the integrity of collected data.

Volatile Data

Volatile data exists only while the system is running and is lost once it is powered off or restarted. This includes information such as running processes, open network connections, logged-in users, active system services, and memory contents. Therefore, volatile data should always be collected first using live forensics tools such as **LiME (Linux Memory Extractor)**, **dd**, or **Volatility** for memory capture. Network connections and processes can also be captured.

Non-Volatile Data

Non-volatile data remains on the disk even after shutdown. It includes log files, configuration files, user home directories, shell histories, and binary executables. This data can be collected later from disk images using forensic tools like **FTK Imager**, **The Sleuth Kit (TSK)**, **Autopsy**, or **dd** for disk cloning.

Key Goals of Linux Forensics

- **Detecting malicious activity** such as privilege escalation, persistence mechanisms, or rootkit installation.
- **Recovering critical evidence,** including deleted files, memory dumps, logs, and command histories.
- **Preserving integrity** by performing acquisitions in a forensically sound manner (using write blockers and hashing evidence).
- **Reconstructing events** to determine the timeline of actions and identify compromised accounts or processes.

2. Collecting Volatile Information

Memory forensics on Linux systems begins with acquiring and examining volatile data, such as active processes, open network connections, loaded kernel modules, and other runtime artifacts stored in RAM. Tools such as LiME or AVML are commonly used for reliable memory acquisition and should be run from trusted external media to minimize contamination of the target system. Once captured, the memory dump is analyzed using frameworks such as Volatility, allowing investigators to extract and interpret evidence from the running system with accuracy and forensic integrity.

2.1 Memory Acquisition Using LiMe

A Loadable Kernel Module (LKM) that allows for volatile memory acquisition from Linux and Linux-based devices, such as Android. This makes LiME unique as it is the first tool that allows for full memory captures on Android devices. It also minimizes its interaction between user and kernel space processes during acquisition, which allows it to produce memory captures that are more forensically sound than those of other tools designed for Linux memory acquisition.

On the Target, we check the release of the OS.

```
abdo@ubuntu:~/Desktop$ cat /etc/os-release
PRETTY_NAME="Ubuntu 24.04 LTS"
NAME="Ubuntu"
VERSION_ID="24.04"
VERSION="24.04 LTS (Noble Numbat)"
VERSION_CODENAME=noble
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies
UBUNTU_CODENAME=noble
LOGO=ubuntu-logo
abdo@ubuntu:~/Desktop$ uname -r
6.8.0-48-generic
```

After identifying the required kernel module, we need to build a kernel object using the same kernel version as the target system. The build directory can be obtained from the target machine at /lib/modules/6.8.0-48-generic/build. Alternatively, we can use another machine that has the same kernel version to compile the kernel object.

In the practical demo, we will use another Linux machine that has the same kernel version.

In our Inverisgraoin machine, we should install some essential gcc and g++ compilers with the same kernel version.

```
sudo apt install -y build-essential gcc-13 g++-13 linux-headers-$(uname -r)
```

Then we download the LiMe tool from this link https://github.com/504ensicsLabs/LiME

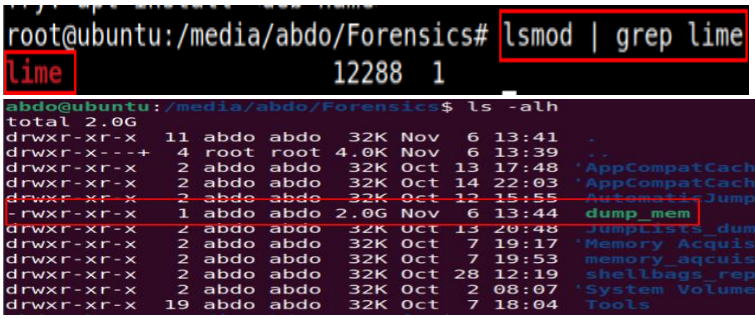
After installing, go to /src and run make

```
root@ubuntu:/home/Memo2/LiME/src# make
make -C /lib/modules/6.8.0-48-generic/build M="/home/Memo2/LiME/src" modules
make[1]: Entering directory '/usr/src/linux-headers-6.8.0-48-generic'
warning: the compiler differs from the one used to build the kernel
  The kernel was built by: x86_64-linux-gnu-gcc-13 (Ubuntu 13.2.0-23ubuntu4) 13.2.0
  You are using:           gcc-13 (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0
  CC [M]  /home/Memo2/LiME/src/tcp.o
  CC [M]  /home/Memo2/LiME/src/disk.o
```

```
root@ubuntu:/home/Memo2/LiME/src# ls -l lime-6.8.0-48-generic.ko
-rw-r--r-- 1 root root 29136 Nov  6 13:08 lime-6.8.0-48-generic.ko
```

The compiled kernel object (.ko) must be copied to the target and inserted into the kernel so LiME can write the memory image directly to the external USB.

sudo insmod lime-6.8.0-48-generic.ko path=/media/Forensics_USB/dump_mem format=lime we can make sure that the process is running. lsmod command to remove run sudo rmmod lime after finishing.



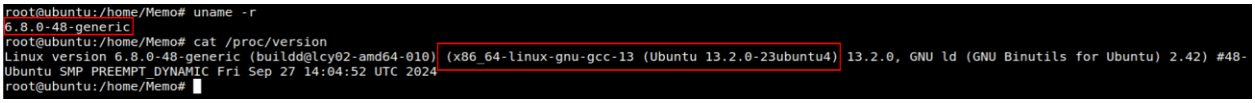
Here, we have RAM acquisition; we can investigate using volatility or other analysis tools.

2.2 Memory Acquisition Using AVML

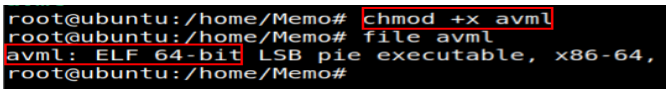
A portable volatile memory acquisition tool for Linux. AVML is an X86_64 userland volatile memory acquisition tool written in Rust, intended to be deployed as a static binary. AVML can be used to acquire memory without knowing the target OS distribution or kernel a priori. No on-target compilation or fingerprinting is needed.

We can download the tool from the repo: [https://github.com/microsoft/avml.](https://github.com/microsoft/avml)

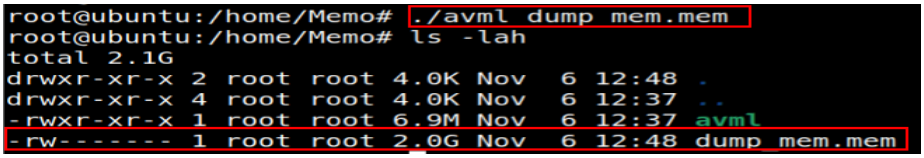
We check the kernel versions with uname -r



After we install it, we need to make an executable permission with chmod +x avml.



The tool is ready to run and write the memory image directly to an external device (USB) to preserve the evidence.



Here, we have RAM acquisition; we can investigate using volatility or other analysis tools.

So what is the difference between the two tools, LiMe and AVML?

LiME is a kernel-level memory acquisition tool that offers high forensic accuracy but requires a compiled module for the exact kernel version and may need Secure Boot disabled.
AVML, on the other hand, is a user-space tool by Microsoft that works across many Linux systems without compilation. It's easier and safer to use in live or cloud environments, but it provides slightly less forensic control than LiME.

2.3 Memory Analysis using Volatility

2.4 Collecting Volatile Artifacts

We can also collect volatile information from a Linux system, which involves acquiring data that resides in the system's RAM and other temporary locations that lose their contents when the system is powered off. Here, we will run some commands to get these artifacts.

Types of Volatile Information we can collect by running these commands

- System Identification
- Network and Connection Data
- System Activity
- Kernel and Filesystem Information
- Storage Details
- Opened files

Collecting Hostname, Date, and Time

```
sansforensics@siftworkstation: /home
$ date
Fri Nov  7 09:45:04 PM MSK 2025
sansforensics@siftworkstation: /home
$ cat /etc/timezone
Europe/Kirov
```

Collecting Uptime Data

```
sansforensics@siftworkstation: /home
$ uptime
 21:46:14 up 48 min,  1 user,  load average: 0.01, 0.06, 0.17
```

This command displays how long the system has been running since the last restart.

Collecting Network Information

```
sansforensics@siftworkstation: /home
$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         192.168.200.2   0.0.0.0         UG        0 0          0 ens33
172.17.0.0      0.0.0.0         255.255.0.0     U         0 0          0 docker0
192.168.200.0   0.0.0.0         255.255.255.0   U         0 0          0 ens33
192.168.200.2   0.0.0.0         255.255.255.255 UH        0 0          0 ens33
```

```
sansforensics@siftworkstation: /home
$ ip r
default via 192.168.200.2 dev ens33 proto dhcp src 192.168.200.130 metric 100
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1 linkdown
192.168.200.0/24 dev ens33 proto kernel scope link src 192.168.200.130 metric 100
192.168.200.2 dev ens33 proto dhcp scope link src 192.168.200.130 metric 100
```

Finding Programs/Processes Associated with a Port

```
sansforensics@siftworkstation: /home
$ netstat -tulnp
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:25              0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:139             0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:445             0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      -
tcp6       0      0 :::80                   :::*                    LISTEN      -
tcp6       0      0 :::25                   :::*                    LISTEN      -
tcp6       0      0 :::22                   :::*                    LISTEN      -
tcp6       0      0 :::139                  :::*                    LISTEN      -
tcp6       0      0 :::445                  :::*                    LISTEN      -
tcp6       0      0 :::1716                 :::*                    LISTEN      2685/kdeconnectd
tcp6       0      0 ::1:631                 :::*                    LISTEN      -
udp        0      0 0.0.0.0:631             0.0.0.0:*                           -
udp        0      0 0.0.0.0:5353            0.0.0.0:*                           -
udp        0      0 0.0.0.0:50501           0.0.0.0:*                           -
```

We can add netstat -tulnp | grep LISTEN to get the only ports that LISTEN.

lsof | more command retrieves information about all active processes and open files.

```
systemd      1                    root   61u     unix 0xffff95d742ceea80      0t0      21233 /run/systemd/io.system.ManagedOOM type=S
systemd      1                    root   62u     unix 0xffff95d74239bb80      0t0      21942 /run/systemd/journal/stdout type=STREAM
systemd      1                    root   63u     unix 0xffff95d742ceddc0      0t0      21943 /run/systemd/journal/stdout type=STREAM
systemd      1                    root   64u     unix 0xffff95d742ceeec0      0t0      22176 /run/systemd/journal/stdout type=STREAM
systemd      1                    root   65u     unix 0xffff95d743dd8cc0      0t0      23749 /run/systemd/journal/stdout type=STREAM
systemd      1                    root   66u     unix 0xffff95d743dd9540      0t0      21466 /run/systemd/journal/stdout type=STREAM
```

```
sansforensics@siftworkstation: /home
$ mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
udev on /dev type devtmpfs (rw,nosuid,relatime,size=1946352k,nr_inodes=486588,mode=755,inode64)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,nodev,noexec,relatime,size=401816k,mode=755,inode64)
/dev/mapper/ubuntu--vg-ubuntu--lv on / type ext4 (rw,relatime)
```

With the mount, we can see all information about the attached devices.

Finding Loaded Kernel Modules

This command reads the contents of/proc/modules and lists all modules loaded in the kernel.

```
root@siftworkstation:/home# lsmod
Module                  Size  Used by
tls                   114688  0
xt_conntrack           16384  1
nft_chain_nat          16384  3
xt_MASQUERADE          20480  1
nf_nat                 49152  2 nft_chain_nat,xt_MASQUERADE
nf_conntrack_netlink    49152  0
nf_conntrack          172032  4 xt_conntrack,nf_nat,nf_conntrack_netlink,xt_MASQUERADE
nf_defrag_ipv6         24576  1 nf_conntrack
nf_defrag_ipv4         16384  1 nf_conntrack
xfrm_user              40960  1
xfrm_algo              16384  1 xfrm_user
nft_counter            16384  15
xt_addrtype            16384  2
nft_compat             20480  4
nf_tables             249856  43 nft_compat,nft_counter,nft_chain_nat
nfnetlink              20480  4 nft_compat,nf_conntrack_netlink,nf_tables
br_netfilter           32768  0
```

Viewing Running Processes in the System: `ps aux | less`. This command lists the process and its child processes.

```
root        5195  0.0  0.1  11804  5904 pts/1    S+   21:58   0:00 sudo su root
root        5196  0.0  0.0  11804   956 pts/0    Ss   21:58   0:00 sudo su root
root        5197  0.0  0.1  10500  4584 pts/0    S    21:58   0:00 su root
root        5198  0.0  0.1   7636  4420 pts/0    S    21:58   0:00 bash
root        5223  0.0  0.0   3212  1004 ?        S    21:59   0:00 sleep 3600
root        5235  0.0  0.0      0     0 ?        I    22:03   0:00 [kworker/u4:2]
root        5240  0.0  0.0  10072  3532 pts/0    R+   22:04   0:00 ps aux
```

Viewing Linux Services Using systemctl

Systemctl allows investigators to list, filter, and analyze active and inactive services, helping them identify processes that may be relevant to a security incident or compromise.

List all system services: `systemctl list-units --type=service --all`

```
apport.service                                    loaded active exited  LSB: automatic crash report generation
avahi-daemon.service                              loaded active running Avahi mDNS/DNS-SD Stack
binfmt-support.service                            loaded active exited  Enable support for additional executable binary
blk-availability.service                          loaded active exited  Availability of block devices
clamav-freshclam.service                          loaded active running ClamAV virus database updater
cloud-config.service                              loaded active exited  Apply the settings specified in cloud-config
cloud-final.service                               loaded active exited  Execute cloud user/final scripts
cloud-init-local.service                          loaded active exited  Initial cloud-init job (pre-networking)
cloud-init.service                                loaded active exited  Initial cloud-init job (metadata service crawler
colord.service                                    loaded active running Manage, Install and Generate Color Profiles
console-setup.service                             loaded active exited  Set console font and keymap
containerd.service                                loaded active running containerd container runtime
cron.service                                      loaded active running Regular background program processing daemon
```

We can also display services by their state `systemctl list-units --type=service --state=<state>`

Collecting Swap Areas and Disk Partition Information

cat /proc/partitions

```
   8       16     262144 sdb
   8       17     261120 sdb1
   8        0  512000000 sda
   8        1       1024 sda1
   8        2    2097152 sda2
   8        3  509899776 sda3
   8       48     268288 sdd
   8       49     267264 sdd1
   8       32     262144 sdc
   8       33     130048 sdc1
   8       34     131072 sdc2
```

We can also check the swap size and area in a Linux system.

```
root@siftworkstation:/home# cat /proc/swaps
Filename                                Type            Size
/swap.img                               file            4019196
```

Collecting Kernel Messages

dmesg | less can help us see the kernel message information

```
[    0.679335] ACPI: bus type PCI registered
[    0.679420] acpiphp: ACPI Hot Plug PCI Controller Driver version: 0.5
[    0.680103] PCI: MMCONFIG for domain 0000 [bus 00-7f] at [mem 0xf0000000-0xf7ffffff]
[    0.680279] PCI: MMCONFIG at [mem 0xf0000000-0xf7ffffff] reserved in E820
[    0.680417] PCI: Using configuration type 1 for base access
[    0.684824] Kprobes globally optimized
[    0.686314] HugeTLB registered 1.00 GiB page size, pre-allocated 0 pages
[    0.686314] HugeTLB registered 2.00 MiB page size, pre-allocated 0 pages
[    0.702143] ACPI: Added _OSI(Module Device)
[    0.702143] ACPI: Added _OSI(Processor Device)
[    0.702209] ACPI: Added _OSI(3.0 _SCP Extensions)
[    0.702298] ACPI: Added _OSI(Processor Aggregator Device)
[    0.702397] ACPI: Added _OSI(Linux-Dell-Video)
[    0.702482] ACPI: Added _OSI(Linux-Lenovo-NV-HDMI-Audio)
[    0.702580] ACPI: Added _OSI(Linux-HPI-Hybrid-Graphics)
[    0.725740] ACPI: 1 ACPI AML tables successfully acquired and loaded
[    0.726537] ACPI: [Firmware Bug]: BIOS _OSI(Linux) query ignored
```

3. Collecting Non-volatile Information

In Linux forensics, non-volatile information refers to data stored on persistent media that remains intact after a system shutdown or reboot.
This includes files and configurations saved on hard drives, SSDs, or other storage devices.

Such information is critical in digital investigations because it provides long-term evidence of user activity, system configuration, installed software, and potential malicious modifications.

Typical examples of non-volatile data in Linux include:

- System Information
- Kernel information
- User accounts
- Currently logged-in users and login history
- System logs
- Linux log files
- User history file
- Hidden files and directories
- Suspicious information
- File signatures

Forensic Significance

1. **Persistence of Evidence**
   Non-volatile data survives reboots, meaning investigators can analyze it long after the event occurred.
2. **Timeline Reconstruction**
   File metadata (creation, modification, and access times) helps build a timeline of user and system activity, showing what happened and when.
3. **Evidence of Actions**
   User accounts, login history, and shell commands can link specific actions to specific users.
4. **Detection of Persistence Mechanisms**
   Attackers often create hidden startup scripts or cron jobs to maintain access.
5. **Recovery of Deleted or Hidden Data**
   Using forensic tools like The Sleuth Kit (TSK), analysts can recover deleted files, fragments, or metadata remnants stored in the file system.

3.1 Disk Acquisition Using dd

The dd (data duplicator) command is a powerful Linux utility used to clone or image entire disks, partitions, or individual files. It copies raw data directly from the source to the destination, ensuring that every byte is preserved exactly as it exists on the original drive.

dd if=<input_file> of=<output_file> conv=noerror,sync

- if= Input file or device (/dev/sda)
- of= Output file or image (/mnt/usb/disk.img)
- conv=noerror, sync Continue copying even if errors occur and pad missing blocks

Before performing disk acquisition, it is essential to identify all connected storage devices and understand their partition layout, sizes, and file systems. This ensures that investigators image the correct device and avoid accidental overwriting of evidence.

In Linux, all physical disks and their partitions are represented as device files located under the /dev directory

- df (Disk Free) Shows mounted file systems and available space.

```
sansforensics@siftworkstation: ~
$ df -h
Filesystem                         Size  Used Avail Use% Mounted on
tmpfs                              393M  3.3M  390M   1% /run
/dev/mapper/ubuntu--vg-ubuntu--lv   98G   78G   16G  84% /
tmpfs                              2.0G     0  2.0G   0% /dev/shm
tmpfs                              5.0M     0  5.0M   0% /run/lock
tmpfs                              2.0G     0  2.0G   0% /run/qemu
/dev/sda2                          2.0G  139M  1.7G   8% /boot
tmpfs                              393M  152K  393M   1% /run/user/1000
/dev/sde1                           15G  7.8G  7.2G  53% /media/sansforensics/Forensics
```

- fdisk (Partition Table Utility) Lists partition tables and their types (MBR or GPT).

```
Disk /dev/sdb: 256 MiB, 268435456 bytes, 524288 sectors
Disk model: VMware Virtual S
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x843a2409

Device     Boot Start    End Sectors  Size Id Type
/dev/sdb1        2048 524287  522240  255M  7 HPFS/NTFS/exFAT
```

- lsblk (List Block Devices)

```
sda                         8:0    0 488.3G  0 disk
├─sda1                      8:1    0     1M  0 part
├─sda2                      8:2    0     2G  0 part /boot
└─sda3                      8:3    0 486.3G  0 part
  └─ubuntu--vg-ubuntu--lv 253:0    0   100G  0 lvm  /
```

Practical Section

$ sudo dd if=/dev/sdb1 of=~/Desktop/acquisition_disk/Image.dd

```
sansforensics@siftworkstation: ~/Desktop/acquisition_disk
$ sudo dd if=/dev/sdb1 of=~/Desktop/acquisition_disk/Image.dd
522240+0 records in
522240+0 records out
267386880 bytes (267 MB, 255 MiB) copied, 5.81859 s, 46.0 MB/s
sansforensics@siftworkstation: ~/Desktop/acquisition_disk
$ ls -lah
total 256M
drwxrwxr-x 2 sansforensics sansforensics 4.0K Nov  1 22:05 .
drwxr-xr-x 6 sansforensics sansforensics  20K Nov  1 22:03 ..
-rw-r--r-- 1 root          root          255M Nov  1 22:05 Image.dd
```

If any sector of dcorruptediks is corrupted and an error has occurred, we can use conv=noerror, sync

$ sudo dd if=/dev/sdb1 of=~/Desktop/acquisition_disk/Image.dd conv=noerror, sync

In cases where the disk is very large, such as 1 TB or more, acquiring the entire disk as a single image can be challenging. To make the acquisition process easier and faster, the disk image can be split into smaller segments, allowing for more manageable and efficient imaging.

$ sudo dd if=/dev/sdb1 | split -b 100m

```
sansforensics@siftworkstation: ~/Desktop/acquisition_disk/splited_acquisition_disks
$ sudo dd if=/dev/sdb1 | split -b 100m
522240+0 records in
522240+0 records out
267386880 bytes (267 MB, 255 MiB) copied, 1.84684 s, 145 MB/s
sansforensics@siftworkstation: ~/Desktop/acquisition_disk/splited_acquisition_disks
$ ls -lah
total 256M
drwxrwxr-x 2 sansforensics sansforensics 4.0K Nov  1 22:12 .
drwxrwxr-x 3 sansforensics sansforensics 4.0K Nov  1 22:12 ..
-rw-rw-r-- 1 sansforensics sansforensics 100M Nov  1 22:13 xaa
-rw-rw-r-- 1 sansforensics sansforensics 100M Nov  1 22:13 xab
-rw-rw-r-- 1 sansforensics sansforensics  55M Nov  1 22:13 xac
```

To recombine the parts into a single image, we can run cat xaa xab xac > Image.dd.

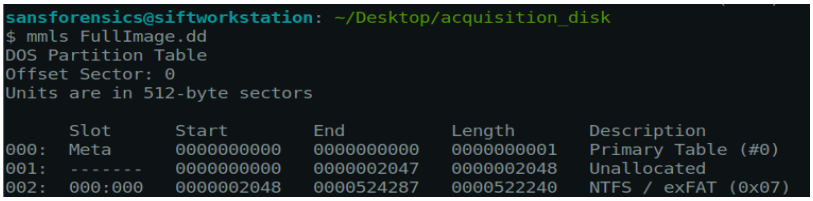
Always analyze a verified copy, not the original disk image, to preserve evidence integrity and ensure it remains admissible in court. Verify copies using cryptographic hashes (MD5 or SHA256) to confirm they are exact replicas of the original.

3.2 Disk Analysis Using Sleuth Kit

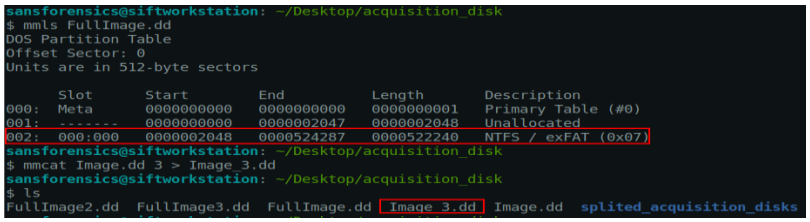
The Sleuth Kit (TSK) is a powerful open-source tool used for disk analysis. It includes many command-line programs that allow investigators to safely examine disks and file systems without changing the original evidence. Here we will use some of these tools. We will work on the image of the previous disk.

img_stat: image type (raw, ewf, etc), size of image, and sector size.

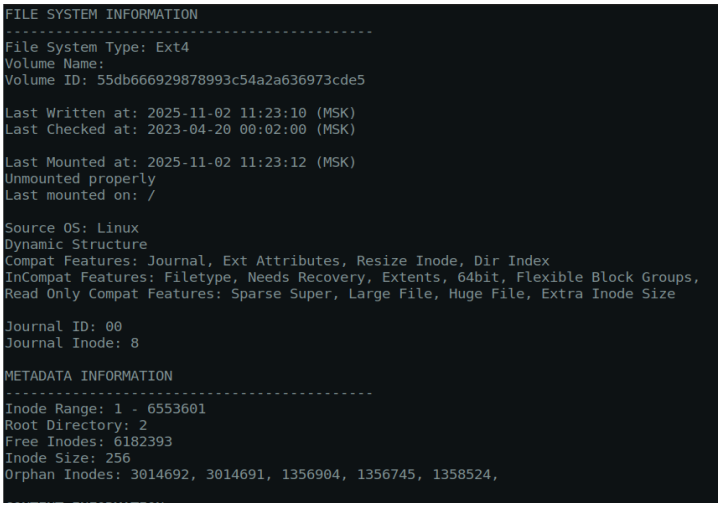
mmls: display partition table of a volume.



mmcat: Extracts (copies) raw data from a partition or volume.



fsstat Extracts a file's content using its metadata address.



We can run fsstat directly on a specific partition image (FullImage_p2.dd) or use it with a sector offset when analyzing a full disk image. fsstat -o <offset> FullImage.dd, for example, fsstat -o 2048 FullImage.dd| less. Here, the -o 2048 option specifies the starting sector of the partition.

`fls` Lists file and directory entries (including deleted ones).

```
sansforensics@siftworkstation: ~/Desktop/acquisition_disk
$ fls -o 2048 FullImage.dd
r/r 4-128-1:    $AttrDef
r/r 8-128-2:    $BadClus
r/r 8-128-1:    $BadClus:$Bad
r/r 6-128-1:    $Bitmap
r/r 7-128-1:    $Boot
d/d 11-144-2:   $Extend
r/r 2-128-1:    $LogFile
r/r 0-128-1:    $MFT
r/r 1-128-1:    $MFTMirr
r/r 9-128-2:    $Secure:$SDS
r/r 9-144-3:    $Secure:$SDH
r/r 9-144-4:    $Secure:$SII
r/r 10-128-1:   $UpCase
r/r 10-128-2:   $UpCase:$Info
r/r 3-128-3:    $Volume
r/r 65-128-2:   Hunt-Evil.pdf
r/r 64-128-2:   Poster_Threat-Intelligence-Consumption.pdf
r/r 66-128-2:   Windows-Forensics-Poster.pdf
V/V 67: $OrphanFiles
```

The second column with a number points to the metadata address of the file, and the column after that is the name of the file.

If we want to see the context of sub-directories, we can run the **-r** option for recursive.

If you want to list or access the contents of a specific directory (for example, /home) inside a disk or partition image, you need to use fls with the inode number of that directory. On the screen, the inode number of /home is (1310721)

```
sansforensics@siftworkstation: ~/Desktop/acquisition_disk
$ sudo fls /dev/ubuntu-vg/ubuntu-lv
d/d 1310721:    home
d/d 11: lost+found
d/d 5242881:    boot
l/l 12: bin
l/l 13: lib
l/l 14: lib32
l/l 15: lib64
l/l 16: libx32
l/l 17: sbin
```

After we see the home of user sansforensics, we can also type the inode of that user's home directory (1321412) to access its contents.

```
sansforensics@siftworkstation: ~/Desktop/acquisition_disk
$ sudo fls /dev/ubuntu-vg/ubuntu-lv 1310721
d/d 1321412:    sansforensics
d/d 1481387:    snasforensics
sansforensics@siftworkstation: ~/Desktop/acquisition_disk
$ sudo fls /dev/ubuntu-vg/ubuntu-lv 1321412
r/r 1321413:    .bashrc
r/r 1321414:    .profile
r/r 1321415:    .bash_logout
d/d 1321416:    .cache
d/d 1321417:    .config
d/d 1321425:    .local
d/d 1321432:    Desktop
d/d 1321433:    Downloads
d/d 1321434:    Templates
d/d 1321435:    Public
d/d 1321436:    Documents
d/d 1321437:    Music
d/d 1321438:    Pictures
```

So we can access the content of any directory in this way.

3.3 Collecting Non-volatile Artifacts

System Information

Retrieves CPU details.

```
sansforensics@siftworkstation: /home
$ cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 94
model name      : Intel(R) Core(TM) i5-6400 CPU @ 2.70GHz
stepping        : 3
microcode       : 0xd6
cpu MHz         : 2712.000
cache size      : 6144 KB
physical id     : 0
siblings        : 1
core id         : 0
cpu cores       : 1
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 22
wp              : yes
```

Retrieves information on mount points and mounted external devices.

```
sansforensics@siftworkstation: /home
$ cat /proc/self/mounts
sysfs /sys sysfs rw,nosuid,nodev,noexec,relatime 0 0
proc /proc proc rw,nosuid,nodev,noexec,relatime 0 0
udev /dev devtmpfs rw,nosuid,relatime,size=1946352k,nr_inodes=486588,mode=755,inode64 0 0
devpts /dev/pts devpts rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000 0 0
tmpfs /run tmpfs rw,nosuid,nodev,noexec,relatime,size=401816k,mode=755,inode64 0 0
/dev/mapper/ubuntu--vg-ubuntu--lv / ext4 rw,relatime 0 0
securityfs /sys/kernel/security securityfs rw,nosuid,nodev,noexec,relatime 0 0
tmpfs /dev/shm tmpfs rw,nosuid,nodev,inode64 0 0
```

Kernel Information

```
sansforensics@siftworkstation: /home
$ uname -r
5.15.0-70-generic
sansforensics@siftworkstation: /home
$ cat /proc/version
Linux version 5.15.0-70-generic (buildd@lcy02-amd64-056) (gcc (Ubuntu 11.3.0-1ubuntu1~22.04)
sansforensics@siftworkstation: /home
$ hostnamectl
 Static hostname: siftworkstation
       Icon name: computer-vm
         Chassis: vm
      Machine ID: da13c52884de466b9904a54213e1d015
         Boot ID: f28dcee0e57c445a98bbe7dc2a3cef04
  Virtualization: vmware
Operating System: Ubuntu 22.04.2 LTS
          Kernel: Linux 5.15.0-70-generic
    Architecture: x86-64
 Hardware Vendor: VMware, Inc.
  Hardware Model: VMware Virtual Platform
```

Collecting User Account Information

The /etc/passwd file on Linux stores the local users' account information

```
sansforensics@siftworkstation: /home
$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
```

cut -d: -f1 /etc/passwd. This command can also help to show only user names.

Collecting Currently Logged-in Users and Login History Information

```
sansforensics@siftworkstation: /home
$ w
 12:19:44 up  5:51,  1 user,  load average: 0.01, 0.07, 0.07
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
sansfore tty2     tty2             Sun11    2days  0.07s  0.06s /usr/libexec/gnome-session-binary --session=ubuntu
sansforensics@siftworkstation: /home
$ last -f /var/log/wtmp
sansfore tty2         tty2             Sun Nov  2 11:25    gone - no logout
reboot   system boot  5.15.0-70-generi Sun Nov  2 11:23   still running
```

Collecting User History

```
sansforensics@siftworkstation: /home
$ history | tail -n 15
  639  uname -r
  640  cat /proc/version
  641  hostnamectl
  642  cat /etc/passwd
  643  cut -d: -f1 /etc/passwd
  644  cut -d: -f1 /etc/passwd | grep *san*
  645  cut -d: -f1 /etc/passwd
  646  w
  647  last -f /var/log/wtmp
  648  w
  649  last -f /var/log/wtmp
  650  history
  651  history | less
  652  history | tail -n 10
  653  history | tail -n 15
```

File Signature

```
$ xxd inside1.jpg | head
00000000: ffd8 ffe0 0010 4a46 4946 0001 0101 012c  ......JFIF.....,
00000010: 012c 0000 ffdb 0043 000a 0707 0807 060a  .,.....C........
00000020: 0808 080b 0a0a 0b0e 1810 0e0d 0d0e 1d15  ................
00000030: 1611 1823 1f25 2422 1f22 2126 2b37 2f26  ...#.%$"."!&+7/&
00000040: 2934 2921 2230 4131 3439 3b3e 3e3e 252e  )4)!"0A149;>>>%.
00000050: 4449 433c 4837 3d3e 3bff db00 4301 0a0b  DIC<H7=>;...C...
00000060: 0b0e 0d0e 1c10 101c 3b28 2228 3b3b 3b3b  ........;("(;;;;
00000070: 3b3b 3b3b 3b3b 3b3b 3b3b 3b3b 3b3b 3b3b  ;;;;;;;;;;;;;;;;
00000080: 3b3b 3b3b 3b3b 3b3b 3b3b 3b3b 3b3b 3b3b  ;;;;;;;;;;;;;;;;
00000090: 3b3b 3b3b 3b3b 3b3b 3b3b 3b3b 3b3b ffc0  ;;;;;;;;;;;;;;..
```

This signature confirms that the file is a JPEG (.jpg) image, regardless of its file extension.

Usage of the Strings Command with dd to extract files

```
sansforensics@siftworkstation: ~/Desktop/acquisition_disk
$ strings -t d FullImage.dd | grep %PDF-
34906112 %PDF-1.4
51675136 %PDF-1.4
136077312 %PDF-1.4
```

Here, we know there are three .pdf files. If we want to extract them, we can run

```
dd if=FullImage.dd of=pdf1.pdf bs=1 skip=34906112 count=$((51675136-34906112)) status=progress
```

```
dd if=FullImage.dd of=pdf2.pdf bs=1 skip=51675136 count=$((136077312-51675136)) status=progress
```

```
dd if=FullImage.dd of=pdf3.pdf bs=1 skip=136077312 status=progress
```

```
sansforensics@siftworkstation: ~/Desktop/acquisition_disk/Output
$ dd if=../FullImage.dd of=pdf2.pdf bs=1 skip=51675136 count=$((136077312-51675136)) status=progress
12486791 bytes (12 MB, 12 MiB) copied, 139 s, 89.8 kB/s^C
12554359+0 records in
12554359+0 records out
12554359 bytes (13 MB, 12 MiB) copied, 139.399 s, 90.1 kB/s
```

```
sansforensics@siftworkstation: ~/Desktop/acquisition_disk/Output
$ dd if=../FullImage.dd of=pdf3.pdf bs=1 skip=136077312 status=progress
3953397 bytes (4.0 MB, 3.8 MiB) copied, 79 s, 50.0 kB/s^C
3964165+0 records in
3964165+0 records out
3964165 bytes (4.0 MB, 3.8 MiB) copied, 79.3253 s, 50.0 kB/s
```

```
sansforensics@siftworkstation: ~/Desktop/acquisition_disk/Output
$ dd if=../FullImage.dd of=pdf1.pdf bs=1 skip=34906112 count=$((51675136-34906112)) status=progress
14135959 bytes (14 MB, 13 MiB) copied, 77 s, 184 kB/s^C
14237652+0 records in
14237652+0 records out
14237652 bytes (14 MB, 14 MiB) copied, 77.6772 s, 183 kB/s
```

Here we successfully extracted the three pdfs files.

```
sansforensics@siftworkstation: ~/Desktop/acquisition_disk/Output
$ ls
pdf1.pdf  pdf2.pdf  pdf3.pdf
```

Examining Cron Jobs for Linux

Cron is a background scheduling service (daemon) in Linux.

- Crontab -l
- cat /etc/crontab
- sudo crontab -u USERNAME -l
- sudo systemctl status cron

All these commands can help us analyze the cron jobs.

```
# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
```

3.4 Log Files Analysis

In digital forensics, log files are one of the most important sources of evidence on a Linux system.
They keep a timeline of system activities, including user actions, system messages, authentication attempts, and software behavior.
Since log files are non-volatile, they remain on the disk even after a reboot, allowing investigators to find traces of attacker activity, privilege escalation, or unauthorized access.

cat /var/log/auth.log | tail -n 10

```
$ cat /var/log/auth.log | tail -n 10
Nov  4 14:52:23 localhost sudo: pam_unix(sudo:session): session opened for user root(uid=0) by (uid=1000)
Nov  4 14:52:23 localhost sudo: pam_unix(sudo:session): session closed for user root
Nov  4 15:06:22 localhost gdm-password]: gkr-pam: unlocked login keyring
Nov  4 15:17:01 localhost CRON[8608]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
Nov  4 15:17:01 localhost CRON[8608]: pam_unix(cron:session): session closed for user root
Nov  4 15:21:02 localhost gdm-password]: pam_unix(gdm-password:auth): authentication failure; logname= uid=0 euid=0 tty=/dev/tty1 ruser= rhost=  user=sansforensics
Nov  4 15:21:06 localhost gdm-password]: pam_unix(gdm-password:auth): authentication failure; logname= uid=0 euid=0 tty=/dev/tty1 ruser= rhost=  user=sansforensics
```

From here, we track user logins, sudo commands, SSH authentication, and privilege escalation.

cat /var/log/syslog | tail -n 10

```
$ cat /var/log/syslog| tail -n 10
Nov  4 15:21:12 localhost gnome-shell[1800]: Window manager warning: last_user_time (29853232) is
nding inaccurate timestamps in messages such as _NET_ACTIVE_WINDOW.  Trying to work around...
Nov  4 15:21:12 localhost gnome-shell[1800]: Window manager warning: W0 appears to be one of the
Nov  4 15:21:30 localhost systemd[1]: fprintd.service: Deactivated successfully.
```

Syslog file contains system events, kernel messages, and background service information.

Here are the most log files we can investigate

Log File	Location	Purpose / Information Contained
System Log	/var/log/syslog or /var/log/messages	Records general system events, kernel messages, and background service information.
Authentication Log	/var/log/auth.log	Tracks user logins, sudo commands, SSH authentication, and privilege escalation.
Kernel Log	/var/log/kern.log	Logs kernel-level messages like driver issues or hardware errors.
Boot Log	/var/log/boot.log	Records boot-time events and services initialized during startup.
Cron Log	/var/log/cron.log or within /var/log/syslog	Lists scheduled jobs, their execution times, and user identities.
Daemon Log	/var/log/daemon.log	Stores output from background services (daemons).
Package Manager Logs	/var/log/apt/history.log or /var/log/yum.log	Tracks software installations, updates, and removals.
Last Login Records	/var/log/wtmp, /var/log/btmp, /var/log/lastlog	Stores user login/logout records, failed logins, and timestamps.
Mail Log	/var/log/mail.log	Contains email system activities (useful in phishing or exfiltration cases).
Secure Log	/var/log/secure (on some distros)	Authentication and security-related events.
Journal Log	journalctl command (systemd-based systems)	Centralized logging system for all services under systemd.

Tools like **cat**, **less**, **more**, **grep**, **awk**, **sed**, **head**, and **tail** are essential for analyzing log files.

3.5 Recover Deleted Bash Histories

Bash stores each user's command history in ~/.bash_history, but it only writes to this file when a shell session ends. This means deleted or missing history may still be recoverable from memory or disk artifacts, making Bash history a valuable source of evidence during digital forensics and incident response.

Recover from memory (if the session is still open)

1. Check currently running shells for history `history`
2. Dump environment variables of the shell: `echo $HISTFILE`

`$HISTFILE` This tells you that your Bash session history is saved in /home/user_name/.bash_history.

Recover deleted history from disk.

1. Use extundelete

```
sudo extundelete /dev/sdX --restore-file /home/user_name/.bash_history
```

- ✓ Replace /dev/sdX with the correct partition (/dev/sda1).
- ✓ extundelete works only on unmounted or read-only mounted partitions.

2. Use testdisk:

1. Install: `sudo apt install testdisk`
2. Run: `sudo testdisk`
3. Navigate through the interface:
   - ✓ Select the disk
   - ✓ Choose the correct partition
   - ✓ Search for deleted files
   - ✓ Restore any recovered .bash_history files

Part 3: Legality of Evidence Handling In Both Environments

Digital forensics on **Windows** and **Linux** systems must follow strict legal standards to ensure that evidence is admissible in court and defensible during investigations. Below is a concise and practical guide covering **what matters legally** and **how to perform it correctly** on both operating systems.

1. Core Legal Principles (Apply to Both Windows and Linux)

1. Chain of Custody

A legal requirement showing the *history of evidence handling*.

Must include:

- Evidence ID
- Who collected it
- Date & time
- Location
- Hash values
- Every transfer, access, or modification

How to implement:

- Assign a unique ID to every disk image, log file, or memory capture.
- Use signed digital chain-of-custody forms.
- Log timestamps and investigator names for all actions.

2. Integrity Verification

Courts require proof that evidence is unaltered.

Tools used:

- Hashes: **SHA-256** or **SHA-512**
- Write blockers (for physical disks)
- Read-only mounting

How to implement:

- Hash before and after imaging (Linux: sha256sum, Windows: Get-FileHash).
- Store originals in read-only storage; perform analysis on copies.

3. Legal Authority & Consent

You must have legal permission to acquire evidence.

Examples:

- Company-approved internal investigation
- Court order or warrant (criminal cases)
- User consent (depends on policy & jurisdiction)

4. Privacy & Compliance

Investigators must avoid violating:

- PII protection regulations
- Local cybercrime laws

How to stay compliant:

- Minimize personal data when possible
- Encrypt evidence storage
- Restrict access to authorized personnel only

2. Legality of Evidence Handling in Windows

Windows investigations often involve:

- NTFS file systems
- Registry hives
- Event logs
- Prefetch files
- Memory captures

3. Legality of Evidence Handling in Linux

Linux investigations often include:

- EXT4/XFS/BTRFS file systems
- System logs
- Bash history
- Configuration files
- Memory & network captures