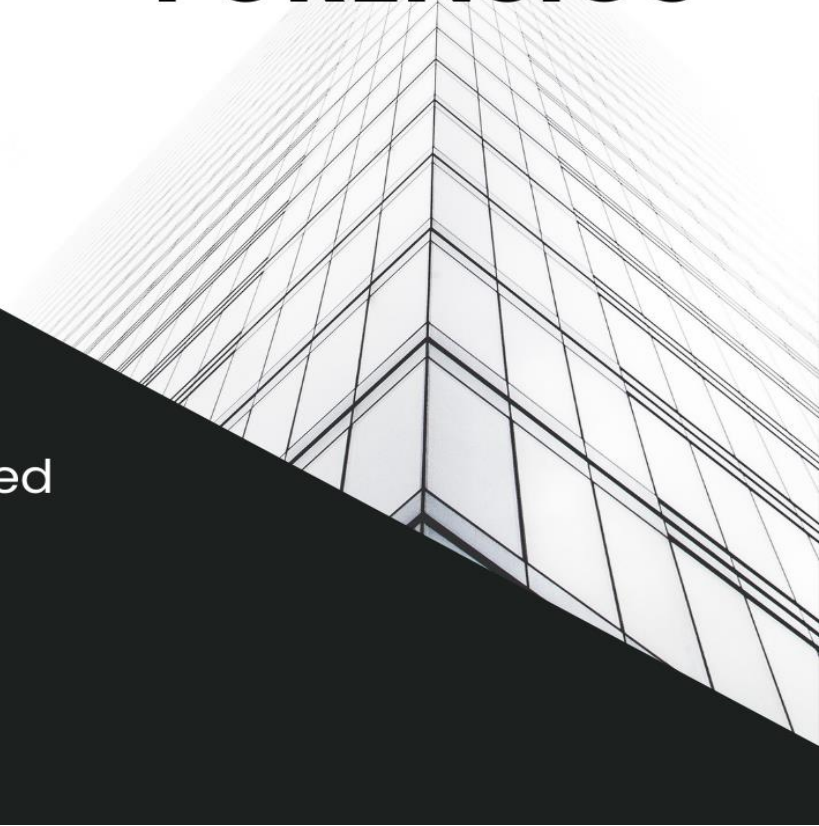
WINDOWS AND LINUX ARTIFACT DEEP DIVE

INFRASTRUCTURE AND SECURITY – FORENSICS INVESTIGATOR

DEPI GRADUATION PROJECT

REPORT TWO

LINUX FORENSICS



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.

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. 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. Use the `lsmod` command to remove, and run `sudo rmmod lime` after finishing.

```
root@ubuntu:/media/abdo/Forensics# lsmod | grep lime
lime                   12288  1
```

```
abdo@ubuntu:/media/abdo/Forensics$ ls -alh
total 2.0G
drwxr-xr-x  11 abdo abdo   32K Nov  6 13:41  .
drwxr-x---+  4 root root 4.0K Nov  6 13:39  ..
drwxr-xr-x   2 abdo abdo   32K Oct 13 17:48 'AppCompatCacheParser output'
drwxr-xr-x   2 abdo abdo   32K Oct 14 22:03 'AppCompatCacheParser output 2'
drwxr-xr-x   2 abdo abdo   32K Oct 12 15:55  AutomaticJumpLists.csv
-rwxr-xr-x   1 abdo abdo 2.0G Nov  6 13:44  dump_mem
drwxr-xr-x   2 abdo abdo   32K Oct 13 20:48  JumpLists_dumping
drwxr-xr-x   2 abdo abdo   32K Oct  7 19:17 'Memory Acquisition'
drwxr-xr-x   2 abdo abdo   32K Oct  7 19:53  memory_aqcuisition
drwxr-xr-x   2 abdo abdo   32K Oct 28 12:19  shellbags_report.txt
drwxr-xr-x   2 abdo abdo   32K Oct  2 08:07 'System Volume Information'
drwxr-xr-x  19 abdo abdo   32K Oct  7 18:04  Tools
```

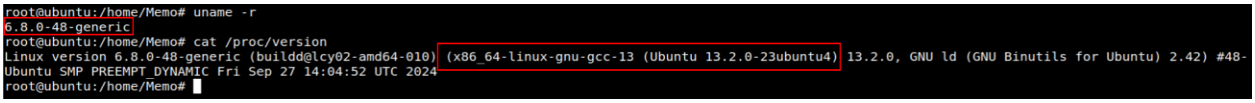
Here, we have RAM acquisition; we can investigate using volatility or other analysis tools.

3. Memory Acquisition Using AVML

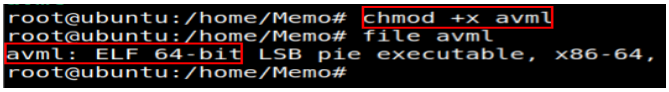
A portable volatile memory acquisition tool for Linux. AVML is an X86_64 userland volatile memory acquisition tool written in Rust, intended to be deployed as a static binary. AVML can be used to acquire memory without knowing the target OS distribution or kernel a priori. No on-target compilation or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

We can download the tool from the repo: https://github.com/microsoft/avml.

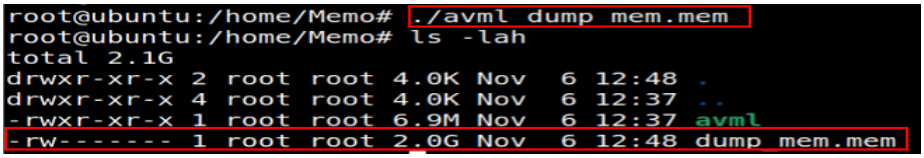
We check the kernel versions with uname -r



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.

4. Memory Analysis using Volatility

After acquiring memory, Linux analysis with Volatility requires a **kernel-specific profile**. Using tools like **dwarfdump** and the build process in Volatility's Linux tools, a custom profile is created to accurately parse processes, memory structures, commands, and logs for forensic investigation.

Here are the steps:

1. Download Volatility

git clone https://github.com/volatilityfoundation/volatility

2. Build a Custom Linux Profile

Linux memory analysis often requires a profile matching the kernel version.

1. **Navigate to the Linux tools directory:**

   `cd ./volatility/tools/linux`

2. **Install dwarfdump**

   dwarfdump is required to extract debug symbols from the kernel for profile creation:

   `sudo apt install dwarfdump`

3. **Build the kernel debug module:**

   `make`

4. **Identify the current kernel version:**

   `uname -a`

5. **Create a custom profile archive:**

   `sudo zip [DISTRO_KERNEL].zip ./volatility/tools/linux/module.dwarf /boot/System.map-[KERNEL_VERSION]`

```
[demo@ubuntu Desktop]$ sudo zip Ubuntu 5.3.0-46-generic.zip ./volatility/tools/linux/module.dwarf /boot/System.map-5.3.0-46-generic
updating: volatility/tools/linux/module.dwarf (deflated 91%)
  adding: boot/System.map-5.3.0-46-generic (deflated 79%)
```

6. **Move the profile to Volatility's overlays directory:**

   `mv [DISTRO_KERNEL].zip ./volatility/overlays/plugins/linux`

```
[demo@ubuntu Desktop]$ mv Ubuntu 5.3.0-46-generic.zip volatility/volatility/plugins/overlays/linux/
[demo@ubuntu Desktop]$ cd volatility/
```

7. **Verify profile is available**

```
[demo@ubuntu volatility]$ python vol.py --info | more
Volatility Foundation Volatility Framework 2.6.1
```

```
Profiles
--------
LinuxUbuntu_5_3_0-46-genericx64 - A Profile for Linux Ubuntu_5.3.0-46-generic x64
```

Now that memory acquisition is complete, we can begin the memory analysis phase and use Volatility's Linux plugins to extract and examine key forensic artifacts such as processes, network activity, open files, and in-memory Bash commands.

linux_pslist: Shows a list of all processes running at the time of memory acquisition.

```
[demo@ubuntu volatility]$ python vol.py -f ../memory.dmp --profile=LinuxUbuntu_5_3_0-46-genericx64 linux_pslist | more
Volatility Foundation Volatility Framework 2.6.1
```

Offset	Name	Pid	PPid	Uid	Gid	DTB	Start Time
0xffff91437441bb00	systemd	1	0	0	0	0x0000000232b14000	2020-04-22 18:51:57 UTC+0000
0xffff914374418000	kthreadd	2	0	0	0	----------	2020-04-22 18:51:57 UTC+0000
0xffff91437441d880	rcu_gp	3	2	0	0	----------	2020-04-22 18:51:57 UTC+0000
0xffff914374419d80	rcu_par_gp	4	2	0	0	----------	2020-04-22 18:51:57 UTC+0000
0xffff91437443d880	kworker/0:0H	6	2	0	0	----------	2020-04-22 18:51:57 UTC+0000
0xffff914374443b00	mm_percpu_wq	9	2	0	0	----------	2020-04-22 18:51:57 UTC+0000
0xffff914374440000	ksoftirqd/0	10	2	0	0	----------	2020-04-22 18:51:57 UTC+0000
0xffff914374445880	rcu_sched	11	2	0	0	----------	2020-04-22 18:51:57 UTC+0000
0xffff91437441d80	migration/0	12	2	0	0	----------	2020-04-22 18:51:57 UTC+0000
0xffff914374489d80	idle_inject/0	13	2	0	0	----------	2020-04-22 18:51:57 UTC+0000
0xffff914374009d80	cpuhp/0	14	2	0	0	----------	2020-04-22 18:51:57 UTC+0000
0xffff91437400bb00	cpuhp/1	15	2	0	0	----------	2020-04-22 18:51:57 UTC+0000
0xffff914374008000	idle_inject/1	16	2	0	0	----------	2020-04-22 18:51:57 UTC+0000
0xffff91437400d880	migration/1	17	2	0	0	----------	2020-04-22 18:51:57 UTC+0000
0xffff914374010000	ksoftirqd/1	18	2	0	0	----------	2020-04-22 18:51:57 UTC+0000

linux_pstree: Displays all processes in a hierarchical tree structure.

```
[demo@ubuntu volatility]$ python vol.py -f ../memory.dmp --profile=LinuxUbuntu_5_3_0-46-genericx64 linux_pstree | more
Volatility Foundation Volatility Framework 2.6.1
```

Name	Pid	Uid
systemd	1	
.systemd-journal	421	
.systemd-udevd	437	
.vmware-vmblock-	544	
.systemd-timesyn	606	62583
.systemd-resolve	608	101
.VGAuthService	619	
.vmtoolsd	623	
.rsyslogd	628	102
.accounts-daemon	635	
.ModemManager	644	
.networkd-dispat	647	
.systemd-logind	654	
.cron	656	

linux_netstat: Shows active and closed network connections, listening ports, and associated processes.

```
[demo@ubuntu volatility]$ python vol.py -f ../memory.dmp --profile=LinuxUbuntu_5_3_0-46-genericx64 linux_netstat | more
Volatility Foundation Volatility Framework 2.6.1

UDP      127.0.0.53      :   53 0.0.0.0         :    0                 systemd-resolve/608
TCP      127.0.0.53      :   53 0.0.0.0         :    0 LISTEN          systemd-resolve/608
UDP      0.0.0.0         : 5353 0.0.0.0         :    0                 avahi-daemon/684
UDP      ::              : 5353 ::              :    0                 avahi-daemon/684
UDP      0.0.0.0         :35931 0.0.0.0         :    0                 avahi-daemon/684
UDP      ::              :41094 ::              :    0                 avahi-daemon/684
```

linux_lsof: Lists open files and the processes that opened them.

```
[demo@ubuntu volatility]$ python vol.py -f ../memory.dmp --profile=LinuxUbuntu_5_3_0-46-genericx64 linux_lsof | more
Volatility Foundation Volatility Framework 2.6.1

Offset             Name                             Pid      FD       Path
------------------ -------------------------------- -------- -------- ----
0xffff91437441bb00 systemd                             1        0 /dev
0xffff91437441bb00 systemd                             1        1 /dev
0xffff91437441bb00 systemd                             1        2 /dev
0xffff91437441bb00 systemd                             1        3 /dev
0xffff91437441bb00 systemd                             1        4 anon_inode:[11406]
0xffff91437441bb00 systemd                             1        5 anon_inode:[11406]
0xffff91437441bb00 systemd                             1        6 anon_inode:[11406]
0xffff91437441bb00 systemd                             1        7 /sys/fs/cgroup/unified
0xffff91437441bb00 systemd                             1        8 anon_inode:[11406]
0xffff91437441bb00 systemd                             1        9 socket:[24703]
0xffff91437441bb00 systemd                             1       10 anon_inode:[11406]
0xffff91437441bb00 systemd                             1       11 /proc
```

linux_bash: Extracts in-memory Bash command history from running shells.

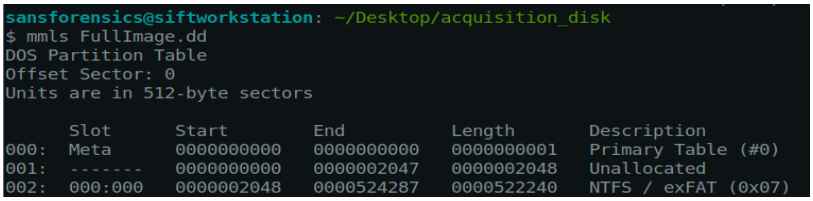
```
[demo@ubuntu volatility]$ python vol.py -f ../memory.dmp --profile=LinuxUbuntu_5_3_0-46-genericx64 linux_bash | more
Volatility Foundation Volatility Framework 2.6.1

Pid      Name                 Command Time                   Command
-------- -------------------- ------------------------------ -------
    2960 bash                 2020-04-22 18:52:01 UTC+0000   cd python3.6/site-packages/
    2960 bash                 2020-04-22 18:52:01 UTC+0000   ll
    2960 bash                 2020-04-22 18:52:01 UTC+0000   sudo apt install python3-pip
    2960 bash                 2020-04-22 18:52:01 UTC+0000   chmod 755 vol.py
    2960 bash                 2020-04-22 18:52:01 UTC+0000   clear
    2960 bash                 2020-04-22 18:52:01 UTC+0000   ./vol.py --info | grep Win10
    2960 bash                 2020-04-22 18:52:01 UTC+0000   sudo apt install python3
    2960 bash                 2020-04-22 18:52:01 UTC+0000   sudo su -
    2960 bash                 2020-04-22 18:52:01 UTC+0000   clear
    2960 bash                 2020-04-22 18:52:01 UTC+0000   sudos u -
    2960 bash                 2020-04-22 18:52:01 UTC+0000   sudo su -
    2960 bash                 2020-04-22 18:52:01 UTC+0000   clear
    2960 bash                 2020-04-22 18:52:01 UTC+0000   pwd
```

5. Disk Analysis Using Sleuth Kit

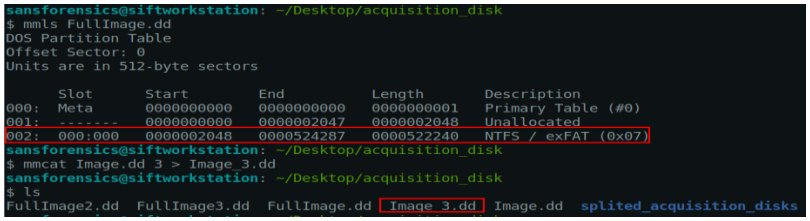
The Sleuth Kit (TSK) is a powerful open-source tool used for disk analysis. It includes many command-line programs that allow investigators to safely examine disks and file systems without changing the original evidence. Here we will use some of these tools. We will work on the image of the previous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

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

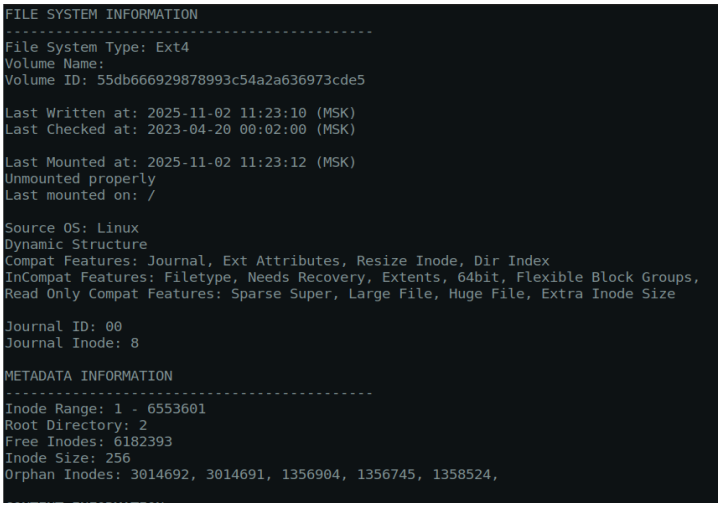
mmls: display partition table of a volume.



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.

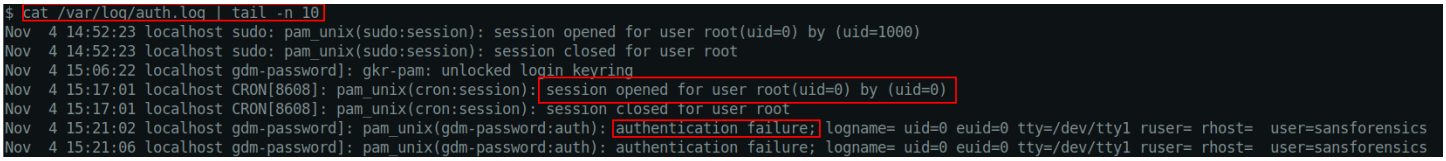
```
sansforensics@siftworkstation: ~/Desktop/acquisition_disk
$ sudo fls /dev/ubuntu-vg/ubuntu-lv 1310721
d/d 1321412:    sansforensics
d/d 1481387:    snasforensics
sansforensics@siftworkstation: ~/Desktop/acquisition_disk
$ sudo fls /dev/ubuntu-vg/ubuntu-lv 1321412
r/r 1321413:    .bashrc
r/r 1321414:    .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.

6. Log Files Analysis

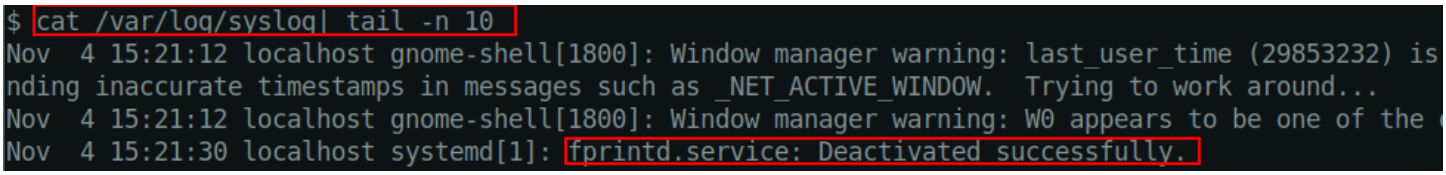
In digital forensics, log files are one of the most important sources of evidence on a Linux system.
They keep a timeline of system activities, including user actions, system messages, authentication attempts, and software behavior.
Since log files are non-volatile, they remain on the disk even after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

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



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

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



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.

7. 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